

Programsko inženjerstvo

Ak. god. 2020./2021.

Planinarski Dnevnik

Dokumentacija, Rev. 1.8

Grupa: *RuntimeTerror*

Voditelj: *Ivan Martinović*

Datum predaje: *13.11.2020.*

Nastavnik: *Katarina Labor*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
2.1 Primjeri sličnih rješenja	7
2.2 Moguće nadogradnje projektnog zadatka	9
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	13
3.1.2 Sekvencijski dijagrami	31
3.2 Ostali zahtjevi	36
4 Arhitektura i dizajn sustava	37
4.1 Baza podataka	41
4.1.1 Opis tablica	43
4.1.2 Dijagram baze podataka	52
4.2 Dijagram razreda	53
4.2.1 Konceptualni model dijagrama razreda	53
4.2.2 Implementacijski dijagrami razreda - model sustava	54
4.3 Dijagram stanja	68
4.4 Dijagram aktivnosti	72
4.5 Dijagram komponenti	74
5 Implementacija i korisničko sučelje	76
5.1 Korištene tehnologije i alati	76
5.2 Ispitivanje programskog rješenja	78
5.2.1 Ispitivanje komponenti	78
5.2.2 Ispitivanje sustava	88
5.3 Dijagram razmještaja	95
5.4 Upute za puštanje u pogon	96

6 Zaključak i budući rad	105
Popis literature	107
Indeks slika i dijagrama	109
Dodatak: Prikaz aktivnosti grupe	110

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	I.M	14.10.2020.
0.2	Dodan opis projektnog zadatka	J.K, H.L, N.K, I.M, D.K	14.10.2020.
0.3	Dodana osnovna verzija funkcionalnih zahtjeva	J.K, I.M, M.R	14.10.2020.
0.3.1	Ispravljeni funkcionalni zahtjevi	I.M	17.10.2020
0.4	Dodana početna verzija obrazaca uporabe	D.K, N.K	19.10.2020
0.4.1	Ispravljeni, povezani i dodani novi scenariji obrazaca uporabe	I.M	21.10.2020.
0.4.2	Izmijenjeni i dodani novi obrasci uporabe	I.M	25.10.2020
0.5	Dodani dijagrami obrazaca uporabe	I.M	25.10.2020
0.6	Dodani sekvencijski dijagrami	J.K, H.L, I.M	29.10.2020
0.7	Dodan opis baze podataka zajedno s tablicama	M.R, I.M	05.11.2020
0.7.1	Dodani nefunkcionalni zahtjevi te zahtjevi domene primjene	D.K, N.K	05.11.2020
0.8	Dodan opis arhitekture sustava	J.K	5.11.2020
0.8.1	Ažuriran opis baze podataka, preimenovane tablice i dorađeni atributi	I.M	10.11.2020
0.8.2	Zamijenjen dijagram baze podataka	M.R	10.11.2020
0.8.3	Ispravljeni opisi i dijagrama, ispravljene gramatičke pogreške i uklonjeni dijelovi dokumentacije nevažni za prvu predaju	I.M	11.11.2020
0.9	Dodani dijagrami razreda te opisi dijagrama razreda	I.M	11.11.2020
1.0	Završna verzija dokumentacije - prva predaja	I.M	13.11.2020
1.1	Popravljeni djagrami obrazaca uporabe	D.K, N.K	06.01.2021.

Rev.	Opis promjene/dodataka	Autori	Datum
1.2	Dodan dijagram komponenti	D.K, N.K	07.01.2021.
1.3	Dodani selenium testovi	I.M, D.K	10.01.2021.
1.4	Dodani JUnit testovi	I.M	11.01.2021.
1.4.1	Dodan zaključak i budući rad	I.M, N.K	11.01.2021.
1.4.2	Opisano poglavlje "Korištene tehnologije i alati"	I.M, D.K	11.01.2021.
1.5	Dodani implementacijski dijagrami razreda	I.M	12.01.2021.
1.6	Dodani dijagrami stanja, aktivnost i razmještaja	M.R	13.01.2021.
1.7	Postavio opis puštanja aplikacije u pogon	I.M, L.R	13.01.2021.
1.8	Prepravljen dijagram i ispravljeni opisi baze podataka	D.K	13.01.2021.

2. Opis projektnog zadatka

U današnje vrijeme većina ljudi živi užurbanim tempom, stoga svaki slobodan trenutak žele iskoristiti za odmor i rekreaciju. Mnoge ljude privlači boravak na svježem zraku te kao rezultat toga, sve više ljudi izabire planinarenje kao jednu od brojnih mogućnosti koje im se nude. Međutim, planinari pri odabiru rute za svoje planinarske izlete često nemaju dovoljno informacija pa se koriste usmenom predajom i nagađanjima. Tako planinari, osobito planinari rekreativci, nailaze na različite probleme od kojih su najčešći krive informacije o stazama i rutama ili planinarski domovi bez odgovarajuće infrastrukture. Upravo zbog toga pokrenut je projekt čiji je cilj razvoj i evolucija programskog proizvoda, odnosno web aplikacije „Planinarski dnevnik“. Aplikacija će uvelike pomoći planinarima u organiziranju svojih planinarskih izleta, ali i ponuditi točne informacije o rutama na pojedinim izletima te povezati planinare poznanike u vlastitu planinarsku zajednicu. Osim toga, planinari će moći pretraživati i planinarske domove koji se nalaze na odabranim stazama, a za svaki dom će biti prikazane koje pogodnosti on nudi (prenoćište, topao obrok, pitka voda, struja, grijanje itd.).

Opseg projektnog zadatka sadrži sve aktivnosti i zadatke koji su vezani uz izradu aplikacije. Za početak se radi analiza aplikacije kako bi se utvrdilo koliko će okvirno vremena biti potrebno za izradu određene komponente aplikacije. U tome dobrom dijelom pomažu dnevnik sastajanja i dnevnik aktivnosti koji služe kao kontrolne točke pomoću kojih se vidi ide li daljnji napredak aplikacije u dobrom smjeru. Na tim sastancima prisutni su asistenti koji su stručnjaci za ovo područje i svojim savjetima višestruko pomažu timu.

Za izradu „Planinarskog dnevnika“ predviđen je vremenski period od 13 tijedana, odnosno jednog fakultetskog semestra. Radi se analiza interesnih sudionika s namjerom da se što točnije odredi broj korisnika koji će biti zainteresirani za korištenje aplikacije. Svrha same aplikacije je educirati studente na fakultetu pa shodno tome ne postoje troškovi prilikom izrade iste. Krajnji cilj je potpuno razvijena aplikacija s ispravnom programskom potporom koja sadrži sve zahtijevane

komponente i podržava rad više korisnika u stvarnom vremenu. Kada je navedeno postignuto, aplikacija je spremna za lansiranje na tržište kako bi se korisnici mogli njome služiti.

Aplikacija „Planinarski dnevnik“ zasigurno će biti najzanimljivija planinarima kojima je i namijenjena, ali također i mnogobrojnim ustanovama poput planinarskih domova kojima će omogućiti promociju u širem krugu korisnika. Potencijalno bi moglo doći do povećanja prihoda kao rezultat brojnih usluga koje planinarski domovi pružaju, ali i poboljšanja kvalitete istih sukladno s porastom broja planinara koji ih posjećuju. Proširit će se opseg planinarskog turizma na manje poznata područja tako što će postati vidljiva širem krugu korisnika aplikacije. Također, korisnost ove aplikacije odrazit će se na HGSS (Hrvatska gorska služba spašavanja) koja će efikasnije saznati sve potrebne informacije o kretanju i ruti planinara u slučaju nesreće ili nestanka. Evidentno je da će se područje pretrage znatno smanjiti jer će planinar unaprijed odrediti rutu svojeg kretanja.

Pokretanjem aplikacije svakom korisniku prvotno će biti dodijeljena uloga Gost koja omogućava pretraživanje postojećih planinarskih domova prema dostupnoj infrastrukturi (pitka voda, hrana, prenoćište...) ili pretraživanje planinarskih staza prema zahtjevnosti, trajanju ili duljini. Za sve daljnje aktivnosti korisnik će se morati registrirati u sustav tako što će u predloženu formu za registraciju unijeti osobne podatke:

- ime
- prezime
- e-mail
- lozinku
- sliku (opcionalno)
- nešto više o sebi (opcionalno)
- datum rođenja (opcionalno)
- mjesto stanovanja (opcionalno)

Nakon što se korisnik registrira dodijelit će mu se uloga Planinar i moći će pristupiti vlastitom profilu. Omogućit će mu se pregled i uređivanje osobnih podataka te u krajnjem slučaju uklanjanje korisničkog računa. Planinar može uspostaviti odnos s ostalim registriranim planinarima tako što šalje zahtjeve za "priateljstvom", odnosno zahtjeve za dodavanje na popis vlastite planinarske zajednice, ali i na

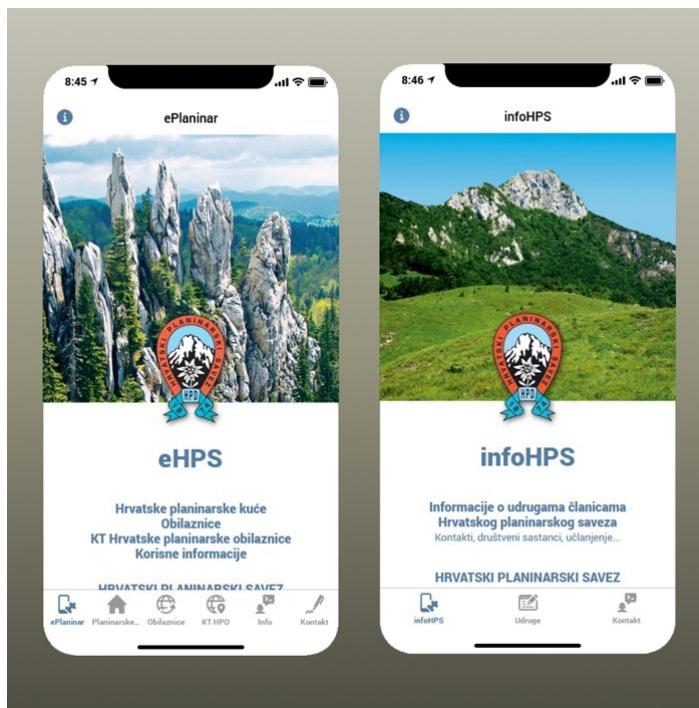
način da ostale članove svoje planinarske zajednice pozove na određeni događaj. Prema unaprijed određenom predlošku dopušta se stvaranje vlastitih planinarskih staza kao i vlastitih događaja. Uz to se nudi i mogućnost ocjenjivanja stvorenih planinarskih staza kao i prijava netočnih ili nepreciznih informacija vezanih uz pojedine staze, što može biti od velike koristi svim planinarima, osobito početnicima koji na osnovu najviše ocjene mogu odabrati svoju željenu stazu. Na naslovniči će biti prikazane objave prijatelja planinara, a na zidu obavijesti će biti vidljiv popis prihvaćenih ili odbijenih pozivnica te prihvaćenih ili odbijenih zahtjeva za prijateljstvom. Nudi se i svrstavanje planinarske staze ili doma na popis željenih te dodavanje ranije odraćenih planinarskih staza u osobnu arhivu. Dolaskom planinara na cilj evidentirat će se njihovo prisutstvo, a nakon određenog broja osvojenih vrhova ostvarit će pravo na bedž kao jednu vrstu motivacije za još veću aktivnost u budućnosti.

Sustav nadgleda Administrator koji ima najveće ovlasti. Ukoliko neki korisnik (planinar) ne poštuje pravila ponašanja, administrator ima pravo obrisati njegov korisnički račun. On će zaprimati primjedbe od korisnika na određene staze te će ovisno o količini netočnih informacija odlučiti hoće li staza biti izmijenjena ili uklonjena s liste. Time se sprječava ponavljanje istih pogrešaka u budućnosti i aplikacija će biti sve točnija i vjerodostojnija.

2.1 Primjeri sličnih rješenja

Slične implementacije rješenja projektnog zadatka već postoje. Na području Republike Hrvatske možemo izdvojiti iduće:

1. Kao prvi primjer navodimo aplikaciju **eHPS** koja je razvijena pod pokroviteljstvom Hrvatskog planinarskog saveza. Njena svrha je omogućavanje korisniku efikasno pretraživanje podataka o svim planinarskim domovima, kućama i skloništima koji postoje na području Republike Hrvatske. Također pruža uslugu iščitavanja i proučavanja podataka o svim kontrolnim točkama i dosad otvorenim planinarskim obilaznicama.
2. Druga slična aplikacija je **infoHPS** koja pruža uslugu pretraživanja postojećih planinarskih udruga koje su članice Hrvatskog planinarskog saveza. Za svaku traženu udrugu omogućuje prikaz informacija bitnih za korisnika poput naziva, OIB-a udruge, email-a itd.



Slika 2.1: Primjeri sličnih aplikacija - eHPS i infoHPS

Na području SAD-a aplikacija **Mountain project** nudi korisnicima pretraživanje postojećih planinarskih ruta, čitanje novosti i razmjenjivanje poruka između prijavljenih korisnika.



Slika 2.2: Primjer slične aplikacije - Mountain Project

2.2 Moguće nadogradnje projektnog zadatka

Postoje brojne funkcionalnosti kojima bi se mogla nadograditi i proširiti postojeća aplikacija te ispraviti eventualne nepravilnosti. Jedna od mogućnosti je implementacija „Chat-a“ za razmjenu poruka i iskustava među planinarima koji pripadaju istoj planinarskoj zajednici. Uz to, mogao bi se dodati i neformalni forum gdje bi svi planinari mogli podijeliti svoja iskustva, doživljaje i preporuke ostatku planinarske zajednice. Aplikacija bi trebala imati i mogućnost instaliranja na pametne satove koji su postali neizostavni dio planinarske i sportske opreme. Također bi bilo korisno kad bi korisnici odlaskom na naslovnu stranu aplikacije mogli vidjeti aktualne novosti, događanja iz planinarskog svijeta te preporučene izlete u skladu s vremenskim uvjetima. Svaki planinar mora imati odgovarajuću opremu prije nego što kreće na izlet pa bi oglašavanje i prodaja planinarske opreme bio izvrstan dodatak aplikaciji. Registrirani planinar bi mogao postaviti oglas sa slikom i opisom opreme koju prodaje, cijenom i lokacijom na kojoj se nalazi. Sadašnja verzija aplikacije sadrži unesene izlete namijenjene većinom za pješačke rute. U budućnosti bi se aplikacija mogla proširiti dodavanjem ruta za bicikliranje ili čak i skijanje. Još jedna od korisnih funkcionalnosti bila bi uvođenje uloge „planinarski dom“. Uloga bi planinarskim domovima omogućila kreiranje vlastitih događaja kao što su organizirani izleti, zabave i slično.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj (FER)
2. Razvojni tim
3. Administrator
4. Planinari (korisnici aplikacije)
5. Zaposlenici planinarskih domova

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) pregledati popis planinarskih staza
 - (b) pretraživati planinarske staze
 - i. prema zemljopisnom položaju
 - ii. prema prosječnom trajanju pješačenja za određenu stazu
 - iii. prema zahtjevnosti planinarske staze
 - (c) pregledati popis planinarskih domova
 - (d) pretraživati planinarske domove
 - i. prema dostupnoj infrastrukturi (voda, prenoćište, struja, hrana, internet)
 - ii. prema zemljopisnom položaju
 - (e) se registrirati u sustav kao planinar tako što popuni formu za registraciju
2. Planinar (inicijator) može:
 - (a) prijaviti se u sustav kao planinar
 - (b) odjaviti se iz sustava
 - (c) upravljati vlastitim korisničkim računom
 - i. pregledati osobne podatke

- ii. uređivati osobne podatke
 - iii. ukloniti korisnički račun
- (d) pretraživati korisnike prema imenu i prezimenu
- (e) upravljati zahtjevima za prijateljstvo
- i. poslati zahtjev za prijateljstvom (zahtjev za dodavanjem drugog planinara na popis vlastite planinarske zajednice)
 - ii. prihvati zahtjev za prijateljstvom
 - iii. pregledati pristigle zahtjeve za prijateljstvom
 - iv. vidjeti obavijest ako je drugi planinar prihvatio njegov zahtjev
- (f) pregledati popis planinara u vlastitoj planinarskoj zajednici
- (g) upravljati vlastitim planinarskim stazama
- i. stvoriti vlastitu planinarsku stazu prema unaprijed definiranom predlošku
 - ii. pregledati staze koje je stvorio
 - iii. obrisati vlastitu planinarsku stazu ukoliko je ona privatna
- (h) pregledati popis željenih planinarskih staza (favoriti)
- (i) dodati planinarsku stazu na popis željenih
- (j) ocjenjivati stvorene planinarske staze drugih planinara
- (k) prijaviti netočne i neprecizne informacije vezane uz planinarske staze
- (l) stvoriti događaj vidljiv na naslovnicu na koji može
- i. pozvati korisnike aplikacije s popisa vlastite planinarske zajednice
 - ii. pregledati popis ljudi koji dolaze na kreirani događaj
- (m) na naslovnicu vidjeti nove objave korisnika s popisa vlastite planinarske zajednice
- i. kreirani događaji
 - ii. ostvareni bedževi
- (n) dodati ranije odradene planinarske staze u arhivu
- (o) dodati ranije posjećene planinarske domove u arhivu
- (p) obzirom na svoju aktivnost zaraditi određeni bedž koji se prikazuje na njegovom profilu
- (q) kontaktirati administratora u slučaju potrebe za stvaranjem novog planinarskog doma ili promjene infrastrukture već postojećeg

3. Administrator (inicijator) može:

- (a) obrisati korisničke račune
- (b) upravljati planinarskim stazama

- i. pregled prijavljenih netočnih i nepreciznih informacija vezanih uz objavljene planinarske staze
- (c) upravljati planinarskim domovima
 - i. pregled prijavljenih netočnih i nepreciznih informacija vezanih uz objavljene planinarske domove
 - ii. stvaranje novog planinarskog doma
 - iii. obrisati postojeći planinarski dom
- (d) pregledati poruke koje su poslali planinari

4. Baza podataka (sudionik):

- (a) komunicira s cjelokupnim sustavom
- (b) pohranjuje sve podatke nužne za uspješno funkcioniranje sustava

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 -Registracija

- **Glavni sudionik:** Neregistrirani korisnik aplikacije (posjetitelj)
- **Cilj:** Stvaranje novog korisničkog računa s ulogom "planinar"
- **Sudionici:** Baza podataka
- **Preduvjet:** /
- **Opis osnovnog tijeka:**
 1. Neregistrirani korisnik pokrene aplikaciju
 2. Neregistrirani korisnik odabere opciju za registraciju i unosi osobne podatke
 3. Sustav validira podatke i sprema ih u bazu podataka u slučaju ispravnosti
 4. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 3.a Unos e-maila koji se već koristi, unos e-maila koji ne zadovoljava propisani format e-maila, neispunjavanje svih potrebnih polja forme za registraciju, lozinka i ponovljena lozinka se ne podudaraju ili lozinka ne zadovoljava minimalne zahtjeve prihvatljivosti
 1. Sustav obavještava korisnika o neispravnim poljima i ostaje na istoj stranici registracije, sva polja forme za registraciju ostaju popunjena unesenim podacima, osim lozinke koju je nužno ponovno unijeti
 2. Korisnik ispravi neispravne podatke te završava unos ili odustaje od registracije

UC2 -Prijava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pristup korisničkom računu i svim funkcionalnostima aplikacije namijenjenih ulozi koju posjeduju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran u sustav kao planinar
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju za prijavu
 2. Korisnik unosi ispravan e-mail i lozinku
 3. Prijava je uspješno obavljena i otvara se naslovna stranica
- **Opis mogućih odstupanja:**

3.a Neispravno korisničko ime i/ili lozinka

1. Sustav javlja pogrešku, ostaje na istoj stranici za prijavu te očisti sva polja forme za prijavu

UC3 -Odjava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Odjaviti se sa svog korisničkog profila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju za odjavu iz sustava
 2. Sustav odjavljuje korisnika, preusmjerava ga na naslovnu stranicu i dodjeli mu ulogu gost

UC4 -Pretraživanje planinarskih staza

- **Glavni sudionik:** Korisnik
- **Cilj:** Pretražiti postojeće planinarske staze prema unaprijed definiranim kriterijima
- **Sudionici:** Baza podataka
- **Preduvjet:** /
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju pretraživanja planinarskih staza
 2. Otvara se stranica za pretragu planinarskih staza s više načina pretraživanja
 - (a) Prema nazivu staze
 - (b) Prema zemljopisnom položaju
 - (c) Prema prosječnom trajanju i/ili zahtjevnosti
 3. Korisnik odabere način pretraživanja te odabere gumb za pretragu
 4. Sustav dohvaća sve podatke iz baze podataka koji zadovoljavaju upit pretrage
 5. Prikazuju se postojeće staze koje ispunjavaju zahtjeve pretrage

UC5 -Pretraživanje planinarskih domova

- **Glavni sudionik:** Korisnik
- **Cilj:** Pretražiti postojeće planinarske domove prema unaprijed definiranim kriterijima
- **Sudionici:** Baza podataka

- **Preduvjet:** /

- **Opis osnovnog tijeka:**

1. Korisnik odabere opciju za pregled i pretraživanje planinarskih domova
2. Otvara se stranica za pretragu planinarskih domova s više načina pretraživanja
 - (a) Prema zemljopisnom položaju
 - (b) Prema dostupnoj infrastrukturi (hrana, pitka voda, prenoćište, internet)
3. Korisnik odabere kategoriju pretrage i/ili unosi naziv doma
4. Sustav dohvaća sve podatke iz baze podataka koji zadovoljavaju upit pretrage
5. Prikazuju se postojeći planinarski domovi koji ispunjavaju zahtjeve pretrage

UC6 -Pregled korisničkog računa

- **Glavni sudionik:** Korisnik

- **Cilj:** Omogućiti korisniku pregled vlastitih osobnih podataka

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Moj profil"
2. Prikazuje se profil korisnika s njegovim osobnim podacima

UC6.1 -Izmjena korisničkog računa

- **Glavni sudionik:** Registrirani korisnik

- **Cilj:** Izmjena osobnih podataka

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Moj profil"
2. Prikazuju se korisnički podaci planinara
3. Korisnik odabire opciju "Uredi osobne podatke"
4. Sustav prikazuje podatke korisnika u sučelju prigodnom za promjenu
5. Korisnik promijeni podatke i potvrđuje promjene odabirom opcije "Spremi"
6. Sustav sprema promijenjene podatke u bazu podataka te prikazuje obavijest korisniku o uspješnoj promjeni podataka

UC6.2 -Uklanjanje korisničkog računa

- **Glavni sudionik:** Planinar
- **Cilj:** Izbrisati vlastiti korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moj profil"
 2. Prikazuju se korisnički podaci planinara
 3. Korisnik odabire opciju "Ukloni račun"
 4. Sustav uklanja korisnički račun iz baze podataka
 5. Sustav preusmjerava korisnika na naslovnu stranicu

UC7 -Pretraživanje korisnika po imenu i prezimenu

- **Glavni sudionik:** Planinar
- **Cilj:** Provjeriti posjeduje li određeni planinar korisnički račun, pronaći poznanike planinare
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik se nalazi na zidu vlastite planinarske zajednice i odabire polje za pretragu drugih korisnika
 2. Sustav omogućava unos podataka za pretragu
 3. Korisnik upisuje ime i/ili prezime željenog planinara
 4. Sustav dohvaća podatke iz baze podataka koji odgovaraju postavljenom upitu
 5. Sustav prikazuje dohvaćene podatke korisniku

UC8 -Stvoriti planinarsku stazu

- **Glavni sudionik:** Planinar
- **Cilj:** Stvoriti novu stazu prema unaprijed definiranom predlošku i dodati je na popis postojećih planinarskih staza
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinar odabire opciju "Stvori novu stazu"

2. Sustav prikazuje stranicu s odgovarajućim predloškom za unos staze
 3. Planinar ispunjava formu za dodavanje planinarske staze
 - (a) Unosi naziv staze
 - (b) Unosi početnu i završnu točku te duljinu staze
 - (c) Unosi razliku nadmorskih visina početne i završne točke
 - (d) Unosi zemljopisno područje staze (npr. planina na kojoj se staza nalazi)
 - (e) Podrazumijevani scenarij je da planinar stazu želi objaviti kao javnu
 4. Sustav dodaje stazu na popis postojećih planinarskih staza
- **Opis mogućih odstupanja:**
 - 3.a Planinar unese neispravne/nepotpune podatke
 1. Dobiva obavijest o neispravnim poljima
 2. Planinar ispravi neispravna polja i završi unos ili odustane od dodavanja nove planinarske staze
 - 4.a Postoji staza s istim imenom
 1. Planinar dobiva obavijest da je staza već postoji i ostaje na istoj formi za unos staze s mogućnošću izmjene unesenih podataka
 - 5.a Planinar označava stazu kao privatnu
 1. Staza ostaje vidljiva samo planinaru koji ju je stvorio

UC9 -Obrisati vlastitu planinarsku stazu

- **Glavni sudionik:** Planinar
- **Cilj:** Obrisati vlastite staze
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinar odabire na profilu opciju "Moje staze"
 2. Sustav prikazuje popis staza koje je stvorio trenutni korisnik
 3. Korisnik pronađe stazu koju želi ukloniti i odabire opciju "Ukloni stazu"
 4. Sustav postavlja poruku "Jeste li sigurni da želite obrisati stazu?"
 5. Korisnik potvrđuje i sustav obavještava korisnika da je staza uspješno obrisana
- **Opis mogućih odstupanja:**
 - 3.a Korisnik se predomišlja i ne želi obrisati stazu
 1. Sustav vraća korisnika na popis vlastitih staza
 - 4.a Staza koju korisnik pokušava obrisati je javna

1. Sustav obavještava korisnika da se javne staze ne mogu obrisati

UC10 -Slanje zahtjeva za prijateljstvom

- **Glavni sudionik:** Planinar
- **Cilj:** Poslati zahtjev za pridruživanjem vlastitoj planinarskoj zajednici drugom planinaru (zahtjev za prijateljstvom)
- **Sudionici:** Baza podataka, Planinar
- **Preduvjet:** Pošiljatelj i primatelj zahtjeva (planinari) posjeduju korisnički račun
- **Opis osnovnog tijeka:**
 1. Korisnik pronađe želenog planinara prema obrascu UC7
 2. Nakon prikaza korisnika koji zadovoljavaju pretragu planinar odabire opciju "Zahtjev za prijateljstvom"
 3. Sustav prikazuje poruku pošiljatelju da je zahtjev uspješno poslan te obavještava primatelja da ima novi zahtjev

UC11 -Pregledavanje pristiglih zahtjeva za prijateljstvom

- **Glavni sudionik:** Planinar
- **Cilj:** Planinar može vidjeti sve trenutno aktivne zahtjeve za prijateljstvom
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinaru stiže obavijest da je primio novi zahtjev za prijateljstvom ili želi pregledati pristigle zahtjeve za prijateljstvom
 2. Planinar odabire opciju pregleda pristiglih zahtjeva za prijateljstvom koja se nalazi na zaglavljku stranice
 3. Otvara se popis pristiglih zahtjeva za prijateljstvom čiji je status još uvek aktivan

UC11.1 -Prihvati zahtjev za prijateljstvom

- **Glavni sudionik:** Planinar
- **Cilj:** Prihvati zahtjev za pridruživanjem planinarskoj zajednici drugog planinara
- **Sudionici:** Baza podataka, Planinar
- **Preduvjet:** Oba korisnika imaju korisnički račun, jedan korisnik je drugom poslao zahtjev za pridruživanjem vlastitoj planinarskoj zajednici
- **Opis osnovnog tijeka:**

1. Planinaru stiže obavijest da je primio novi zahtjev za prijateljstvom koja se pokazuje na zaglavlju stranice pod opcijom "Pristigli zahtjevi"
 2. Planinar potvrđuje zahtjev za prijateljstvom te se on uklanja s popisa
 3. Na popis planinarske zajednice obojice planinara dodan je novi član ("prijatelj")
- **Opis mogućih odstupanja:**
 - 2.a Planinar može odbiti zahtjev za prijateljstvom
 1. Zahtjev za prijateljstvom se uklanja s popisa pristiglih zahtjeva
 2. Odbijeni planinar ima mogućnost ponovnog slanja zahtjeva za prijateljstvo planinaru koji ga je odbio

UC12 -Pregled prihvaćenih zahtjeva

- **Glavni sudionik:** Planinar
- **Cilj:** Planinar može vidjeti sve zahtjeve za prijateljstvom koje su mu prihvatiли drugi planinari
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinaru stiže obavijest da je netko prihvatio njegov zahtjev za prijateljstvom ili planinar želi vidjeti sve prihvaćene zahtjeve za prijateljstvom
 2. Planinar odabire opciju pregleda pristiglih zahtjeva za prijateljstvom koja se nalazi na zaglavlju stranice
 3. Trenutnom planinaru otvara se popis zahtjeva koje su prihvatili ostali planinari

UC13 -Pregled vlastitih planinarskih staza

- **Glavni sudionik:** Planinar
- **Cilj:** Omogućiti pregled svih staza koje je planinar stvorio
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinar odlazi na svoj profil odabirom opcije "Moj profil"
 2. Planinar na svom profilu odabire opciju "Moje staze"
 3. Sustav dohvaća sve staze iz baze podataka koje je stvorio planinar te ih prikazuje u odgovarajućem obliku
- **Opis mogućih odstupanja:**

3.a Planinar nema nijednu stvorenu stazu

1. Prikazuje se odgovarajuća poruka i nudi se opcija za stvaranje vlastite staze

UC14 -Stvoriti događaj

- **Glavni sudionik:** Planinar
- **Cilj:** Stvoriti novi događaj koji će biti vidljiv svim korisnicima s popisa planinarske zajednice kreatora događaja
- **Sudionici:** Baza podataka, članovi planinarske zajednice kreatora događaja (pozvani planinari)
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinar odabire opciju "Organiziraj događaj" na zidu njegove planinarske zajednice
 2. Sustav preusmjerava korisnika na stranicu s unaprijed definiranim predloškom za stvaranje događaja
 3. Planinar ispunjava predložak za stvaranje događaja
 4. Sustav stvara novi događaj i sprema ga u bazu podataka
 5. Stvoreni događaj vidljiv je svim članovima planinarske zajednice organizatora
- **Opis mogućih odstupanja:**
 - 3.a Planinar unosi nepotpune podatke
 1. Na predlošku se ispisuju poruke pogreške
 2. Planinar ispravi neispravna polja i završi unos ili odustane od stvaranja novog događaja

UC15 -Sudjelovati u događajima planinarske zajednice

- **Glavni sudionik:** Planinar
- **Cilj:** Događaje stvorene u vlastitoj planinarskoj zajednici označiti s "Dolazim"
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav s ulogom Planinar
- **Opis osnovnog tijeka:**
 1. Planinar odlazi na zid objava vlastite planinarske zajednice
 2. Planinar pregledava stvorene događaje te određeni događaj označi sa "Dolazim"
 3. Sustav pohranjuje njegovu odluku u bazu podataka
 4. Planinar je dodan na popis ljudi koji dolaze na odabrani događaj

- **Opis mogućih odstupanja:**

- 3.a Planinar je shvatio da ipak ne može doći na odabrani događaj
 1. Na konkretnom događaju odabire opciju "Otkaži dolazak"
 2. Sustav uklanja planinara s popisa ljudi koji dolaze na događaj

UC16 -Dodavanje posjećenih domova u arhivu

- **Glavni sudionik:** Planinar
- **Cilj:** Omogućiti planinaru arhiviranje posjećenih domova
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav kao Planinar
- **Opis osnovnog tijeka:**
 1. Korisnik pretražuje posjećeni dom i odabire opciju dodavanja u arhivu
 2. Odabrani dom se spremi u bazu podataka kao arhivirani dom tog planinara

UC17 -Zaslužiti priznanje (bedž)

- **Glavni sudionik:** Planinar
- **Cilj:** S obzirom na aktivnost planinara dodijeljuju mu se priznanja za ostvarena postignuća u obliku bedževa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav s ulogom Planinar te je ostvario aktivnosti potrebne za dobivanje bedža (npr. 10 posjećenih planinarskih domova, prepješaćenih 30km i sl.)
- **Opis osnovnog tijeka:**
 1. Planinar posjećuje domove i staze te ih dodaje u arhivu posjećenih domova/staza
 2. Sustav prati njegovu aktivnost te u slučaju zadovoljavanja određenih kriterija dodijeljuje mu bedž
 3. Na "Zidu planinarske zajednice" svim članovima planinarske zajednice planinara koji je dobio bedž prikazuje se obavijest o zaprimanju bedža
 4. Bedž postaje vidljiv na korisničkom profilu planinara

UC18 -Pregledati poruke koje su poslali korisnici

- **Glavni sudionik:** Administrator
- **Cilj:** Administrator može pregledati poruke koje su korisnici poslali vezano uz neprecizne i netočne informacije za neki dom ili stazu, te poruke o otvaranju novog planinarskog doma

- **Sudionici:** Baza podataka, Planinari
- **Preduvjet:** Korisniku je u bazi podataka dodijeljena uloga Administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Poruke korisnika" na svom profilu
 2. Sustav prikazuje sve poruke koje su korisnici poslali

UC19 -Pregledavanje popisa planinara u vlastitoj planinarskoj zajednici

- **Glavni sudionik:** Planinar
- **Cilj:** Vidjeti tko je sve uključen u planinarsku zajednicu u kojoj se nalazi planinar
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i pripada određenoj planinarskoj zajednici
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na profilnu stranicu
 2. Pretražuje planinare u vlastitoj planinarskoj zajednici pomoću opcije "Moja planinarska zajednica"

UC20 -Pregledavanje popisa željenih planinarskih staza (favoriti)

- **Glavni sudionik:** Planinar
- **Cilj:** Vidjeti koje staze planinara najviše zanimaju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odlazi na profilnu stranicu
 2. Odabire opciju "Favoriti"
 3. Prikazuje se lista planinarskih staza koje je korisnik označio sa zvjezdicom, tj. koje je prethodno dodao u favorite

UC21 -Dodavanje planinarskih staza na popis željenih

- **Glavni sudionik:** Planinar
- **Cilj:** Spremiti izlete za koje je planinar zainteresiran na popis željenih
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Pronalazak željenih izleta

2. Označavanje izleta za koje je korisnik zainteresiran sa zvjezdicom
3. Aplikacija dodaje označeni izlet na popis željenih izleta

UC22 -Ocjenvivanje stvorenih planinarskih staza drugih planinara

- **Glavni sudionik:** Planinar
- **Cilj:** Ocijeniti planinarske staze koje su stvorili drugi planinari
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Pronalazak tražene planinarske staze
 2. Ocjenjivanje nađene staze
 3. Aplikacija sprema ocjenu u bazu podataka i ažurira se prosječna ocjena staze

UC23 -Prijaviti netočne i neprecizne informacije vezane uz planinarske staze

- **Glavni sudionik:** Planinar
- **Cilj:** Prijaviti administratoru pogrešne informacije vezane uz planinarske staze
- **Sudionici:** Baza podataka, Administrator
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Pronalazak određene planinarske staze koja sadrži grešku
 2. Korisnik odabire opciju "Prijavi grešku"
 3. Sustav otvara model u kojem korisnik može opisati grešku
 4. Sustav obavještava korisnika da je greška uspješno prijavljena administratoru

UC24 -Prijaviti netočne i neprecizne informacije vezane uz planinarske domove

- **Glavni sudionik:** Planinar
- **Cilj:** Prijaviti administratoru pogrešne informacije vezane uz planinarske domove
- **Sudionici:** Baza podataka, Administrator
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Pronalazak određenog planinarskog doma koji sadrži grešku
 2. Korisnik odabire opciju "Prijavi grešku"

3. Sustav otvara model u kojem korisnik može opisati grešku
4. Sustav obavještava korisnika da je greška uspješno prijavljena administratoru

UC25 - Pregledati naslovnicu vlastite planinarske zajednice

- **Glavni sudionik:** Planinar
- **Cilj:** Na naslovniči vidjeti događaje koje su kreirali planinari iz vlastite planinarske zajednice, kao i njihove planinarske uspjehe (dobivene bedževe) te imati mogućnost pretraživanja korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i pripada određenoj planinarskoj zajednici
- **Opis osnovnog tijeka:**
 1. Odlazak na stranicu "Moja planinarska zajednica"
 2. Korisniku je omogućen pregled liste budućih događaja koje su kreirali planinari iz njegove planinarske zajednice
 3. Korisnik vidi obavijesti o ostvarenim bedževima planinara iz svoje planinarske zajednice
 4. Korisnik ima mogućnost pretraživanja drugih planinara prema imenu i prezimenu
- **Opis mogućih odstupanja:**
 - 2.a Korisnik uočava netočne informacije vezane uz objavljeni događaj
 1. Korisniku je omogućeno da prijavi administratoru netočne informacije kako bi se što prije ispravile pogreške

UC26 - Arhivirati odradene planinarske staze

- **Glavni sudionik:** Planinar
- **Cilj:** Na jednom mjestu (arhiva) imati sve planinarske staze koje je planinar odradio
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Pronalazak određene planinarske staze
 2. Planinar odabranu stazu označava kao posjećenu
 3. Staza se dodaje u arhivu gdje se nalaze i sve ostale odradene staze
- **Opis mogućih odstupanja:**
 - 1.a U bazu podataka nije unesena tražena staza

1. Planinar unosi u bazu novu stazu
2. Nova staza se sprema u bazu podataka
3. Planinar označava novostvorenu stazu kao posjećenu

UC27 -Kontaktirati administratora

- **Glavni sudionik:** Planinar
- **Cilj:** Otvaranje novog doma ukoliko planinar posjeduje nekretninu koja bi se mogla preuređiti u dom ili promjena infrastrukture već postojećeg doma
- **Sudionici:** Baza podataka, Administrator
- **Preduvjet:** Korisnik prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Planinar odabire opciju "Kontaktiraj administratora"
 2. Unosi podatke o novom domu ili navodi promjene infrastrukture koje želi uraditi kod već postojećeg doma
 3. Šalje navedene podatke administratoru koji onda provjerava dobivene informacije

UC28 -Stvoriti novi planinarski dom

- **Glavni sudionik:** Administrator
- **Cilj:** Dodati novi planinarski dom na popis svih planinarskih domova
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisniku je u bazi podataka dodijeljena uloga Administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju stvaranja novog planinarskog doma
 2. Prikazuje se predložak za unos podataka o planinarskom domu
 3. Administrator popunjava predložak i sprema novostvoren planinarski dom
 4. Sustav stvara novi dom u bazi podataka te o tom obavještava administratora
 5. Novostvoren dom nalazi se na popisu svih domova

UC29 -Administrator briše korisnički račun

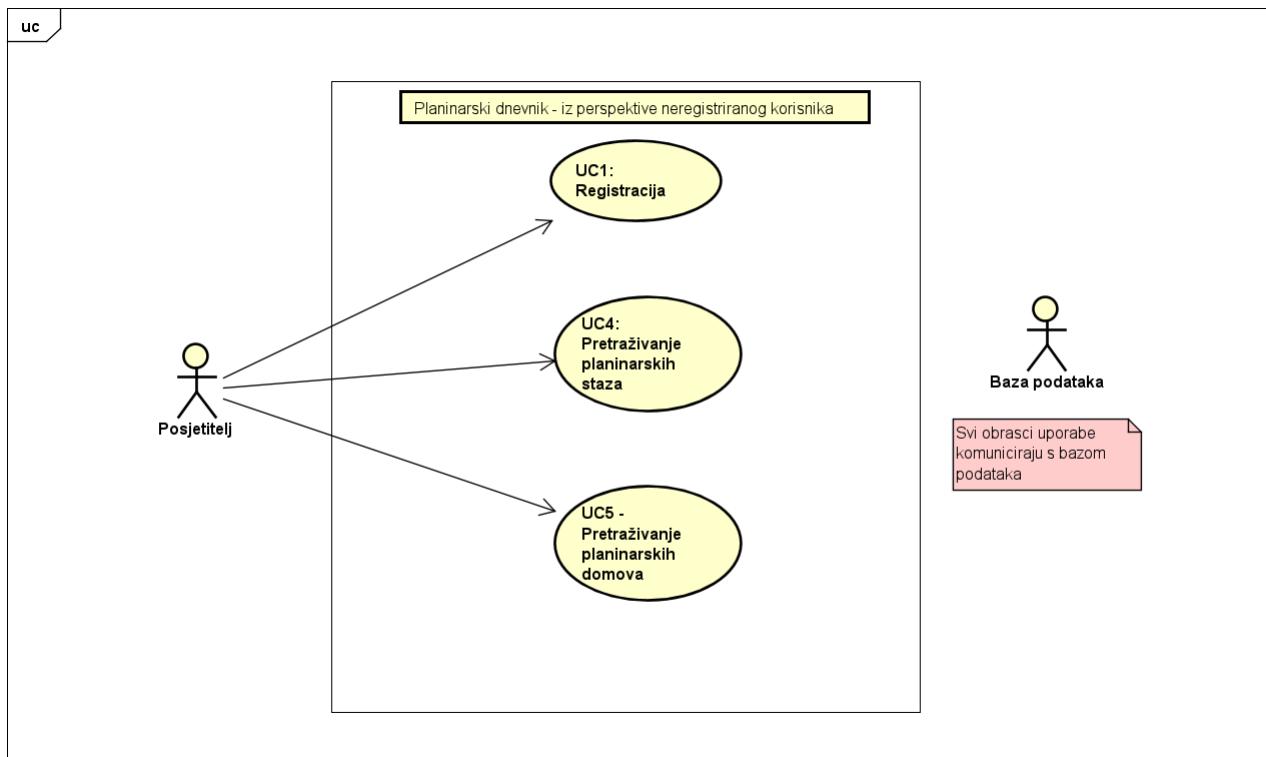
- **Glavni sudionik:** Administrator
- **Cilj:** Administrator može izbrisati korisnički račun određenog planinara
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisniku je u bazi podataka dodijeljena uloga Administratora
- **Opis osnovnog tijeka:**

1. Administrator pronađe željenog planinara i na njegovom profilu odabire opciju ukloni račun
2. Podaci tog korisnika se uklanjuju iz baze podataka kao i sam korisnički račun

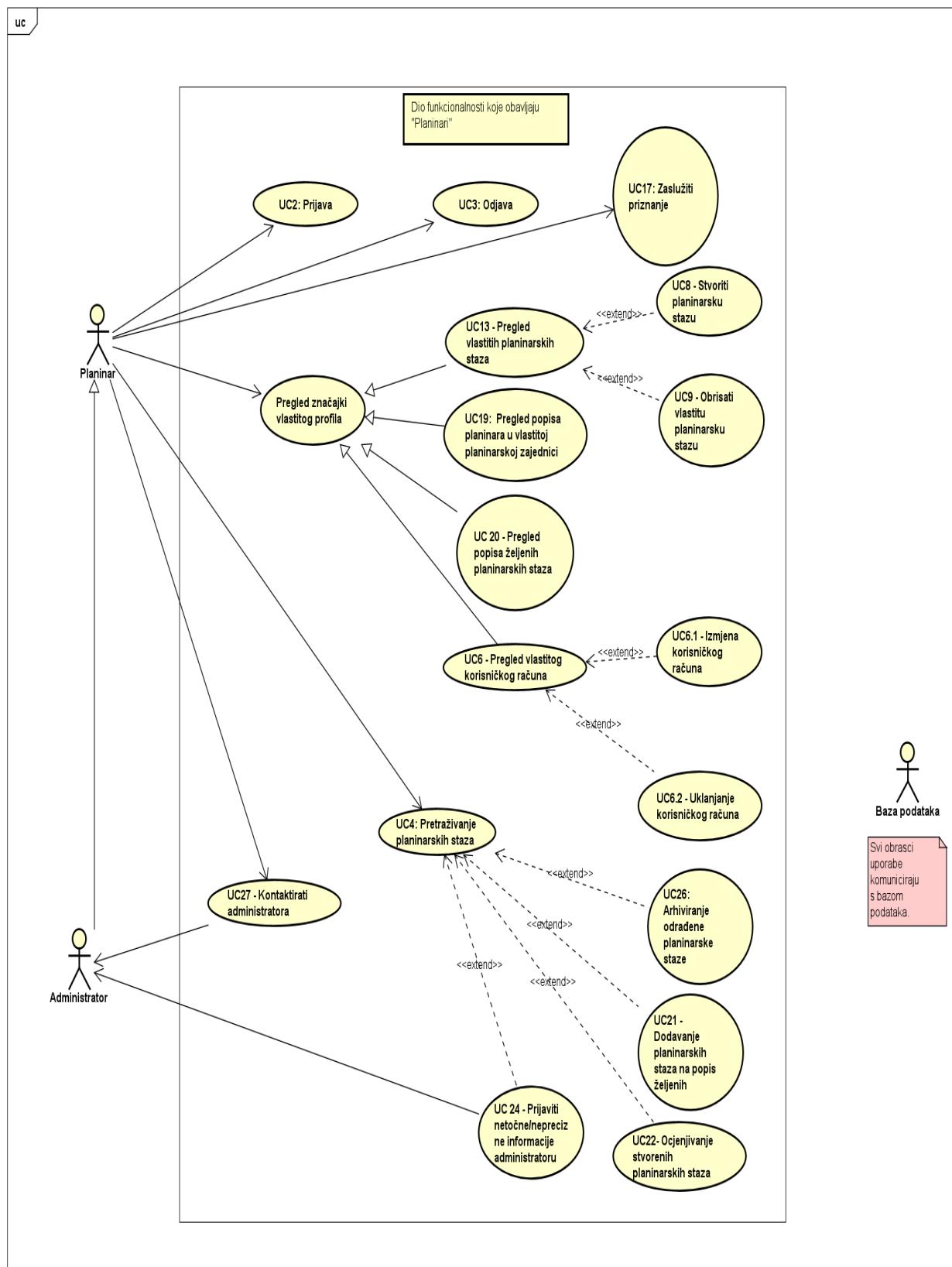
UC30 -Obrisati postojeći planinarski dom

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati postojeće planinarske domove
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisniku je u bazi podataka dodijeljena uloga Administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregleda svih planinarskih domova
 2. Administrator odabire opciju "Uklonite planinarski dom"
 3. Sustav pita korisnika je li siguran da želi obrisati planinarski dom
 4. Administrator je siguran da želi obrisati planinarski dom
 5. Sustav uklanja planinarski dom iz baze podataka
- **Opis mogućih odstupanja:**
 - 1.a Administrator odustaje od brisanja planinarskog doma
 1. Sustav vraća administratora na popis svih planinarskih domova

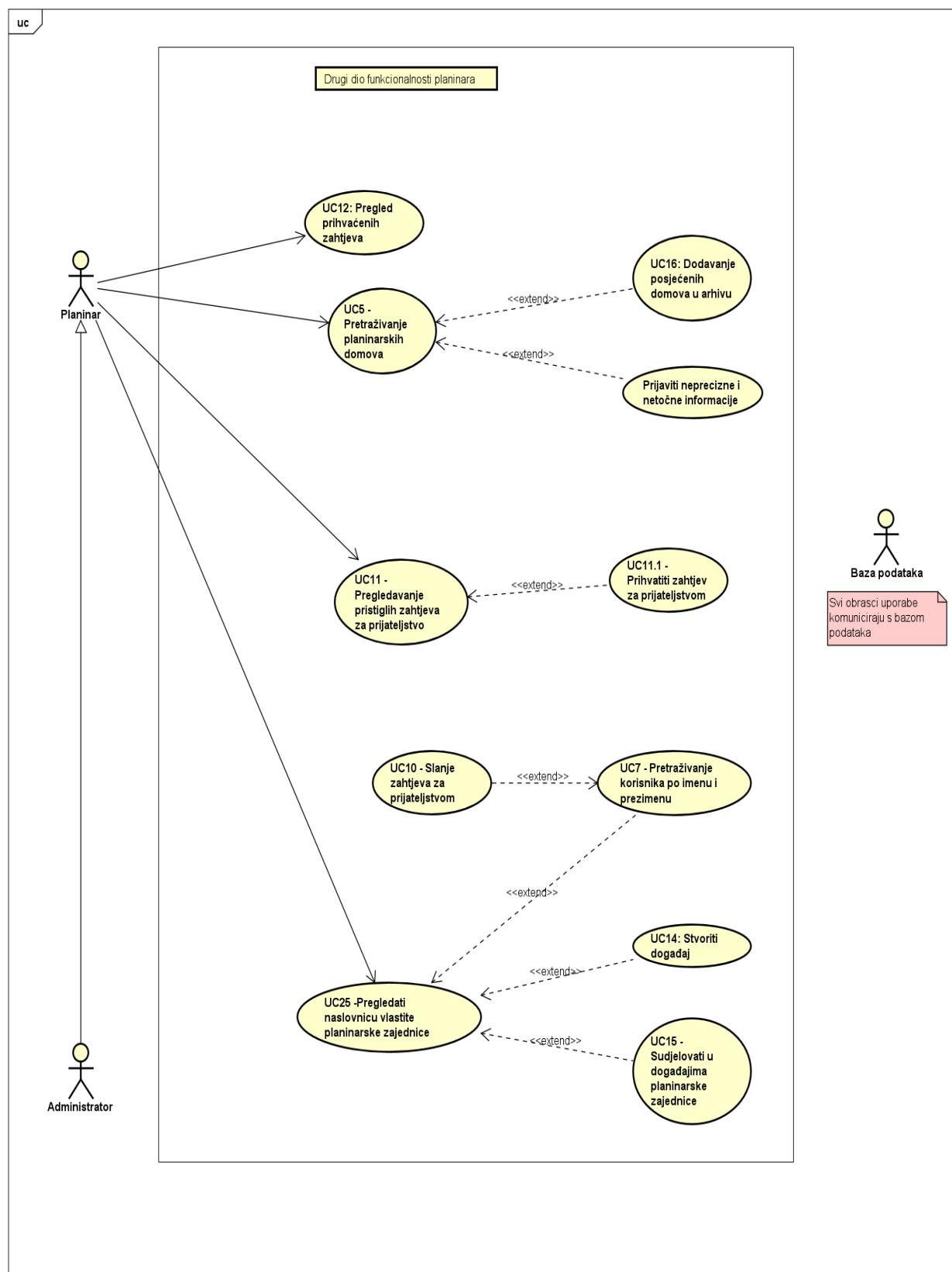
Dijagrami obrazaca uporabe



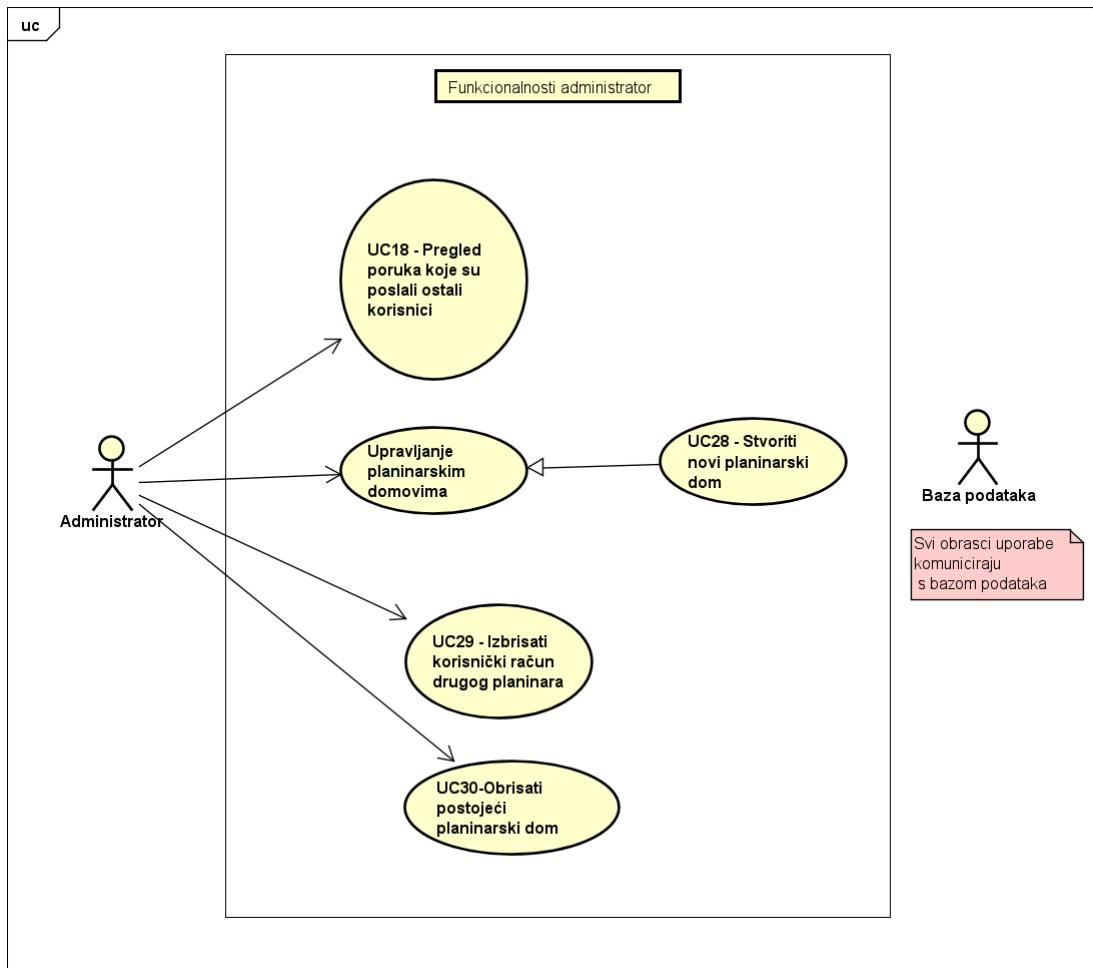
Slika 3.1: Prikaz funkcionalnosti dostupnih neregistriranom ili neprijavljenom korisniku



Slika 3.2: Dio funkcionalnosti koje obavljaju planinari



Slika 3.3: Drugi dio funkcionalnosti planinara i zasebne aktivnosti dežurnog planinara

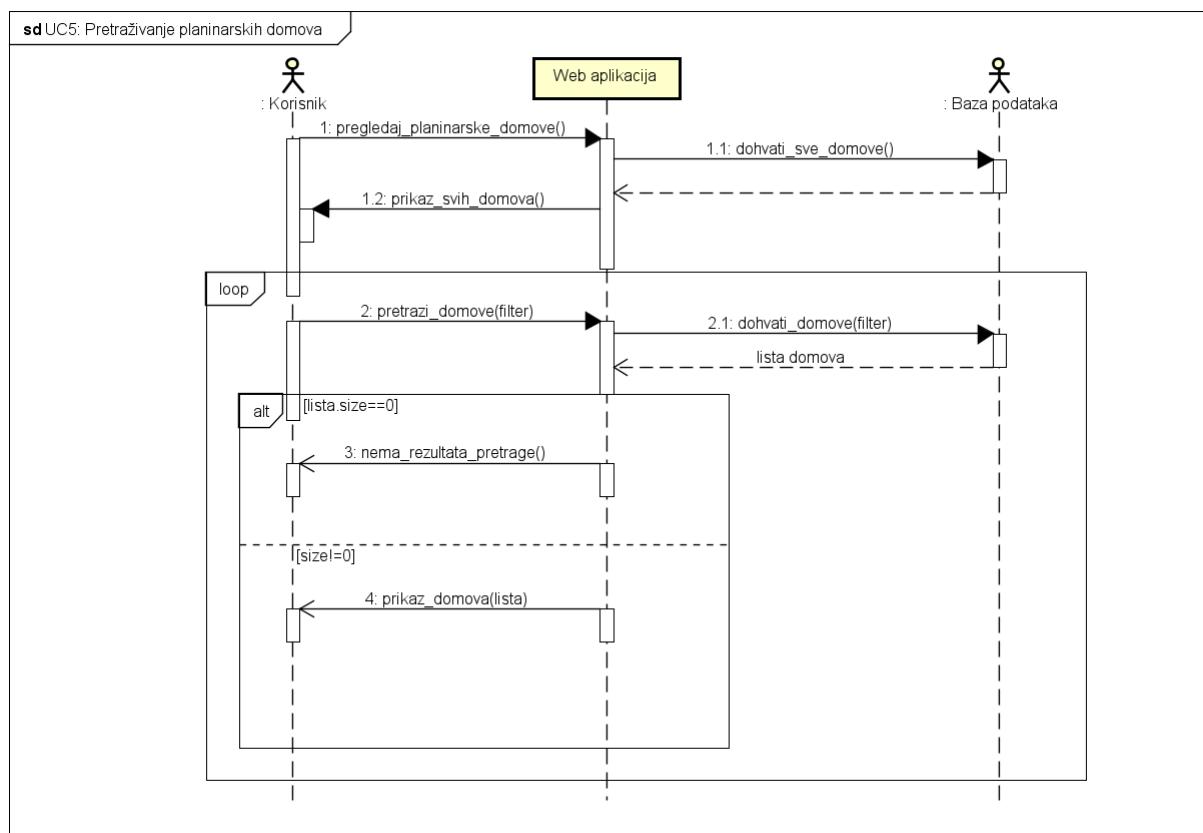


Slika 3.4: Prikaz funkcionalnosti koje obavlja administrator

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC5 - Pretraživanje planinarskih domova

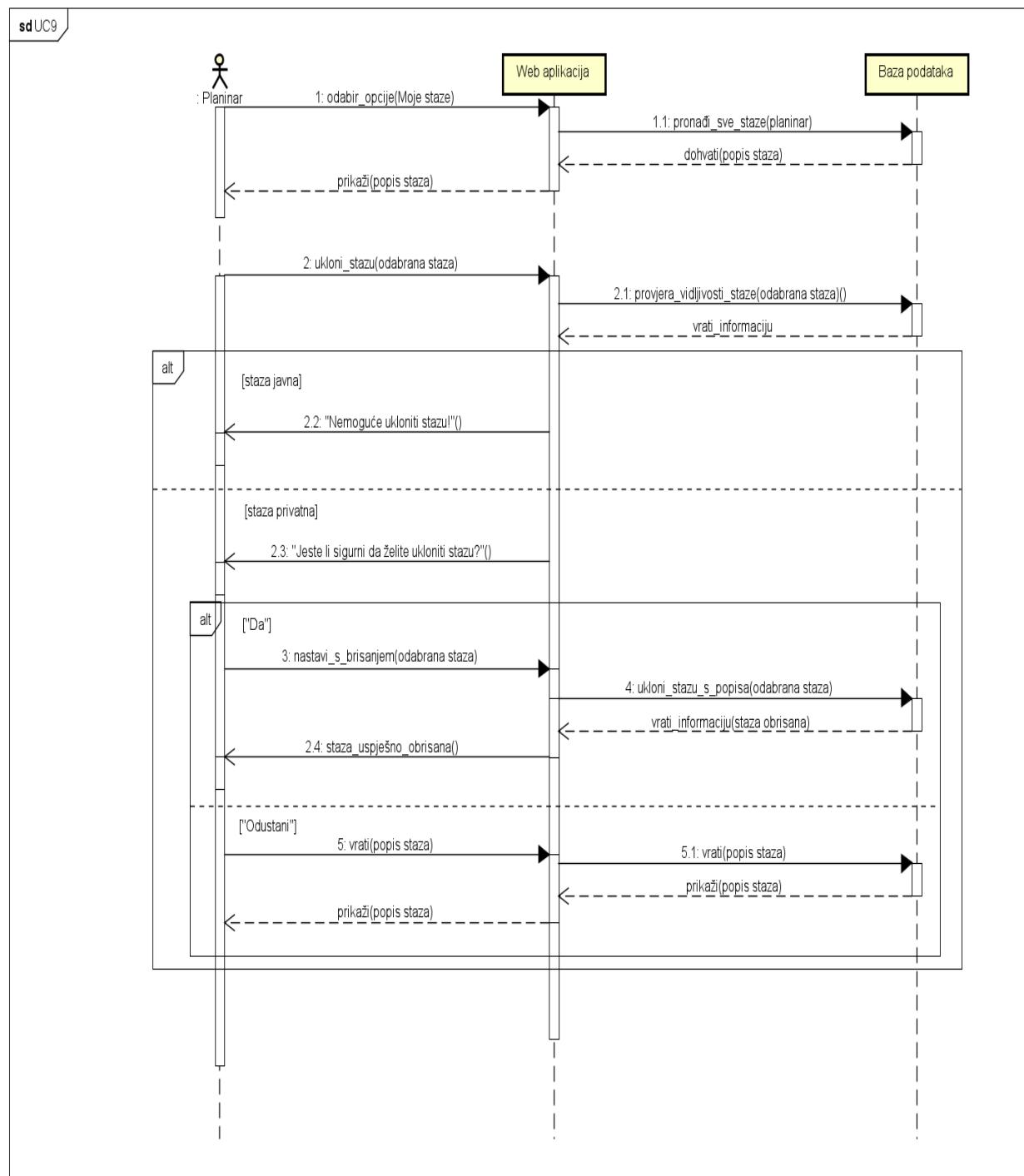
Korisnik odabire funkciju pregleda svih planinarskih domova. Web aplikacija, na traženi zahtjev, dohvaća planinarske domove iz baze podataka. Baza podataka zatim šalje odgovor Web aplikaciji. Odgovor baze podataka Web aplikacija prikazuje korisniku. Nadalje, ako korisnik želi užu pretragu planinarskih domova, odabire jedan od filtera koje aplikacija nudi. Web aplikacija dohvaća iz baze podataka planinarske domove koji zadovoljavaju filtere korisnika. Baza podataka šalje listu odgovarajućih domova aplikaciji. U slučaju da je lista prazna, Web aplikacija korisniku šalje povratnu informaciju o nepostojećem domu. Ako su odgovarajući domovi pronađeni u bazi podataka Web aplikacija prikazuje listu planinarskih domova korisniku.



Slika 3.5: Ponašajni prikaz pretraživanja planinarskih domova

Obrazac uporabe UC9 - Obrisati vlastitu planinarsku stazu

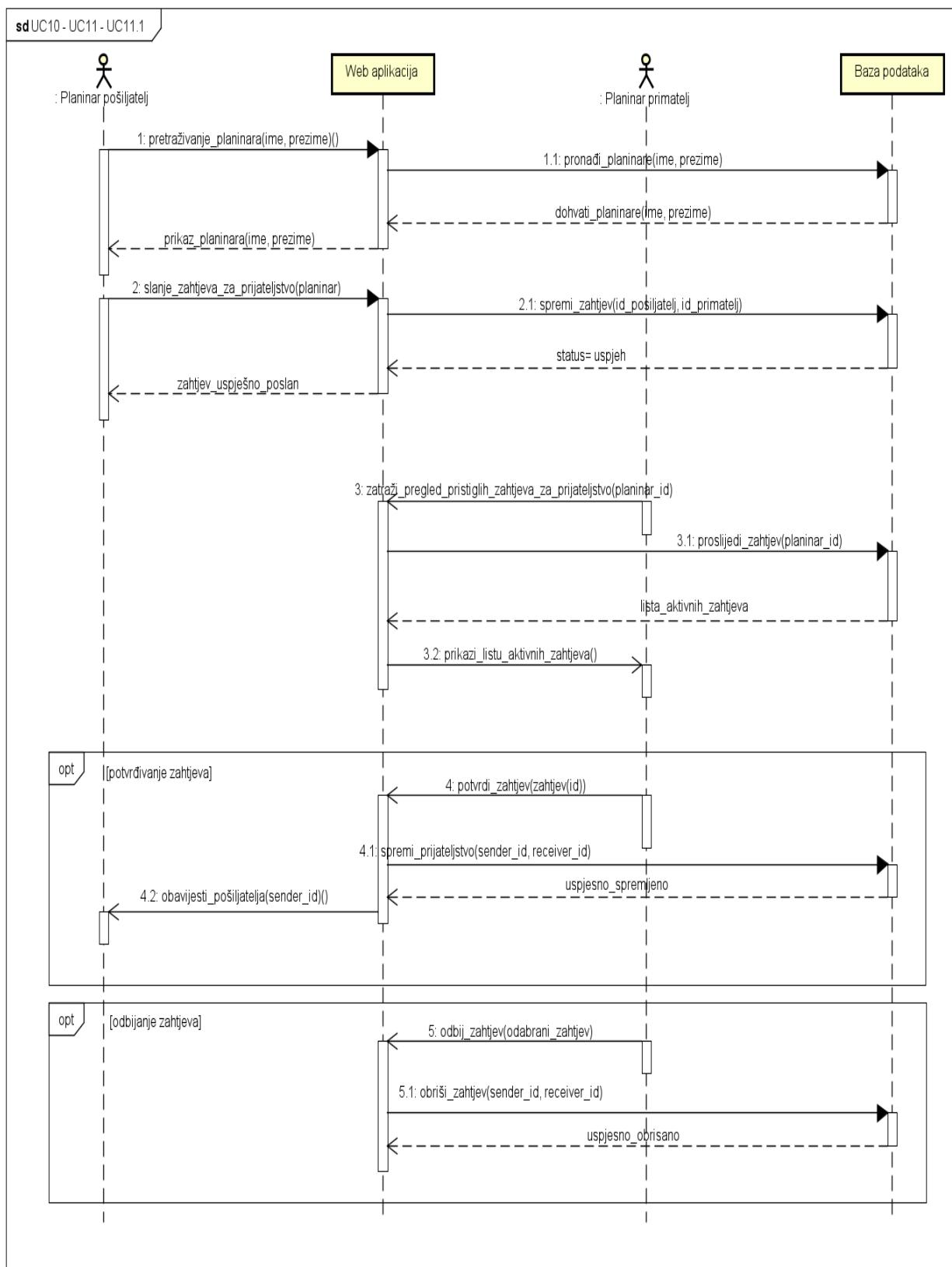
Prijavljeni korisnik (planinar) na vlastitom profilu ima mogućnost pregleda popisa svih staza koje je stvorio. Odabirom opcije "Moje staze" planinar šalje zahtjev poslužitelju za prikaz tih staza. Poslužitelj dohvaća sve njegove staze iz baze podataka i prikazuje ih planinaru na njegovom profilu. Planinar zatim može odabratи stazu koju želi ukloniti tako što odabere opciju "Ukloni stazu". Poslužitelj u bazi podataka provjerava vidljivost odabrane staze, odnosno je li ona javna ili privatna. Ako u bazi podataka za odabranu stazu vrijedi da je javna, onda će korisnik od poslužitelja primiti poruku da je stazu nemoguće ukloniti. Inače, ako je staza privatna, poslužitelj će korisniku postaviti pitanje: "Jeste li sigurni da želite ukloniti stazu?" na što planinar može odgovoriti potvrđno ili odustati od brisanja. U slučaju potvrđnog odgovora, kojeg planinar šalje poslužitelju, poslužitelj će narediti bazi podataka da ukloni odabranu stazu s popisa staza i vratit će se odgovor prema planinaru da je staza uspješno uklonjena. Ako planinar odustane od brisanja, sustav će ga vratiti na popis vlastitih staza.



Slika 3.6: Ponašajni prikaz brisanja vlastitih planinarskih staza

Obrasci uporabe UC10-UC11-UC11.1 - Zahtjevi za prijateljstvom

Prijavljeni korisnik (planinar) može drugom korisniku poslati zahtjev za pridruživanjem vlastitoj planinarskoj zajednici, odnosno zahtjev za prijateljstvom. Uvjet je da pošiljatelj i primatelj (planinari) imaju korisnički račun, drugim riječima da su se uspješno registrirali u aplikaciju. Planinar ima mogućnost pretraživanja drugih planinara po imenu i prezimenu. Poslužitelj će tu pretragu proslijediti bazi podataka koja će pronaći sve korisnike koji zadovoljavaju pretragu. Baza podataka vraća odgovarajuće korisnike poslužitelju koji ih onda prikazuje planinaru. Zatim planinar pošiljatelj odabire planinara iz prikazanih mu korisnika i odluči mu poslati zahtjev za prijateljstvom. Bilo koji planinar prijavljen u aplikaciju može vidjeti sve trenutno aktivne zahtjeve za prijateljstvom. U našem slučaju planinar primatelj može ili dobiti obavijest da je primio zahtjev ili može zatražiti pregled pristiglih zahtjeva. Kada planinar prima novi zahtjev poslužitelj prolongira zahtjev do planinara primatelja i vraća se povratna informacija do planinara pošiljatelja da je zahtjev uspješno poslan. Za to vrijeme u bazu podataka će se dodati taj zahtjev među aktivne zahtjeve za prijateljstvom. Ako planinar sam zatraži pregled svih zahtjeva to će napraviti tako da kontaktira poslužitelja koji će naredbu proslijediti bazi podataka. Baza podataka će tada dohvati aktivne zahtjeve za prijateljstvom i poslati ih poslužitelju koji će ih konačno prikazati planinaru. Planinar primatelj kod obrađivanja primljenog zahtjeva za prijateljstvo ima dvije mogućnosti: potvrditi ili odbiti zahtjev. U slučaju povrđivanja zahtjeva planinar primatelj će poslati potvrdu poslužitelju koji će javiti planinaru primatelju da je prihvaćen njegov zahtjev za prijateljstvo. Istovremeno baza podataka će ukloniti zahtjev s popisa aktivnih zahtjeva. Ukoliko planinar primatelj odluči odbiti zahtjev, onda se taj zahtjev također uklanja s popisa aktivnih zahtjeva u bazi podataka i planinar pošiljatelj može ponovno poslati zahtjev tom istom planinaru koji ga je već odbio.



Slika 3.7: Ponašajni prikaz UC10 - UC11 - UC11.1

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Sustav treba biti implementiran kao Web aplikacija koja će biti prilagođena i prikazu na mobilnim uređajima (aplikacija mora imati responzivan dizajn)
- Korisnicima korištenje aplikacije treba biti intuitivno jasno, bez potrebe za dodatnim korisničkim uputama
- Korisničko sučelje i sustav trebaju koristiti hrvatski standardni jezik (uključujući dijakritičke znakove)
- Eventualne pogreške korisnika i/ili administratora ne smiju utjecati na uspješno funkcioniranje aplikacije
- Baza podataka treba biti brza, učinkovita i dobro povezana sa sustavom, otporna na bilo kakve greške korisnika i administratora
- Aplikacija mora biti dostupna svim zainteresiranim korisnicima, odnosno svim već aktivnim planinarima, ali i onima koji to tek namjeravaju postati
- Korisnik sustavu treba moći pristupiti iz javne mreže pomoću protokola HTTPS
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava

4. Arhitektura i dizajn sustava

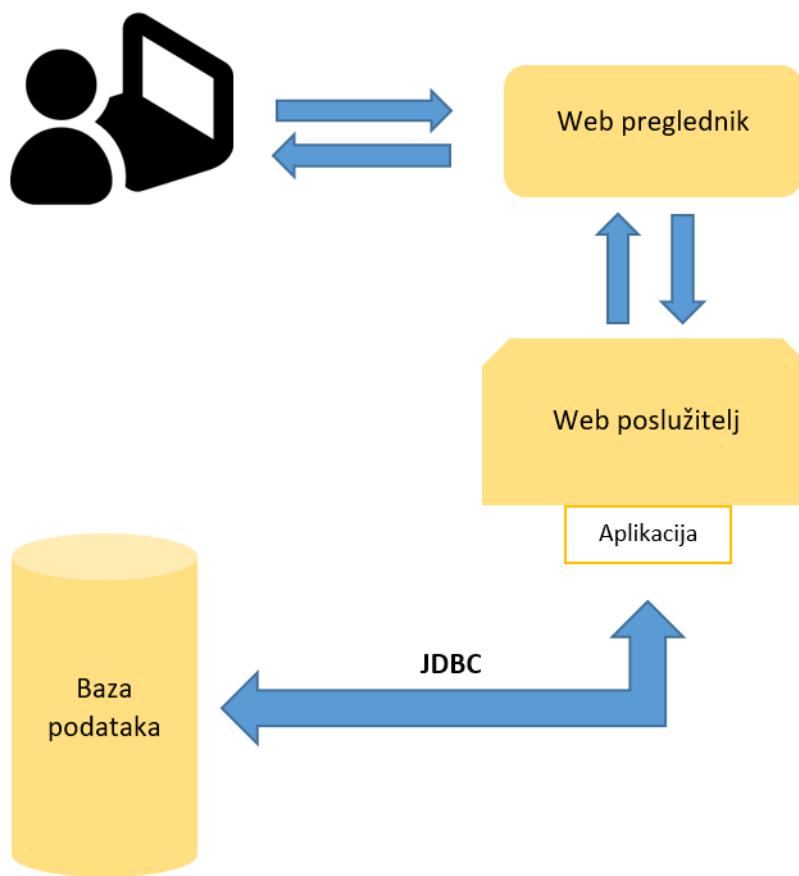
Arhitektura programske potpore predstavlja strukturu sustava ili više njih koje sadrži elemente, njihova obilježja i odnose među njima. Temeljni razlozi definiranja arhitekture:

- poboljšava razumljivost i komunikaciju sudionika
- pomaže u donošenju temeljnih odluka pri izradi projekta
- omogućava rano uočavanje pogrešaka u oblikovanju
- moguće ponovno korištenje rješenja (engl. reuse)

U konačnici, efikasno strukturiranje arhitekture programske potpore dovest će do poboljšanja kvalitete finalnog produkta projekta.

Koristimo objektno usmjerenu arhitekturu koja najbolje odgovara razvoju složene Web aplikacije namijenjene za što više korisnika u stvarnom vremenu. Možemo ju klasificirati na četiri ključna dijela koji osiguravaju izvršavanje naredbi korisnika:

1. Web preglednik
2. Web poslužitelj
3. Web aplikacija
4. Baza podataka



Slika 4.1: Arhitektura sustava

Web aplikacija će se temeljiti na modelu klijent-poslužitelj, što je danas i najčešće korišteni model. Korisnik šalje zahtjeve na koje odgovara poslužitelj, dok i jedna i druga strana mogu imati korisničku i poslužiteljsku aplikaciju.

Web preglednik:

Klijentski program, zvan preglednik, služi kao korisničko sučelje za pregledavanje sadržaja na Webu. On je taj koji šalje zahtjev Web poslužitelju i prikazuje primljene podatke u obliku Web stranica korisniku. Dakle, preglednik će primljene podatke u obliku koda interpretirati u nešto korisniku razumljivo, odnosno prikazat će korisničko sučelje naše aplikacije. Konačan prikaz aplikacije može uključivati više dohvata resursa i često može sadržavati dodatke te pomoćne aplikacije za prikaz formata koje izvorno ne podržava.

Web poslužitelj:

Poslužiteljski program poslužuje resurse smještene na poslužiteljskom računalu ili na drugim izvorima i odgovara na zahtjeve korisnika. Komunikacija se odvija preko HTTP/HTTPS (*HyperText Transfer Protocol/Secure*) standardnog internetskog aplikacijskog protokola koji ima mogućnost prijenosa raznih vrsta podataka i proširiv je prema novim formatima podataka. Poslužitelj je zaslužan za pokretanje Web aplikacije.

Web aplikacija:

Za realizaciju frontend-a, odnosno korisničkog sučelja upotrijebit ćemo React kao bazu unutar kojega ćemo koristiti jezike HTML, TypeScript i CSS. TypeScript nam omogućava izradu dinamičkih web stranica u kombinaciji s HTML-om i CSS-om i njime možemo mijenjati sadržaj na stranici ovisno o načinu interakcije korisnika sa stranicom. Uz ove navedene tehnologije moguće je napraviti moderno korisničko sučelje jedne Web i mobilne aplikacije. Nakon što Web preglednik korisniku prikaže aplikaciju "Planinarski dnevnik", korisnik može izvršiti određenu naredbu odabirom neke od funkcionalnosti aplikacije. Hoće li pristupiti bazi podataka ovisi o samoj akciji. Za komunikaciju s bazom podataka koristi se JDBC koji predstavlja sučelje aplikacijskog programiranja za jezik Java i definira kako klijent može pristupiti bazi podataka. Pruža metode za upit i ažuriranje u bazi podataka te je orijentiran prema relacijskim bazama podataka. Što se tiče backend-a koristimo Spring Boot i MVC arhitekturu.

Spring Boot pruža fleksibilan način konfiguriranja Java Beans, XML konfiguracija i transakcija baze podataka te bitno olakšava upravljanje ovisnostima. Svaka pokrenuta usluga ima svoj postupak, a time se postiže jednostavan model za podršku aplikacijama.

Model – View – Controller je obrazac koji razdvaja aplikaciju u tri glavne logičke komponente: Model, View i Controller. Svaka od nabrojenih komponenti ima zadatak rukovati s određenim razvojnim aspektima aplikacije. Također, one su nezavisne jedna od druge i kao rezultat toga je jednostavno dodavanje i preoblikovanje svojstava.

- **Model**

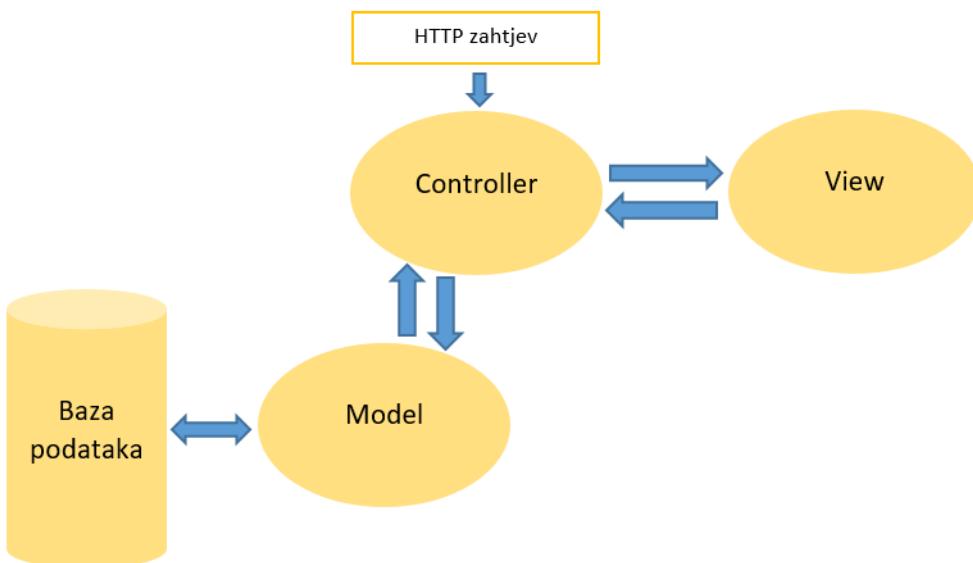
Poznat je kao najniža razina što znači da je odgovoran za održavanje podataka s kojima korisnik radi. Glavni zadatak je dohvati, manipulacija podatcima i uglavnom on surađuje s bazom podataka. Reagira na zahtjeve Controller-a jer on nikada sam ne razgovara s bazom podataka i nakon komunikacije prosljeđuje potrebne podatke Controller-u. Jedna od bitnijih stvari za napomenuti je da Model nikada izravno ne komunicira s View.

- **View**

Služi za prikazivanje podataka na način da zapravo generira korisničko sučelje za korisnika. Ti podaci su rezultat rada Model-a, ali se oni ne preuzimaju izravno već putem Controller-a tako da View surađuje samo s Controller-om.

- **Controller**

Djeluje u službi posrednika između komponenti Model i View. Ne mora brinuti o rukovanju logikom podataka, već samo govori Model-u što treba učiniti. Nakon primanja podataka od Model-a, on ih obraduje i konačno rezultat prosljeđuje do View-a gdje objašnjava kako ih prikazati korisniku. Ako je došlo do promjena, Controller je zadužen za ažuriranje View-a.



Slika 4.2: Model - View - Controller

4.1 Baza podataka

Za naš projekt odabrali smo **relacijsku bazu podataka** zbog njezine pogodnosti da prikaže mali dio stvarnog svijeta bez redundancije unutar same baze. Dohvaćanje podataka je relativno brzo i lagano se može paralelizirati. Specifičnu implementaciju relacijske baze podataka koju smo odabrali je **PostgreSQL**. To je baza podataka otvorenog koda s preko 30 godina aktivnog razvoja zbog kojeg je zaslužila svoju čvrstu reputaciju za pouzdanost, bogatstvo opcijama i visokim performansama. Zbog načina na koji je SQL standard napisan, vrlo je slična ostalim SQL bazama podataka.

Tijekom razvoja koristimo **H2 bazu**. To je privremena baza podataka koja služi za testiranje koda. Svi zapisi u njoj se nalaze u privremenoj memoriji i nisu perzistentni. Zbog toga je odlična za testiranje. Ona je otvorenog koda, napisana u Javi i ima čvrste sigurnosne postavke. Na nju se možemo povezati s više konekcija i baza podataka je enkriptirana SHA-256 enkripcijom. Ima vrlo malu potrošnju memorije i zauzima relativno malo prostora na disku (oko 2MB).

Također koristimo **JPA (Java Persistence API)** koje samo sadrži sučelja za stvaranje Persistence layouta. Dopušta nam da mapiramo entitete tablica i veze između tablica na objekte u Javi. Ovo sučelje definira svoj vlastiti jezik za upite (JPQL). JPQL prevoditelj interpretira kod i piše SQL upite. JPA ne možemo samostalno koristiti, već nam treba konkretna implementacija toga sučelja. Implementacija koju ćemo mi koristiti se zove Hibernate.

Na bazu podataka se iz Java spajamo preko **JdbcTemplatea**. To je snažni mehanizam za spajanje i izvođenje SQL upita. On nam smanjuje količinu koda koju moramo napisati kako bismo izvršavali upite, kao što je spajanje na bazu, kreiranje izraza i zatvaranje konekcija.

Naš model baze podataka sastoji se od sljedećih **glavnih** i veznih tablica:

- **user**
- **roles**
- **mountain_lodge**
- **mountain_path**
- **event**
- **hill**
- **utility**
- **badge**
- **message**
- friendships
- friendship_request
- friendship_notifications
- user_badge
- mountain_lodge_utility
- event_attendance
- mountain_path_report
- mountain_path_grade
- path_user_wishlist
- mountain_path_user_archived
- community_event_mountain_path
- mountain_lodge_user_archived

4.1.1 Opis tablica

Primarni ključevi su označeni **podebljano**, dok su strani ključevi podvučeni. Prikazane su i sve vezne tablice, tako da se nad glavnim tablicama Many-To-Many veze neće navoditi.

user - ovaj entitet sadrži sve važne informacije o korisniku aplikacije. Sadrži atributе: "id", "name", "email", "password", "image", "place_of_residence", "date_of_birth", "role_id", "current_user_id" i "description" koji redom predstavljaju jedinstveni identifikacijski broj korisnika, ime i prezime, e-mail korisnika, lozinku korisnika, sliku vidljivu na profilu korisnika, mjesto stanovanja korisnika te datum rođenja korisnika, id role korisnika, id trenutnog korisnika i opis. Neki podaci o korisniku, kao što su mjesto stanovanja i datum rođenja su optionalni, i ako ih korisnik ne ispuni njihova vrijednost je *NULL*. Atribut "role_id" je strani ključ koji referencira entitet **roles** te se radi o Many-To-One vezi.

user - (Korisnik)		
id	BIGINT	jedinstveni identifikacijski broj korisnika
name	VARCHAR	ime i prezime korisnika
email	VARCHAR	e-mail pomoću kojeg se korisnik prijavljuje u sustav
description	VARCHAR	opis korisnika, unos je optionalan
password	VARCHAR	lozinka pomoću koje se korisnik prijavljuje u sustav
place_of_residence	VARCHAR	mjesto stanovanja korisnika, unos je optionalan
image	BYTEA	slika vidljiva na profilu korisnika
current_user_id	BIGINT	id trenutnog korisnika
<u>role_id</u>	BIGINT	strani ključ role iz tablice roles
date_of_birth	DATE	datum rođenja korisnika, unos je optionalan

roles Ovaj entitet modelira ulogu korisnika unutar aplikacije, tzv. "aplikativne role". Određuje razinu ovlasti koje korisnik ima. Sadrži atributе: "id" i "name" koji predstavljaju jedinstveni identifikator uloge te naziv uloge. Dopuštene uloge u našoj aplikaciji su: Planinar, Dežurni planinar i Administrator.

roles - (Uloga)		
id	BIGINT	jedinstveni ID uloge
name	VARCHAR	ime uloge

mountain_lodge Ovaj entitet modelira jedan planinarski dom. Sadrži atribute: "id", "name", "image", "elevation" i "hill_id" koji predstavljaju jedinstveni identifikator planinarskog doma, naziv planinarskog doma, sliku planinarskog doma, nadmorsku visinu te jedinstveni identifikator zemljopisnog područja (visočja) na kojemu se planinarski dom nalazi. Unos slike za neki planinarski dom je opcionalan, a ako se atribut ne popuni njegova vrijednost je *NULL*. Atribut "hill_id" predstavlja strani ključ koji referencira entitet **hill** te se radi o Many-To-One vezi.

mountain_lodge - (Planinarski dom)		
id	BIGINT	jedinstveni identifikacijski broj planinarskog doma
name	VARCHAR	naziv planinarskog doma
image	BYTEA	slika planinarskog doma koja se prikazuje unutar aplikacije, unos je opcionalan
elevation	INTEGER	nadmorska visina na kojoj se planinarski dom <u>nalazi izražena u kilometrima</u>
<u>hill_id</u>	BIGINT	strani ključ na tablicu hill , a predstavlja identifikator visočja na kojemu se nalazi planinarski dom

utility Ovaj entitet modelira značajke infrastrukture pojedinog doma. Sadrži atribute "id" i "name" koji predstavljaju jedinstveni identifikacijski broj značajke te naziv značajke. Primjeri takvih značajki su pitka voda, hrana, smještaj, internet.

utility - (Pogodnost, značajka)		
id	BIGINT	jedinstveni identifikator značajke
name	VARCHAR	naziv značajke/pogodnosti

hill Ovaj entitet modelira pojedino zemljopisno područje, odnosno visočje na kojemu se nalazi pojedini planinarski dom ili planinarska staza. Sadrži atribute: "id" i "name" koji predstavljaju jedinstveni identifikacijski broj visočja te naziv visočja.

hill - (Visočje)		
id	BIGINT	jedinstveni ID visočja
name	VARCHAR	ime visočja

mountain_path Ovaj entitet sadrži informacije o pojedinoj planinarskoj stazi. Sadrži atribute: "id", "name", "start_point", "end_point", "avg_walk_time", "length", "sea_level_diff", "date_created", "is_private", "author_id" i "hill_id" koji redom predstavljaju jedinstveni identifikacijski broj planinarske staze, naziv staze, polazišnu točku, završnu točku, prosječno vrijeme potrebno da se prepješači stazu, duljinu staze, razliku u nadmorskoj visini između početne i završne točke, datum kreiranja pojedine staze od strane korisnika unutar naše aplikacije, vrijednost koja govori je li staza privatna ili javna, korisnika koji je stvorio stazu te identifikator visočja na kojemu se staza nalazi. Atribut "author_id" te atribut "hill_id" su strani ključevi obzirom na tablicu **user** odnosno **hill** te se radi o Many-To-One vezama.

mountain_path - (Planinarska staza)		
id	BIGINT	jedinstveni identifikacijski broj planinarske staze
name	VARCHAR	naziv staze
start_point	VARCHAR	naziv početne točke staze
end_point	VARCHAR	naziv završne točke staze
avg_walk_time	TIME	prosječno vrijeme potrebno za prepješaćiti stazu
length	INTEGER	duljina staze u metrima

mountain_path - (Planinarska staza)		
sea_level_diff	INTEGER	razlika u nadmorskoj visini između početne i završne točke staze
date_created	DATE	datum stvaranja staze unutar aplikacije
is_private	BOOLEAN	atribut koji govori je li korisnik stazu stvara privatno ili javno
author_id	BIGINT	identifikacijski broj korisnika koji je stvorio stazu unutar aplikacije
hill_id	BIGINT	identifikacijski broj visočja na kojem se staza nalazi

event Ovaj entitet modelira jedan događaj koji stvara korisnik unutar naše aplikacije. Jedan događaj počinje u određeno vrijeme "start_date", te završava u određeno vrijeme: "end_date". Svaki događaj, odnosno planinarski izlet sastoji se od jedne ili više planinarskih staza koje se obilaze tijekom tog planinarskog izleta. Osim toga, planinarski izlet ima svoj opis: "description", gdje korisnik može reći više pojedinosti o samom planinarskom izletu. Osim toga, ovaj entitet sadrži attribute: "event_id", "name", "date_created" i "author_id" koji predstavljaju jedinstveni identifikacijski broj događaja, naziv događaja, vrijeme kada je korisnik stvorio događaj unutar aplikacije te jedinstveni identifikacijski broj korisnika koji je stvorio događaj. Atribut "user_id" je strani ključ koji se odnosi na tablicu **user** te se radi o Many-To-One vezi.

event - (Planinarski događaj)		
event_id	BIGINT	jedinstveni identifikacijski broj planinarskog izleta
name	VARCHAR	naziv događaja
description	VARCHAR	pojedinosti vezane uz događaj
start_date	TIMESTAMP	vrijeme i datum početka planinarskog izleta
end_date	TIMESTAMP	vrijeme i datum završetka planinarskog izleta
date_created	TIMESTAMP	vrijeme i datum stvaranja izleta unutar aplikacije
user_id	BIGINT	jedinstveni identifikacijski broj korisnika koji je stvorio događaj unutar aplikacije

badge Ovaj entitet modelira jedan bedž, odnosno priznanje koje korisnik može zaslužiti kao planinar. Korisnik priznanja zaslužuje za svoju aktivnost koja se gleda na osnovu arhive planinarskih domova i staza, te se uzimaju u obzir značajke kao što su: broj posjećenih planinarskih domova, broj odraćenih planinarskih staza i sl. Entitet sadrži atribute "id", "name" i "description" koji predstavljaju jedinstveni identifikacijski broj priznanja, naziv priznanja te opis priznanja.

badge - (Priznanje)		
id	BIGINT	jedinstveni identifikacijski broj priznanja
name	VARCHAR	naziv priznanja
description	VARCHAR	opis priznanja

message Ovaj entitet modelira poruke koje korisnik može slati administratoru u slučaju npr. otvaranja novog planinarskog doma. Sadrži atribute: "id", "status", "error", "message_name", "message_content" i "user_id" koji redom predstavljaju jedinstveni identifikacijski broj poruke, status poruke, mjesto gdje je prijavljena greška, naslov poruke, sadržaj poruke i id korisnika koji je poslao poruku.

message - (Poruke administratoru)		
id	BIGINT	jedinstveni identifikacijski broj poruke
status	VARCHAR	status poruke
error	VARCHAR	mjesto gdje je nastala pogrška
message_name	VARCHAR	naslov poruke
message_content	VARCHAR	sadržaj poruke
user_id	BIGINT	jedinstveni identifikacijski broj korisnika koji je stvorio poruku unutar aplikacije

friendships Ovo je vezna tablica koja sadrži Many-To-Many vezu između dva korisnika, a služi nam za modeliranje prijateljstava, odnosno pripadanja istoj planinarskoj zajednici. Sadrži atribute "current_user_id" i "friend_id" koji predstavljaju strane ključeve na tablicu **user** i govore nam tko su korisnici koji pripadaju istoj zajednici.

friendships - (Prijatelji)		
<u>current_user_id</u>	BIGINT	strani ključ koji se odnosi na jednog korisnika člana veze "prijatelji"
<u>friend_id</u>	BIGINT	strani ključ koji se odnosi na drugog korisnika člana veze "prijatelji"

friendship_request Ovaj entitet sadrži informaciju o posланом заhtjevu za prijateljstvo između 2 korisnika. Sadrži atributе: "sender" te "receiver" koji su oba strani ključevi iz tablice **user**, a predstavljaju identifikacijski broj pošiljatelja te primatelja zahtjeva za prijateljstvo. Radi se o refleksivnoj Many-to-Many vezi.

friendship_request - (Zahtjevi za prijateljstvom)		
<u>sender</u>	BIGINT	strani ključ korisnika koji šalje zahtjev za prijateljstvom iz tablice user
<u>receiver</u>	BIGINT	strani ključ korisnika koji prima zahtjev za prijateljstvom iz tablice user

friendships_notifications Ovaj entitet sadrži informaciju o Many-To-Many odnosu između dva korisnika, a služi za modeliranje obavijesti korisnicima o prihvaćenim zahtjevima za prijateljstvo. Sadrži atributе "friendships_notifications_sender_id" i "friendships_notifications_receiver_id" koji predstavljaju strane ključeve na tablicu user i govore koji je korisnik kojem poslao zahtjev za prijateljstvo.

friendships_notifications		
<u>sender_id</u>	BIGINT	strani ključ korisnika koji je poslao zahtjev iz tablice user
<u>receiver_id</u>	BIGINT	strani ključ korisnika koji je primio zahtjev iz tablice user

user_badge Ovaj entitet sadrži informaciju o Many-To-Many odnosu između bedževa (priznanja) i korisnika. Sadrži atributе: "user_id", "badge_id" i "date_recieved" koji redom predstavljaju identifikacijski broj korisnika, identifikacijski broj priznanja te vrijeme dobivanja priznanja.

user_badge		
<u>user_id</u>	BIGINT	strani ključ korisnika kojem je pripisan pojedini bedž iz tablice user
<u>badge_id</u>	BIGINT	strani ključ bedža kojeg je dobio pojedini korisnik iz tablice badge
date_recieved	DATE	datum dobivanja pojedinog bedža za određenog korisnika

event_attendance Modelira Many-To-Many vezu između korisnika i događaja (planinarskog izleta) na kojemu korisnik želi sudjelovati. Sadrži atribute "user_id" te "event_id" koji predstavljaju jedinstveni identifikator korisnika koji sudjeluje na događaju te jedinstveni identifikator događaja.

event_attendance		
<u>user_id</u>	BIGINT	strani ključ korisnika koji će prisustvovati događaju iz tablice user
<u>event_id</u>	BIGINT	strani ključ događaja kojemu određena osoba pristupa iz tablice event

community_event_mountain_path Ovaj entitet sadrži informaciju o Many-To-Many vezi između planinarske staze i nekog planinarskog događaja. Sadrži atribute: "path_id", "event_id", "id" i "date_traveled" koji predstavljaju identifikacijski broj planinarske staze, identifikacijski broj nekog planinarskog događaja, identifikacijski broj tablice. Osim toga, sadrži atribut "date_traveled" koji nam govori o rednom broju dana tijekom kojeg se na određenom planinarskom događaju odraćuje određena planinarska staza.

community_event_mountain_path		
<u>id</u>	BIGINT	jedinstveni identifikacijski broj
<u>path_id</u>	BIGINT	strani ključ staze koja je dio nekog događaja iz tablice mountain_path
<u>event_id</u>	BIGINT	strani ključ događaja u kojem se pojedina staza odraćuje, iz tablice event

community.event.mountain_path		
date_traveled	INTEGER	redni broj dana koliko se unutar jednog događaja obilazi određena staza

mountain_lodge.utility Vezna tablica koja modelira Many-to-Many vezu između planinarskih domova i infrastrukturnih značajki koje ti domovi posjeduju. Sadrži atribute "lodge_id" te "utility_id" koji predstavljaju identifikator planinarskog doma te identifikator odgovarajuće značajke.

mountain_lodge.utility		
<u>lodge_id</u>	BIGINT	strani ključ kojem je pripisan pojedini planinarski dom iz tablice mountain_lodge
<u>utility_id</u>	BIGINT	strani ključ kojem je pripisana pojedina značajka iz tablice utility

mountain_path.grade Vezna tablica koja nam govori o ocjenama koje pojedini korisnik dodijeljuje pojedinoj planinarskoj stazi. Radi se o Many-To-Many veznoj tablici. Sadrži atribute "user_id", "path_id" te "grade", koji redom predstavljaju identifikacijski broj korisnika, identifikacijski broj planinarske staze te ocjenu koju je taj korisnik dodijelio toj stazi.

mountain_path.grade		
<u>user_id</u>	BIGINT	strani ključ korisnika koji daje ocjenu iz tablice user
<u>path_id</u>	BIGINT	strani ključ planinarske staze koja se ocjenjuje iz tablice mountain_path
grade	INTEGER	ocjenu koju je korisnik dodijelio za konkretnu stazu

path_user.wishlist Vezna tablica koja modelira željene planinarske staze za pojedinog korisnika. To su staze koje korisnik ima namjeru prepješaćiti, ali još nije uspio. Sadrži atribute "user_id" i "path_id" koji predstavljaju identifikator korisnika te identifikator staze koju korisnik želi prepješaćiti.

path_user_wishlist		
<u>user_id</u>	BIGINT	strani ključ korisnika koji želi prepješaćiti neku stazu iz tablice user
<u>path_id</u>	BIGINT	strani ključ staze koju korisnik želi prepješaćiti iz tablice mountain_path

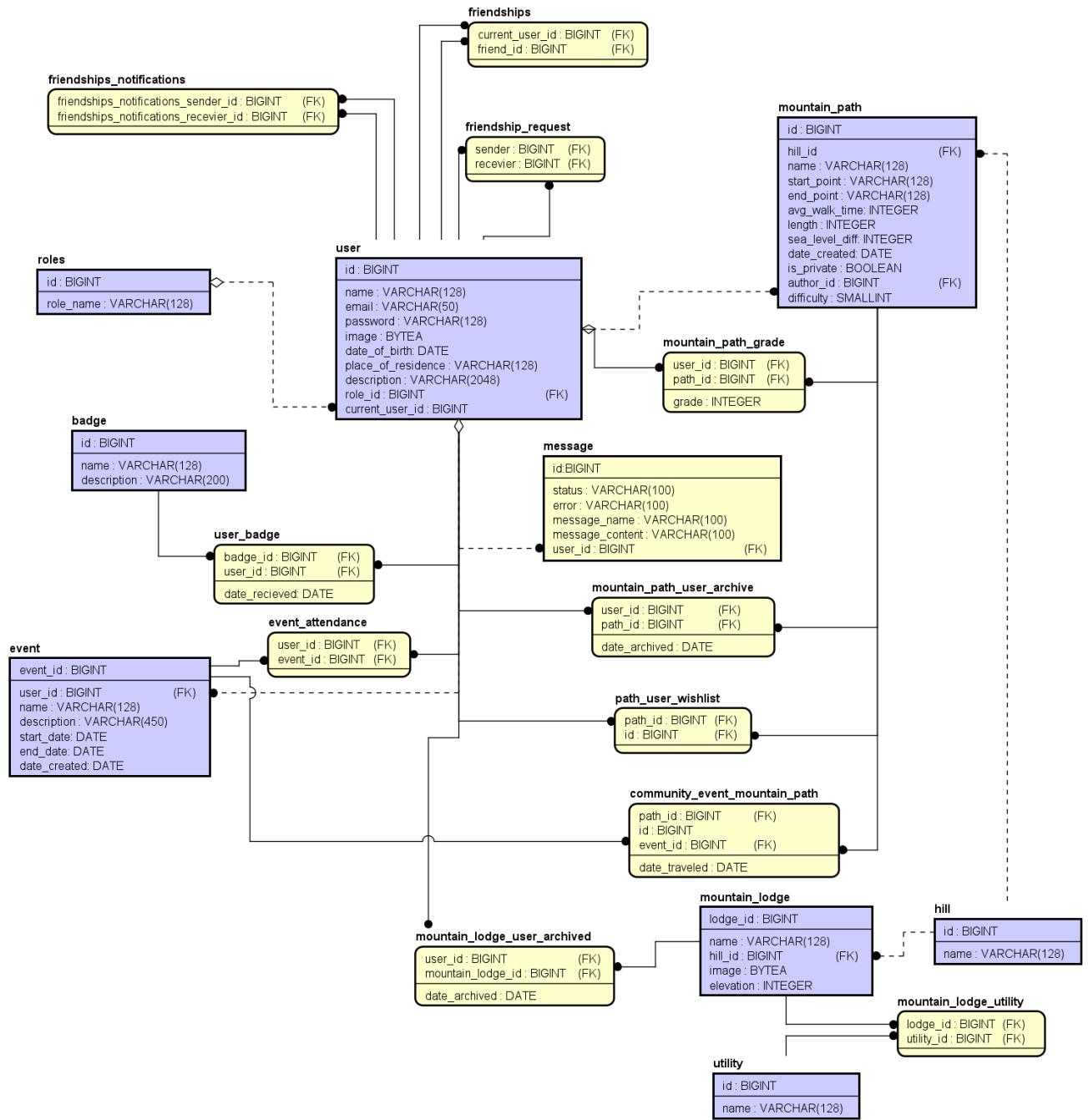
mountain_path_user_archived Ovaj entitet sadrži informaciju o Many-To-Many odnosu između pojedinog korisnika i staza koje je već prepješaćio, odnosno ovaj entitet modelira arhivu staza.

mountain_path_user_archived		
<u>user_id</u>	BIGINT	strani ključ korisnika koji je prepješaćio pojedinu stazu iz tablice user
<u>path_id</u>	BIGINT	strani ključ staze koju je konkretni korisnik prepješaćio iz tablice mountain_path
<u>date_archived</u>	DATE	datum kojeg je određeni korisnik prepješaćio određenu stazu

mountain_lodge_user_archived Ovaj entitet sadrži informaciju o Many-To-Many odnosu između pojedinog korisnika i domova koje je već posjetio, odnosno ovaj entitet modelira arhivu planinarskih domova.

mountain_lodge_user_archived		
<u>user_id</u>	BIGINT	strani ključ korisnika koji je prepješaćio pojedinu stazu iz tablice user
<u>mountain_lodge_id</u>	BIGINT	strani ključ doma kojeg je konkretni korisnik posjetio iz tablice mountain_lodge
<u>date_archived</u>	DATE	datum kojeg je određeni korisnik prepješaćio određenu stazu

4.1.2 Dijagram baze podataka

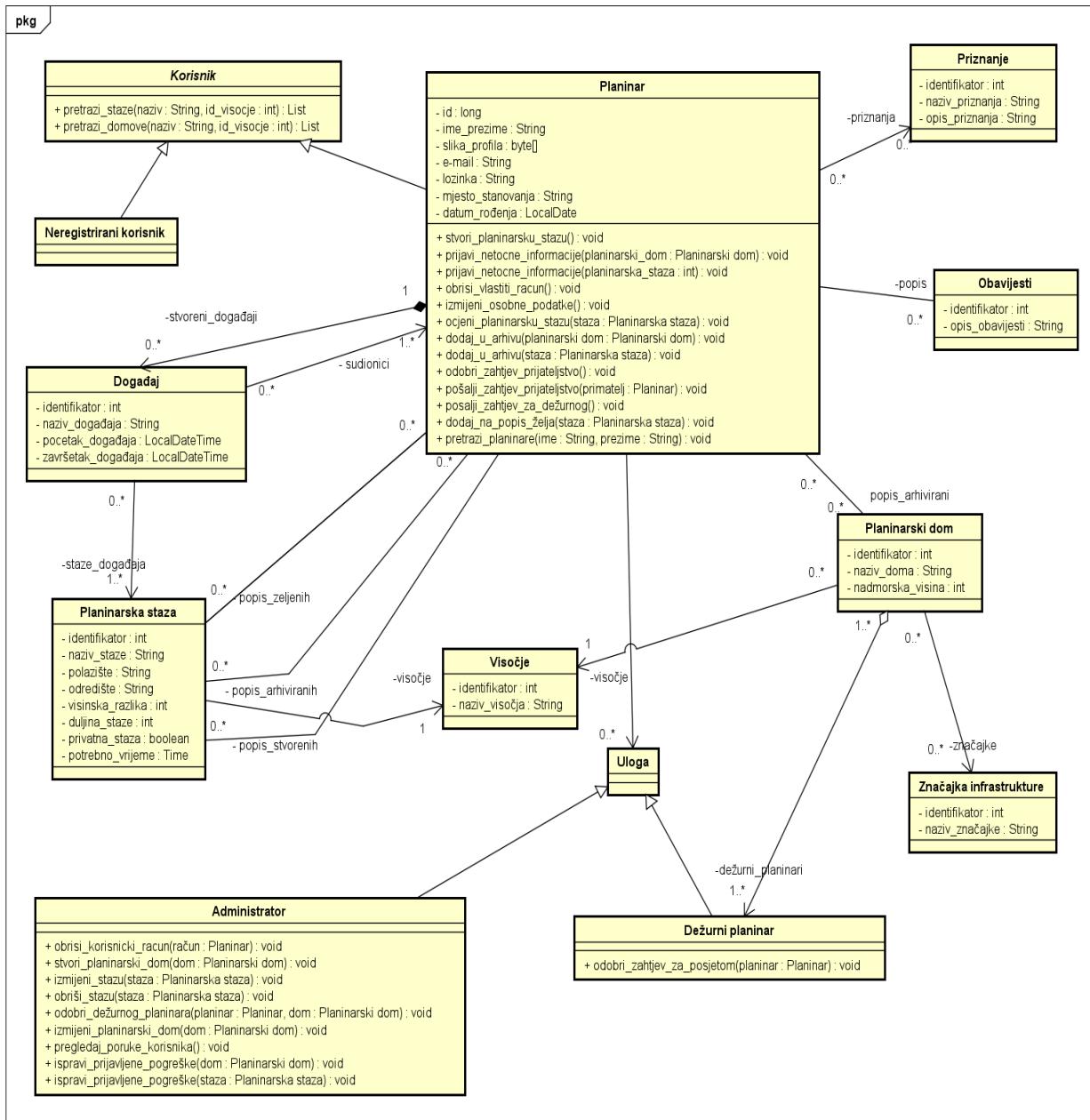


Slika 4.3: Dijagram baze podataka

4.2 Dijagram razreda

4.2.1 Konceptualni model dijagrama razreda

Prvi prikazani dijagram je konceptualni model dijagrama razreda. Na njemu su idejno prikazani razredi, njihove funkcionalnosti te odnosi između razreda. Svi razredi napravljeni su obzirom na obrasce uporabe i opise funkcionalnih zahtjeva.

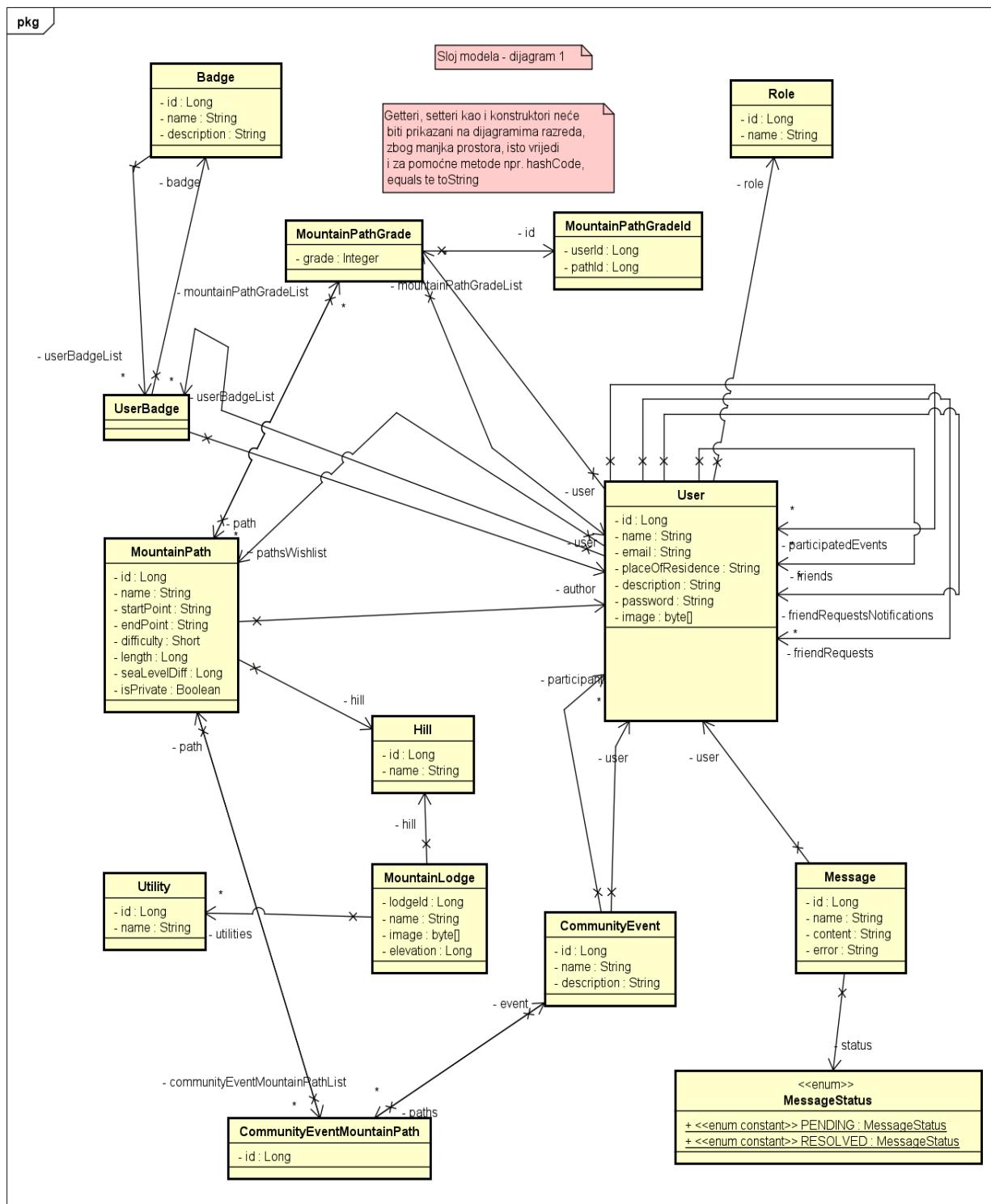


Slika 4.4: Dijagram razreda - konceptualni model

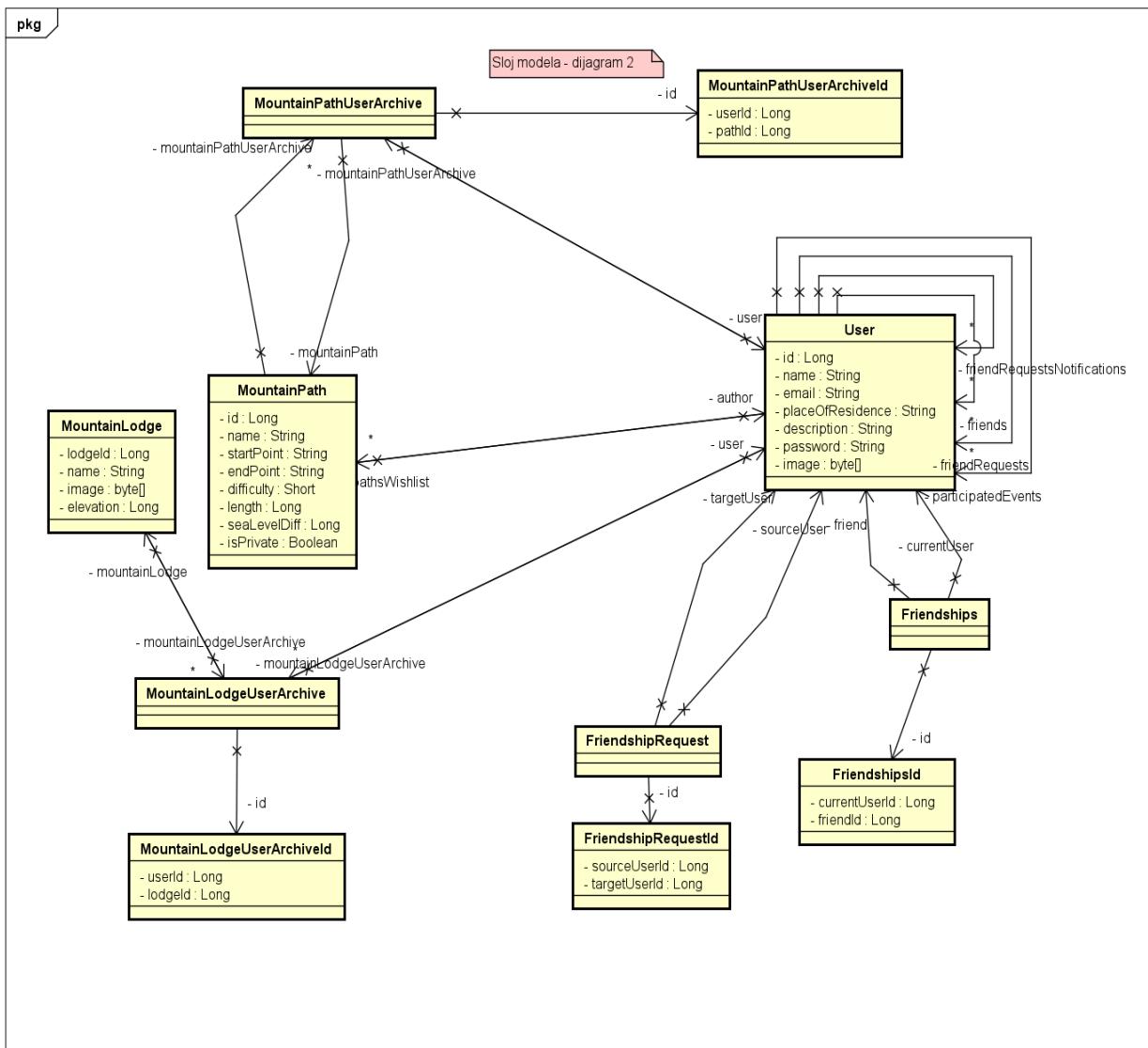
4.2.2 Implementacijski dijagrami razreda - model sustava

Sloj modela

Na dijagramima razreda nismo navodili podrazumijevane metode, gettere, setttere te konstruktore. Na prva dva implementacijska dijagrama razreda prikazan je sloj modela. Temeljni razred je razred *User*, za koji vidimo da ima najviše asocijacije s drugim razredima kao i nekoliko refleksivnih veza. Refleksivne veze redom čine članske variable *participatedEvents*, *friends*, *friendRequestNotifications*, *friendRequests*, a one predstavljaju događaje na kojima korisnik sudjeluje, popis prijatelja, popis obavijesti o prihvaćenim/odbijenim zahtjevima za prijateljstvo kao i popis zahtjeva za prijateljstvo. Osim toga, na prvom dijagramu *User* je povezan i s razredom *MountainPathGrade*, a ta veza predstavlja popis ocjena koje je korisnik dodijelio pojedinoj planinarskoj stazi, te je povezan s razredom *MountainPath*, a ta veza predstavlja popis planinarskih staza koje je korisnik dodao u svoje favorite. Razred *Role* predstavlja ulogu koju korisnik ima ako je registriran unutar aplikacije, a može biti **Admin** ili **Planinar**. Razred *Badge* predstavlja priznanja koje korisnik može dobiti, a razredom *UserBadge* povezan je s korisnikom. Razred *MountainLodge* predstavlja planinarski dom. Razred *MountainLodgeUserArchive* predstavlja vezu između korisnika i njegovih posjećenih planinarskih domova. Na isti način razred *MountainPathUserArchive* predstavlja vezu između korisnika i njegovih pre-pješaćenih planinarskih staza. Razredi *FriendshipRequest* te *Friendship* vezani su za obavijesti o zahtjevima za prijateljstvo te same veze prijateljstva između dva korisnika. Razred *Utility* predstavlja infrastrukturne pogodnosti koje pruža pojedini planinarski dom, dok razred *Hill* predstavlja visočje na kojemu se nalazi planinarski dom ili planinarska staza. Razred *Message* predstavlja poruke/pogreške koje planinari šalju administratoru sustava. Nапослјетку razred *CommunityEvent* predstavlja neki planinarski događaj, a kao člansku varijablu ima *participants* koja predstavlja korisnike koji su se prijavili za sudjelovanje. Planinarski događaj je preko razreda *CommunityEventMountainPath* povezan s planinarskim stazama koje se obilaze unutar tog planinarskog događaja.



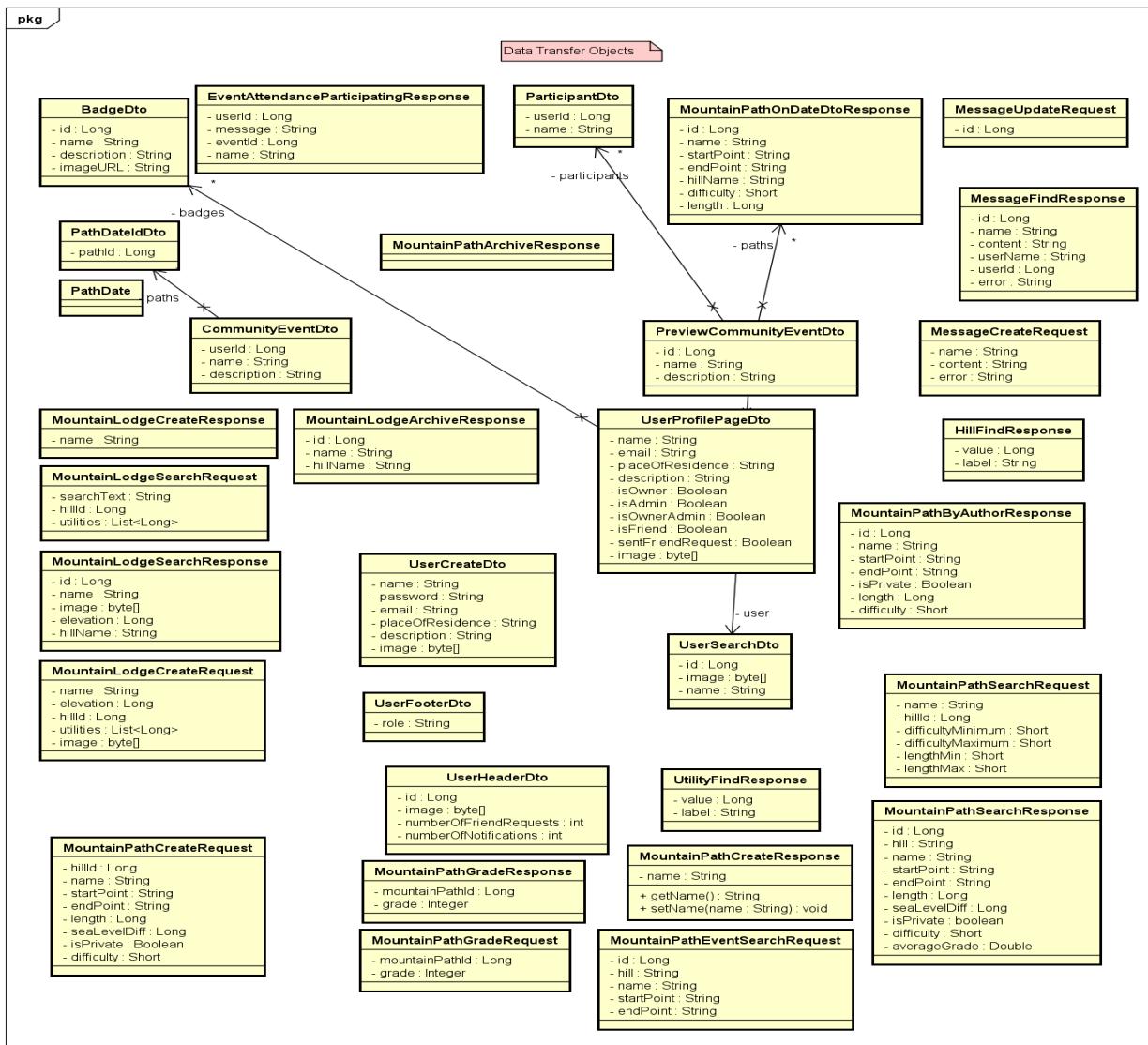
Slika 4.5: Dijagram razreda - sloj modela prvi dijagram



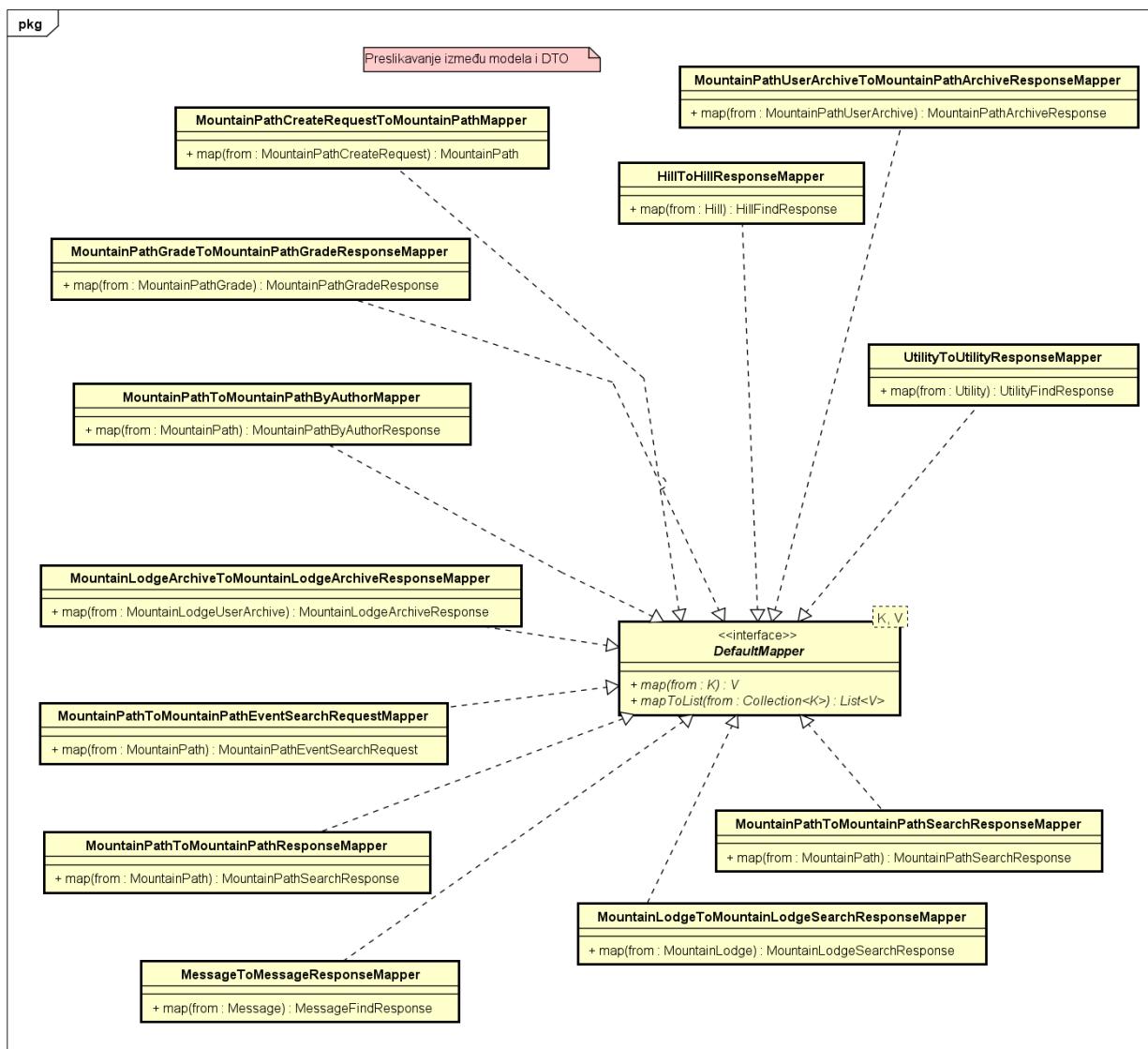
Slika 4.6: Dijagram razreda - sloj modela drugi dijagram

Objekti za prijenos podataka i preslikavanja iz sloja modela

Na prvom od sljedeća dva dijagrama prikazani su objekti za prijenos podataka između klijenta i poslužitelja. Oni omogućuju da sloj nadglednika nikada ne komunicira sa stvarnim modelima. Iz imena DTO razreda možemo vidjeti za koji je razred taj objekt zadužen. Na drugom dijagramu prikazani su razredi koji služe za preslikavanje između sloja modela i objekata za prijenos podataka. Svi ti razredi implementiraju parametrizirano sučelje *DefaultMapper*.



Slika 4.7: Dijagram razreda - razredi za prijenos podataka



Slika 4.8: Dijagram razreda - razredi za preslikavanje između sloja modela i DTO

Sloj nadglednik - servis - rezervorij

Ovaj sloj predstavlja sve operacije koje naš sustav obavlja. Razrađen je po oblikovnom obrascu "MVC". Zapravo se ovaj sloj sastoji od tri podsloja, a to su:

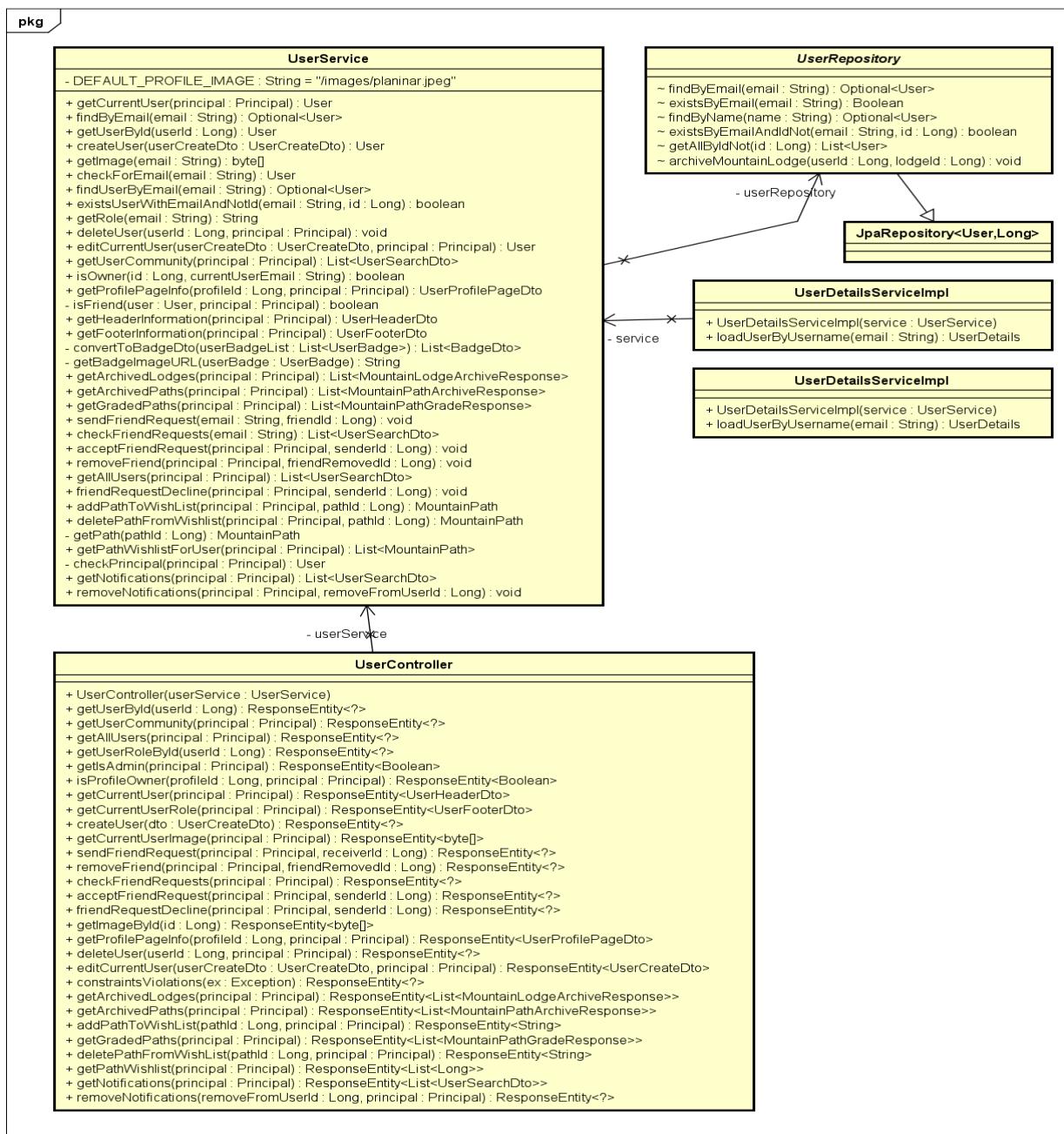
- Sloj rezervorija - metode pristupa bazi podataka, dohvati modela
- Sloj servisa - tu se obavlja sva naša poslovna logika, pozivaju se metode rezervorija dohvaćaju se podaci
- Sloj nadglednika - prima zahtjeve od korisnika i poziva metoda servisnog sloja

Dijagram ovog sloja podijeljeni su na nekoliko manjih dijagrama. Razredi koji su trenutno implementirani u ovom sloju prikazani su na dijagramu slijedno, od najnižeg podsloja odnosno pristupa bazi podataka pa sve do podsloja nadglednika koji prima i delegira zahtjeve korisnika. Dijagrami se sastoje od nekoliko nadglednika: *UserController*, *MountainLodgeController*, *MountainPathController*, *HillController*, *UtilityController*, *MountainLodgeUserArchiveController*, *MountainLodgePathArchiveController*, *MessageController* te *CommunityEventController*.

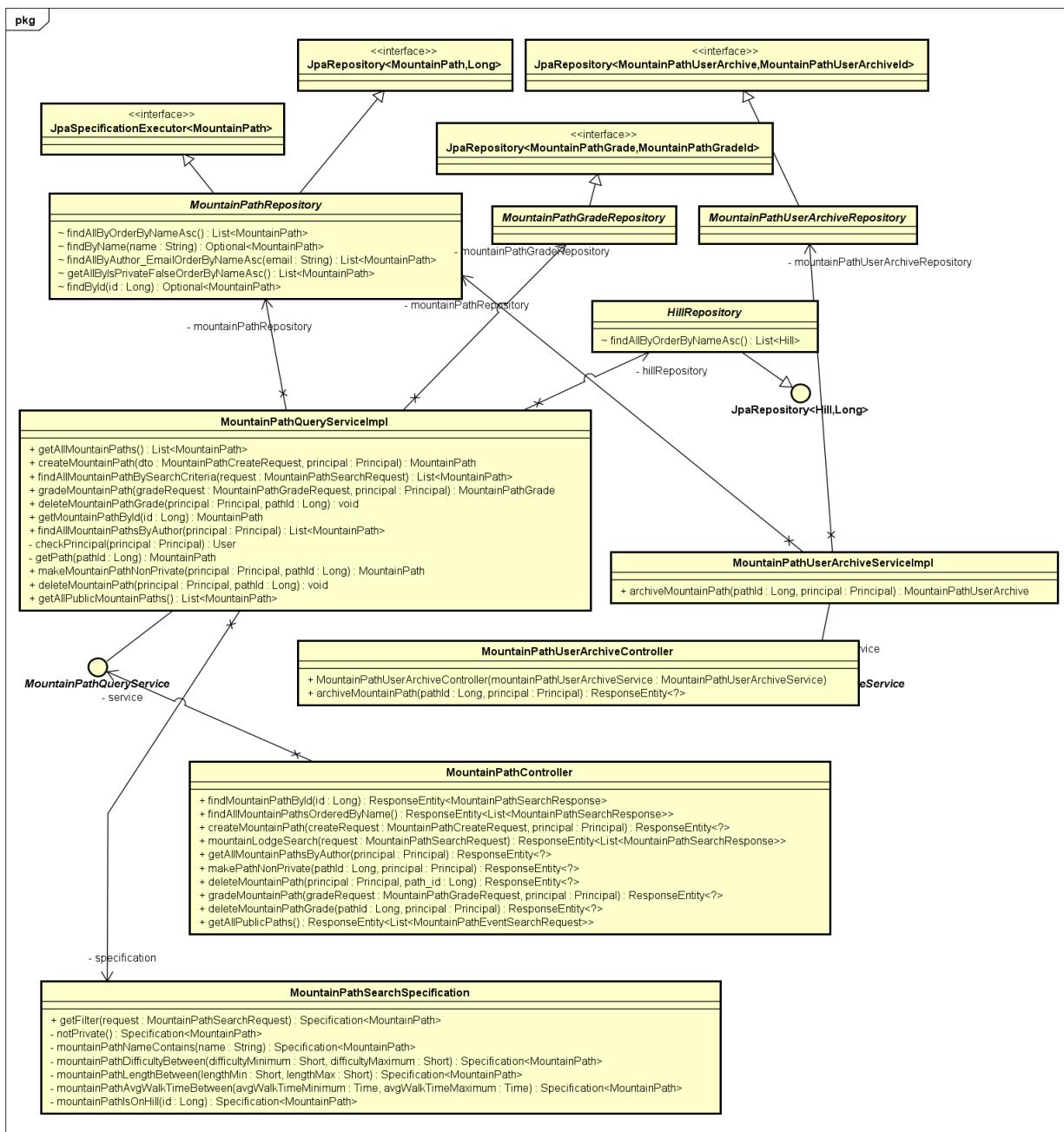
Podsloj nadglednika povezan je sa servisnim slojem preko konkretnog primjerka razreda servisnog sloja kojeg čine razredi: *UserService*, *MountainLodgeQueryServiceImpl*, *MountainPathQueryServiceImpl*, *HillQueryServiceImpl*, *UtilityQueryServiceImpl*, *UserDetailsService*, *UserBadgeService*, *MountainPathUserArchiveService*, *MountainLodgeUserArchiveService*, *MessageQueryService* te *CommunityEventService*, a oni predstavljaju implementacije odgovarajućih servisnih sučelja koji su također označeni na dijagramima.

Naposljetku, servisni sloj povezan je sa slojem rezervorija preko sučelja konkretnog rezervorija relevantnog za taj servis. U našem slučaju to su sučelja: *BadgeRepository*, *BadgeUserRepository*, *CommunityEventMountainPathRepository*, *CommunityEventRepository*, *FriendshipRequestRepository*, *FriendshipRepository*, *HillRepository*, *MessageRepository*, *MountainLodgeRepository*, *MountainLodgeUserArchiveRepository*, *MountainPathGradeRepository*, *MountainPathRepository*, *MountainPathUserArchiveRepository*, *RoleRepository*, *UserRepository* te *UtilityRepository*. Svako sučelje rezervorija nasljeđuje sučelje *JpaRepository* koje sadrži konkretnu implementaciju upita nad bazom podataka.

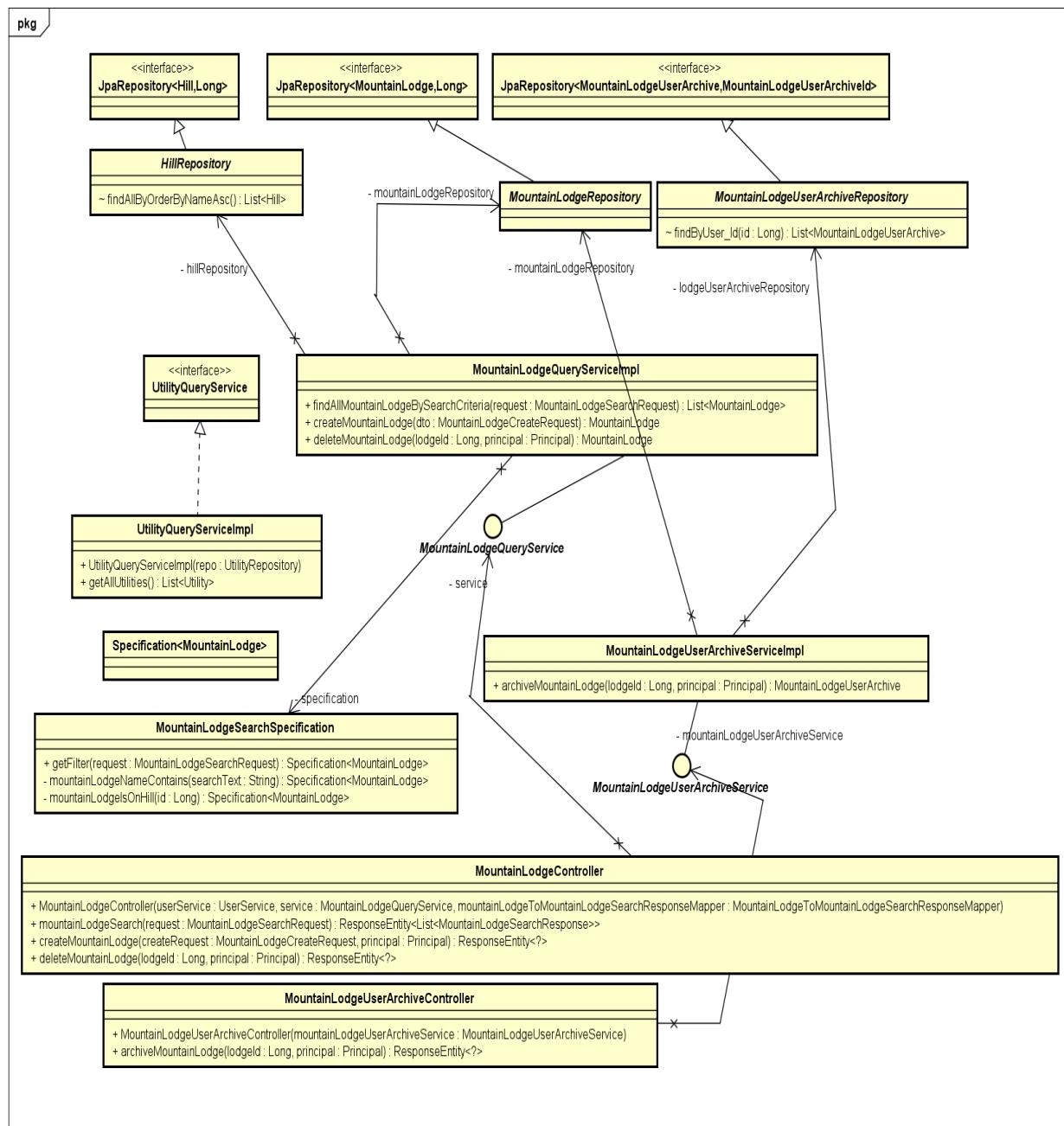
Sučelje *JpaSpecificationExecutor* omogućuje nam korištenje specifikacija *MountainLodgeSearchSpecification* i *MountainPathSearchSpecification* prilikom pretraživanja planinarskih domova i planinarskih staza iz baze podataka.



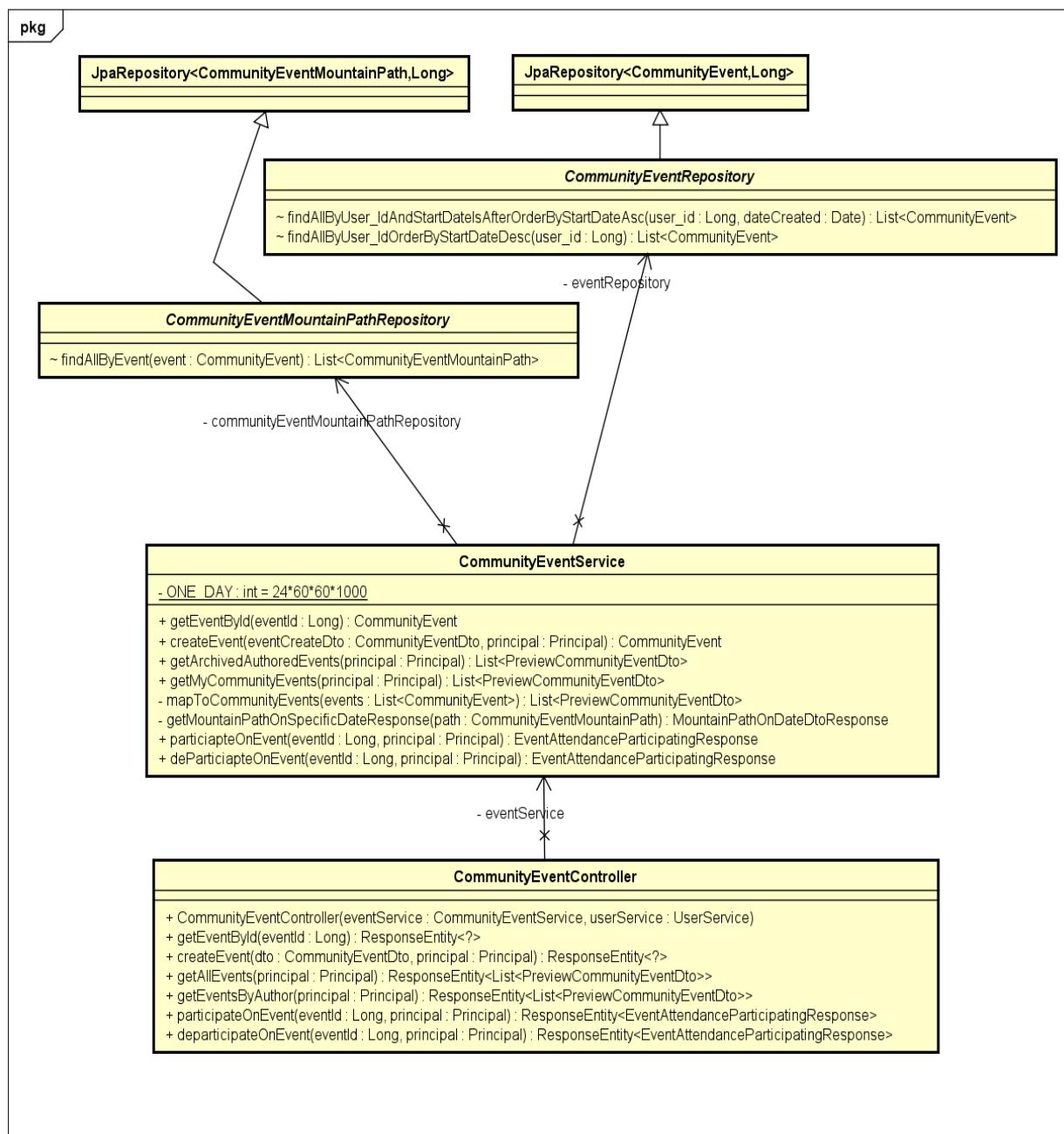
Slika 4.9: Dijagram razreda korisnik - sloj nadglednik - servis - repozitorij



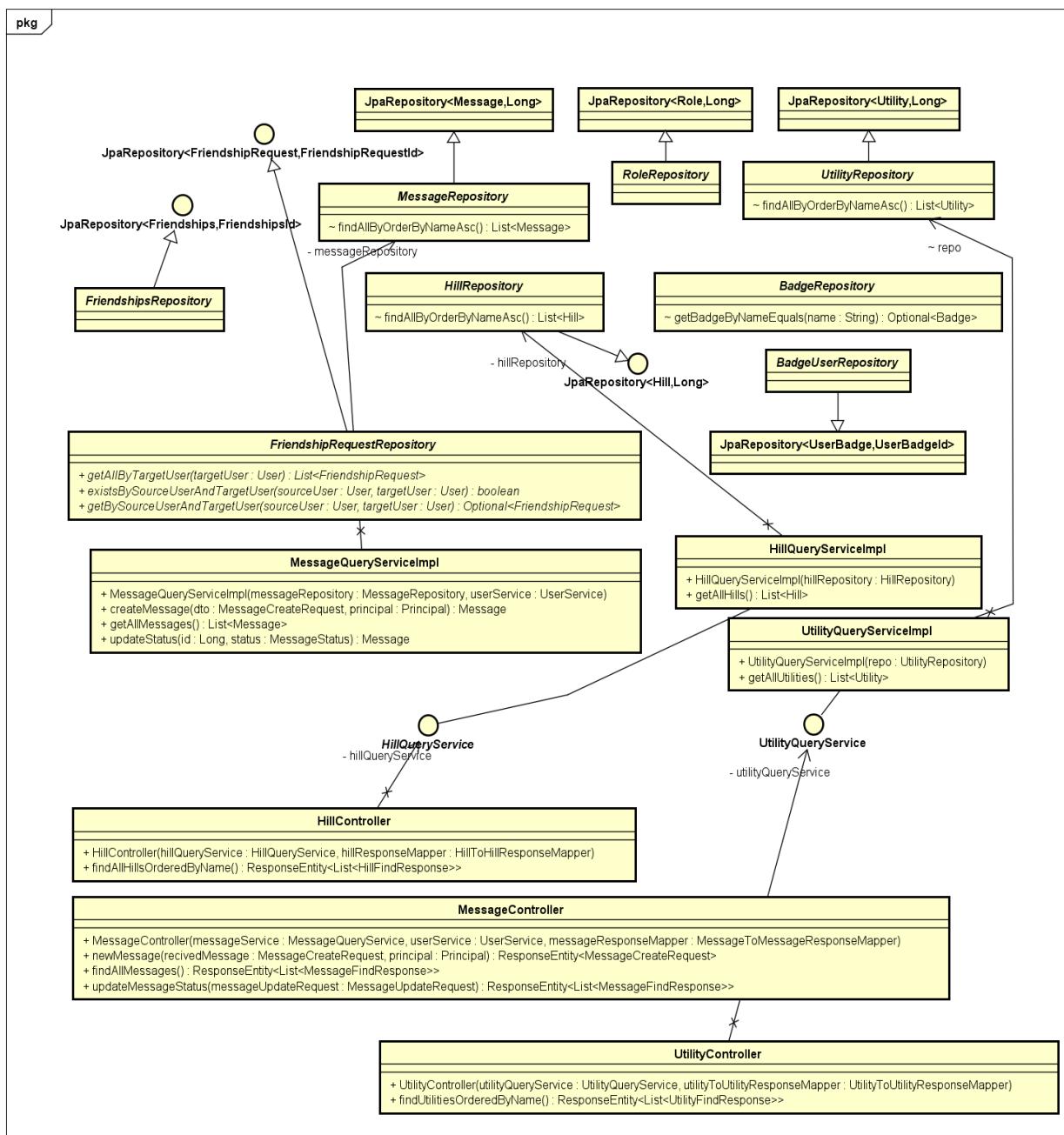
Slika 4.10: Dijagram razreda planinarske staze - sloj nadglednik - servis - repozitorij



Slika 4.11: Dijagram razreda planinarski dom - sloj nadglednik - servis - repozitorij



Slika 4.12: Dijagram razreda planinarski događaji - sloj nadglednik - servis - rezervorij



Slika 4.13: Dijagram razreda, ostalo - sloj nadglednik - servis - repozitorij

Razredi vezani uz sigurnost i pomoćni razredi

Prilikom prijave korisnika, zahtjev za prijavu se šalje na back end, točnije do nadglednika. Zahtjev se provjerava u razredu *JWTAuthenticationFilter* pomoću metode *attemptAuthentication* koja stvara instancu razreda *UserLogin* i šalje ga na provjeru.

Provjera se vrši tako da se zove metoda *loadUserByUsername* razreda *UserDetailsServiceImpl*, a ona provjerava postoji li korisnik koji sadrži e-mail koji je poslan unutar zahtjeva.

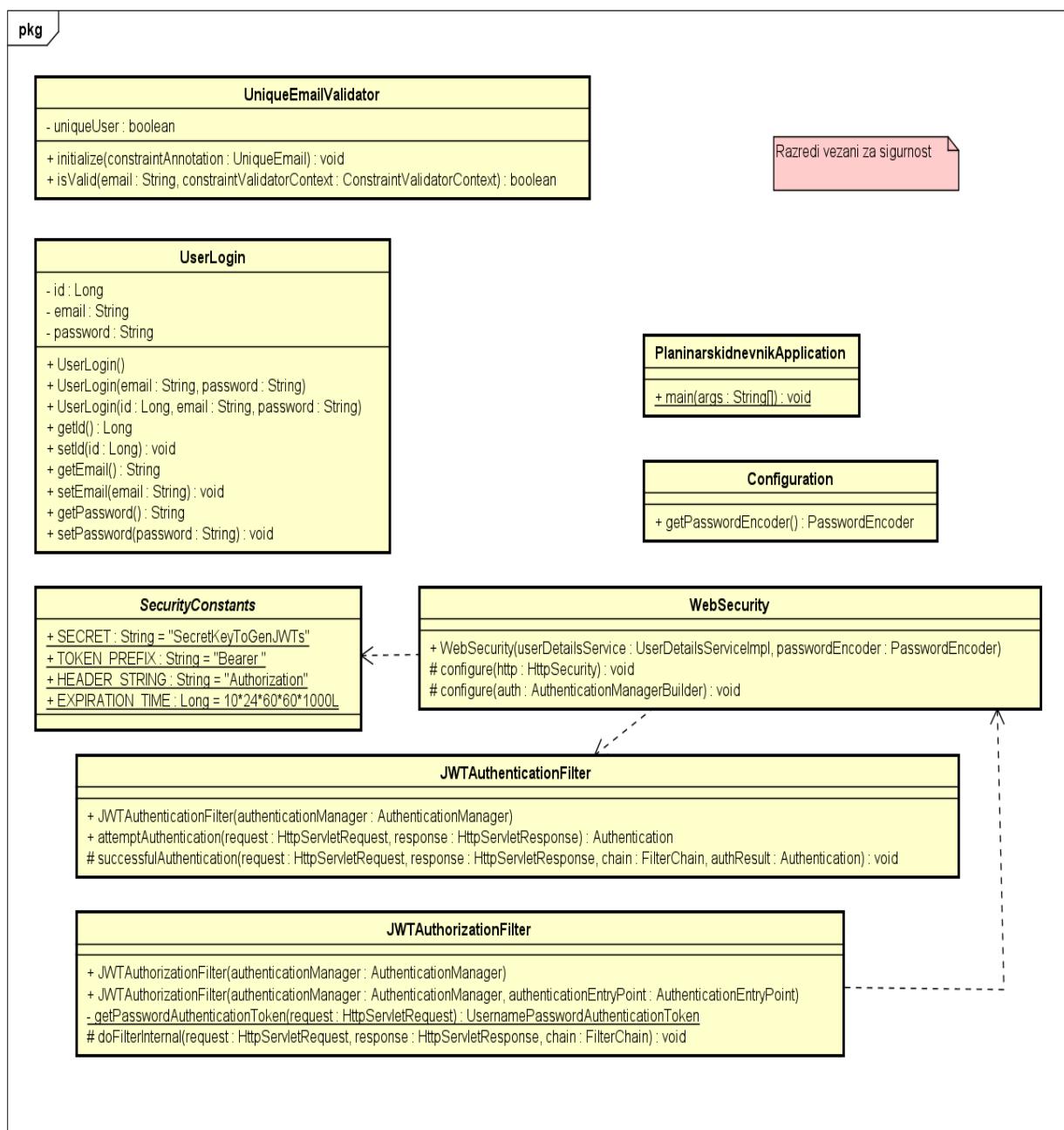
Nakon utvrđivanja da ta osoba postoji *Spring security modul* provjerava lozinku i druge podatke vezane uz prijavu. Nakon uspješne prijave pomoću metode *successfulAuthentication* se generira jedinstveni token (identifikator) i šalje korisniku. Svojstva tokena poput trajanja se određuju u razredu *SecurityConstants*.

Prijavljeni korisnik za vrijeme rada unutar Web aplikacije svaki zahtjev obavlja pomoću jedinstvenog tokena koji mu je dodijeljen, a koji se provjerava unutar razreda *JWTAuthorizationFilter*.

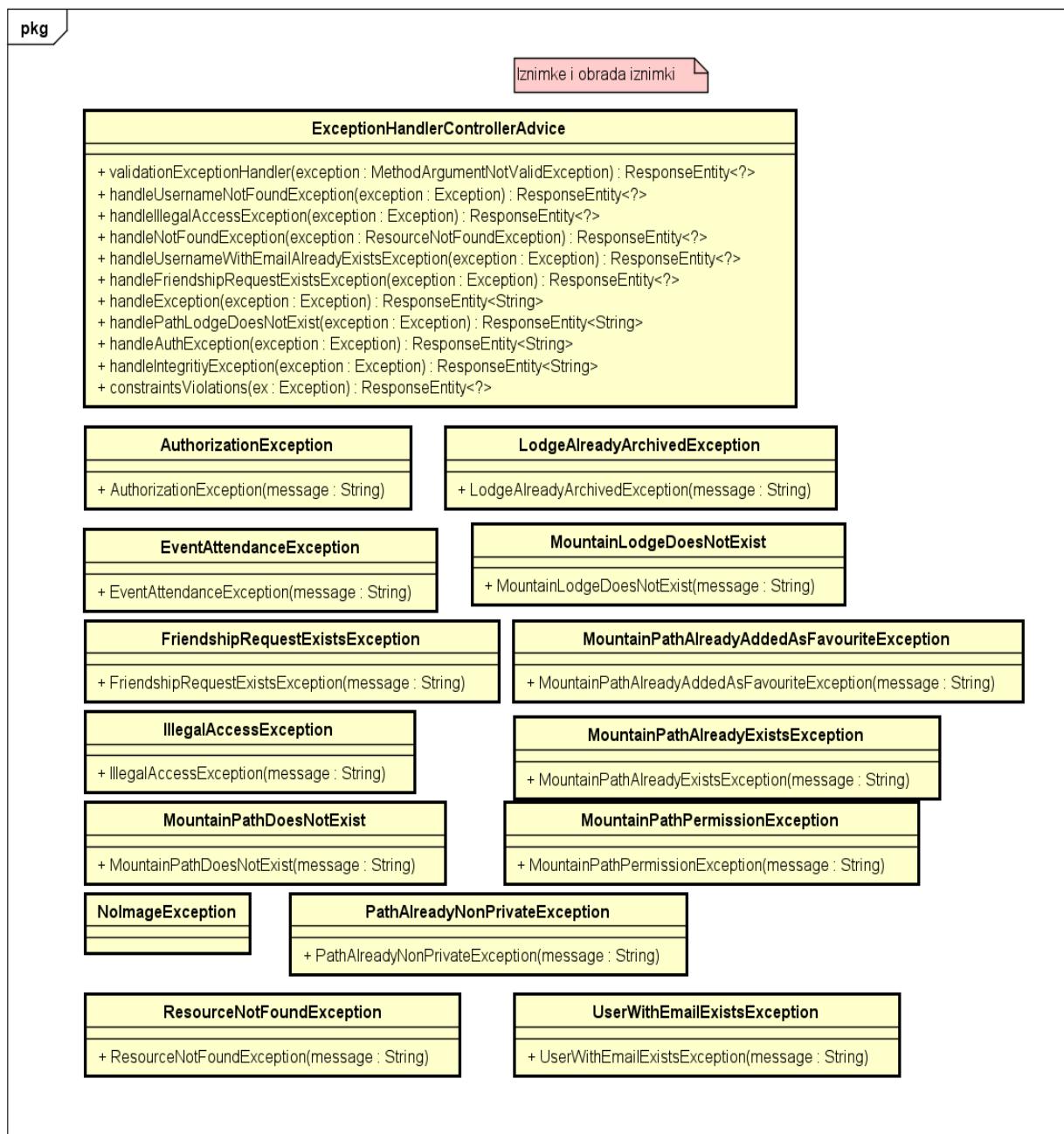
Metoda *getPasswordEncoder* razreda *Configuration* služi da bi definirali enkoder za zaštitu lozinki.

Razred *UniqueEmailValidator* prilikom registracije provjerava postoji li u bazi podataka korisnik kojem je e-mail isti e-mailu zahtjeva. Budući da ne možemo imati dva korisnika s istom adresom elektroničke pošte, u tom slučaju događa se iznimka *UserWithEmailExistsException*.

Na drugom dijagramu prikazane su iznimke koje je moguće izazvati na poslužiteljskoj strani, kao i razred koji obrađuje iznimke.



Slika 4.14: Dijagram razreda - sigurnost

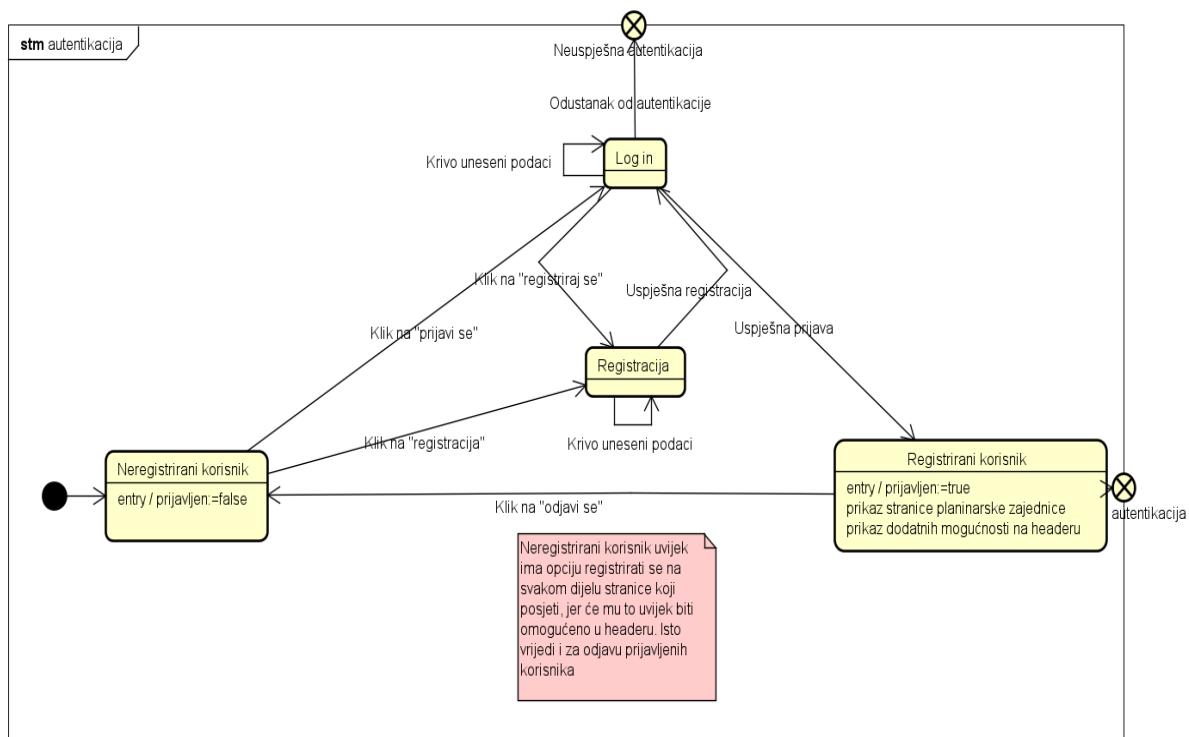


Slika 4.15: Dijagram razreda - iznimke i upravljanje iznimkama

4.3 Dijagram stanja

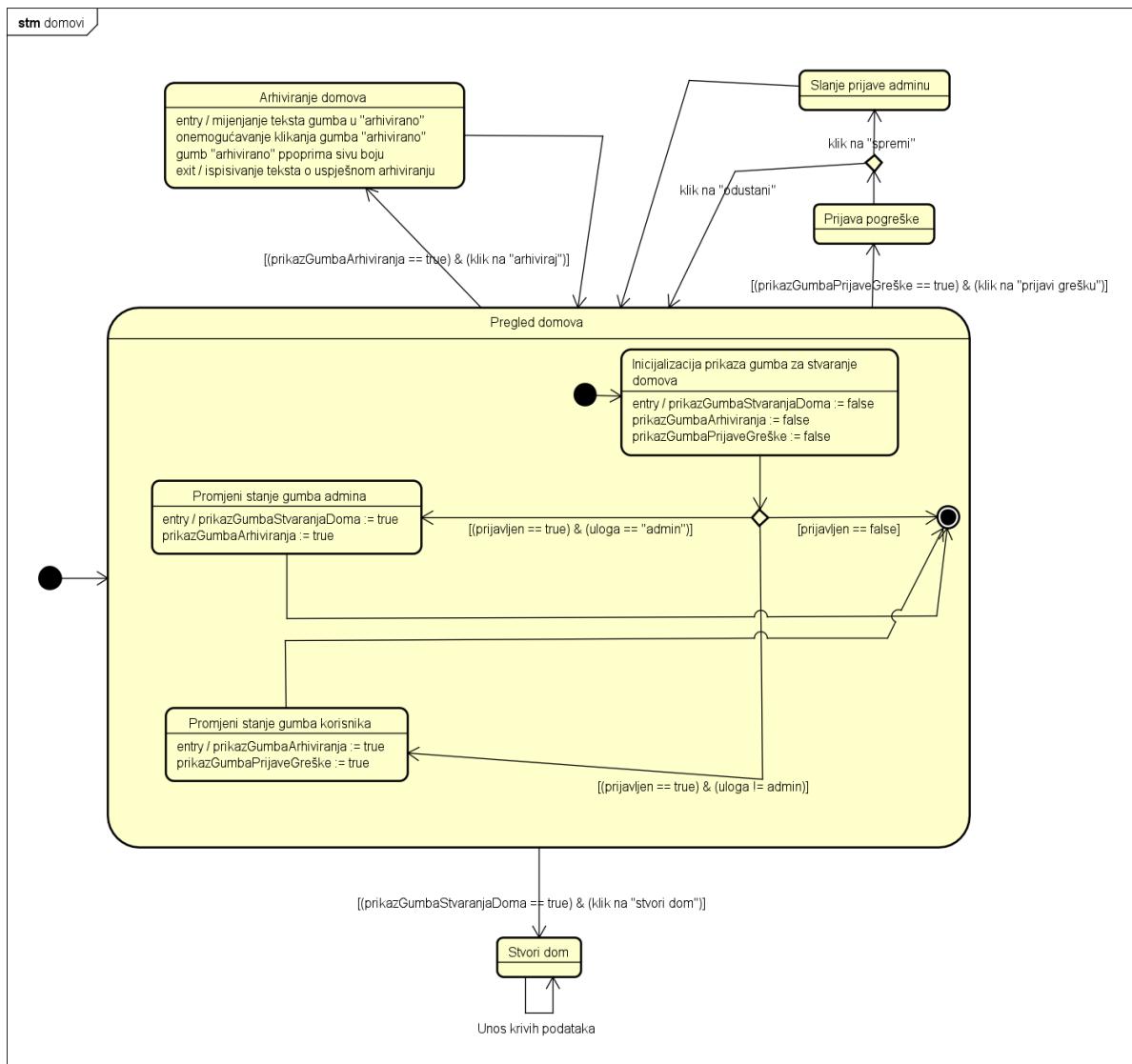
Dijagrami stanja prikazuju stanja objekata i njihov prelazak iz jednog stanja u drugo. Takav pogled nam dozvoljava veću preglednost sustava. Zbog čitljivosti smo mi naše dijagrame stanja podijelili u više odvojenih dijagrama.

Na slici 4.9 prikazan je proces autentikacije. Korisnik dolazi u aplikaciju u neprijavljenom stanju. Iz toga stanja može vršiti određene akcije, poput pregledavanja domova i staza. Da mu se omogući pregled profila, kreiranje staza i ako je admin, kreiranje domova, mora se ulogirati/registrirati. Taj proces je prikazan na dijagramu. Klikom na "log in" korisnik prelazi u stanje ulogiranja. Ako unese pogrešne informacije, stanje se resetira. Ako klikne na "nemate račun", otići će na stranicu registracije. Ako unese pogrešne podatke, stanje mu se resetira. Na registraciju može otići i direktno iz headera. Kada se uspješno registrira, vraća ga na log in. Kada se ulogira, onda ima globalno stanje "prijavljen" što mu omogućava pristup većem broju sadržaja same stranice.



Slika 4.16: Dijagram stanja - autentikacija

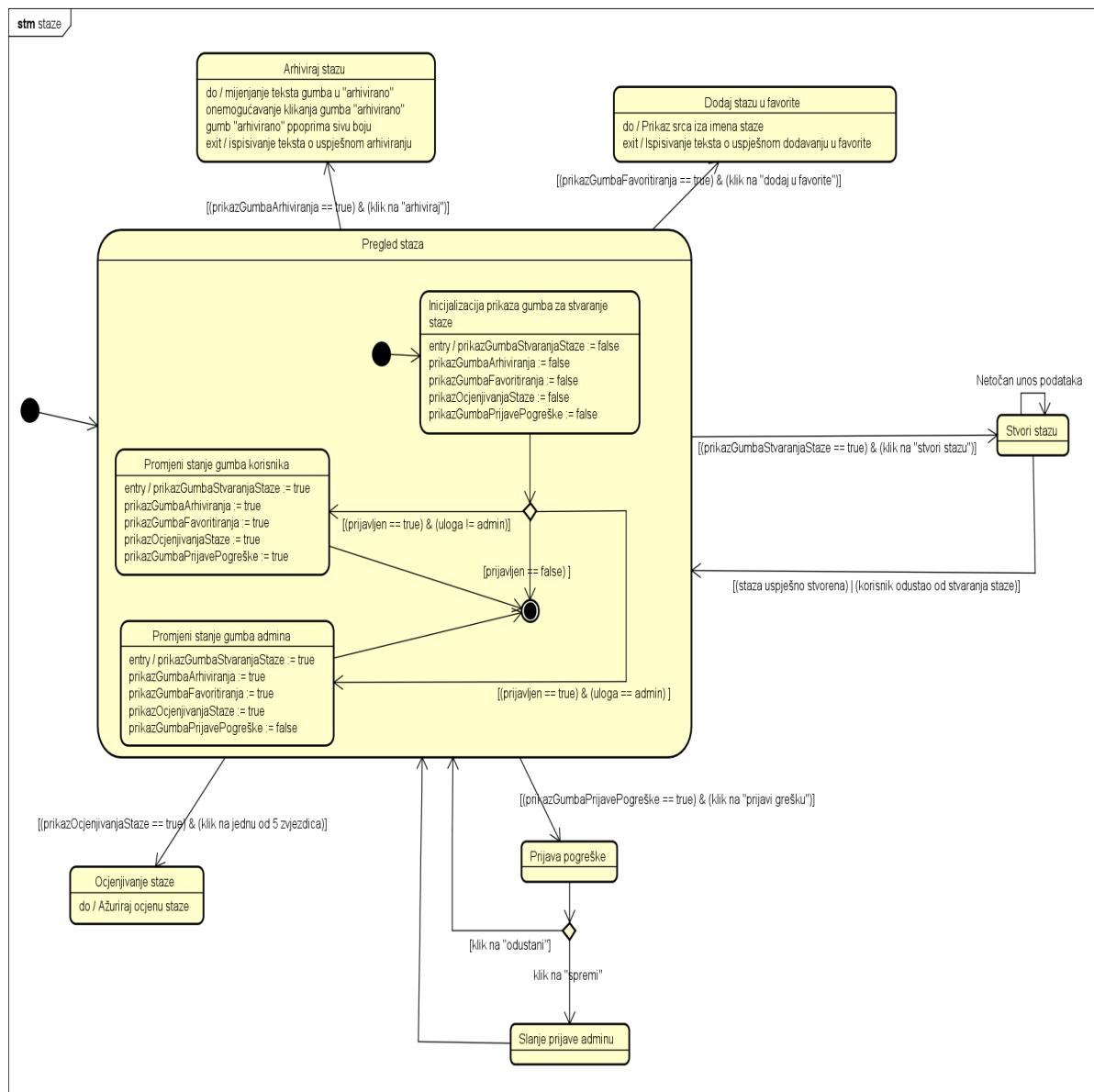
Na slici 4.10 prikazana su stanja vezana za domove. Početno stanje provjerava je li korisnik admin, i ako je, prikazuje mu gumb za kreiranje domova. Ako korisnik nije admin, ne prikazuje mu se taj gumb i ne može kreirati domove, samo pregledavati. Prijavljenom korisniku se još prikazuju gumbi za arhiviranje staze i prijavljivanje pogreške na stazi



Slika 4.17: Dijagram stanja - domovi

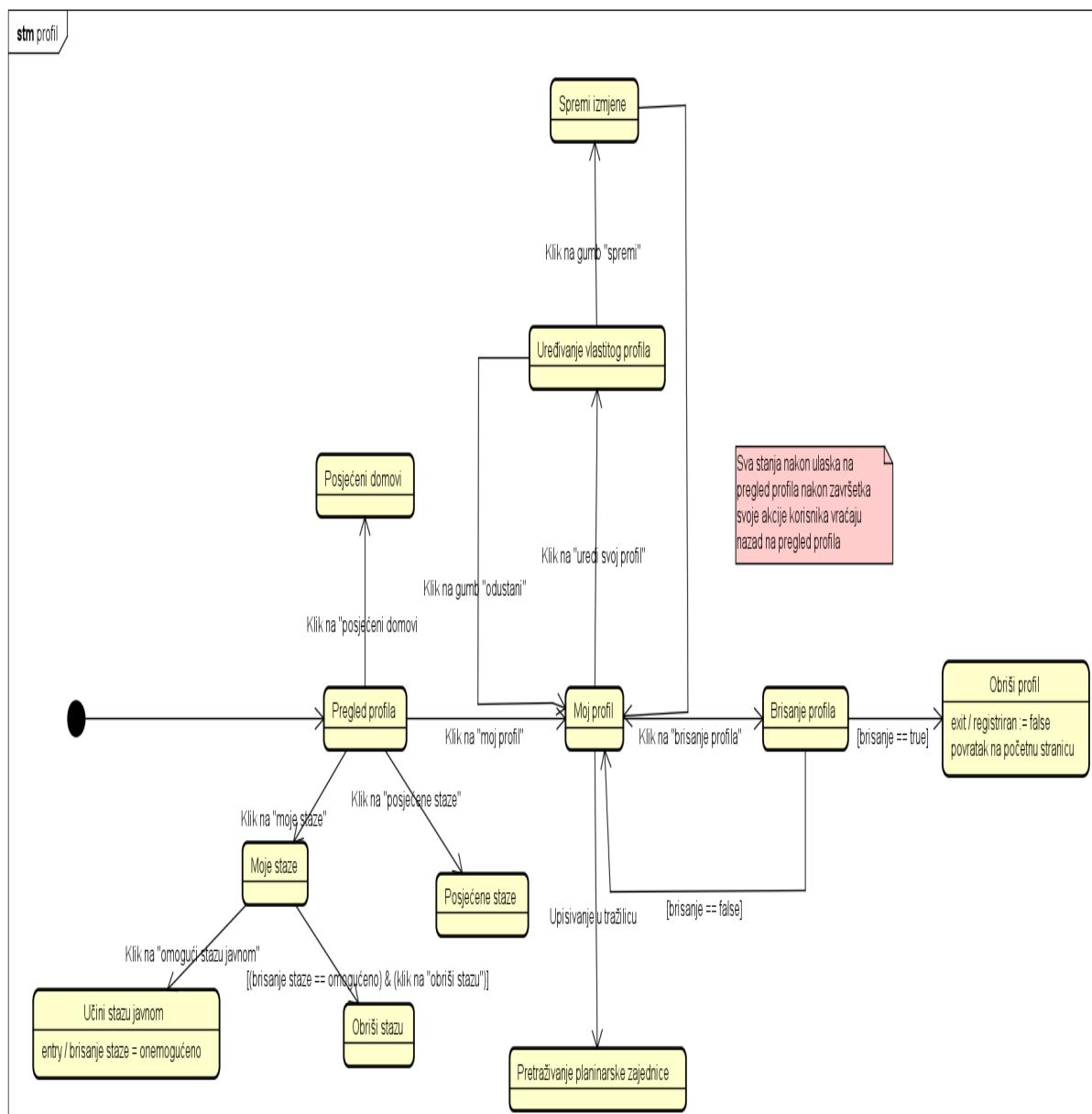
Na slici 4.11 prikazana su stanja vezana za planinarske staze. Početno stanje provjerava je li korisnik prijavljen, i ako je, prikazuje mu gumb za kreiranje staza. Ako korisnik nije prijavljen, ne prikazuje mu se taj gumb i ne može kreirati staze, samo pregledavati. Dodatno se prijavljenom korisniku prikazuju gumbi za doda-

vanje u favorite, ocjenjivanje staze, arhiviranje staze i prijavljivanje pogreške na stazi



Slika 4.18: Dijagram stanja - staze

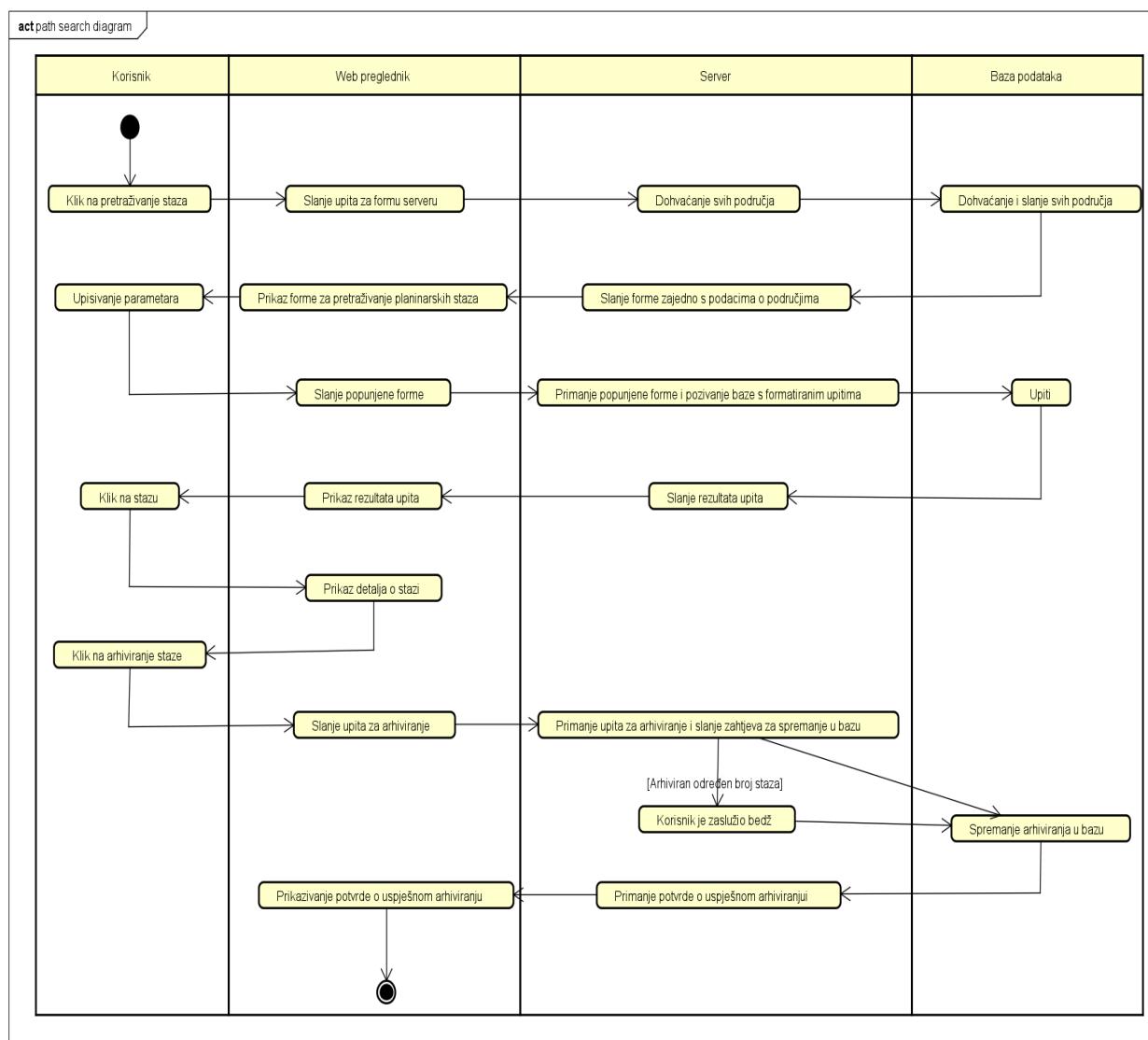
Na slici 4.12 prikazana su stanja vezana za korisnički profil. Ako je prijavljen, korisnik može otići na svoj profil gdje su mu omogućene radnje poput pregleda profila, uređivanje vlastitog profila, pretraživanja planinarske zajednice, brisanje profila i potvrda brisanja profila.



Slika 4.19: Dijagram stanja - profil

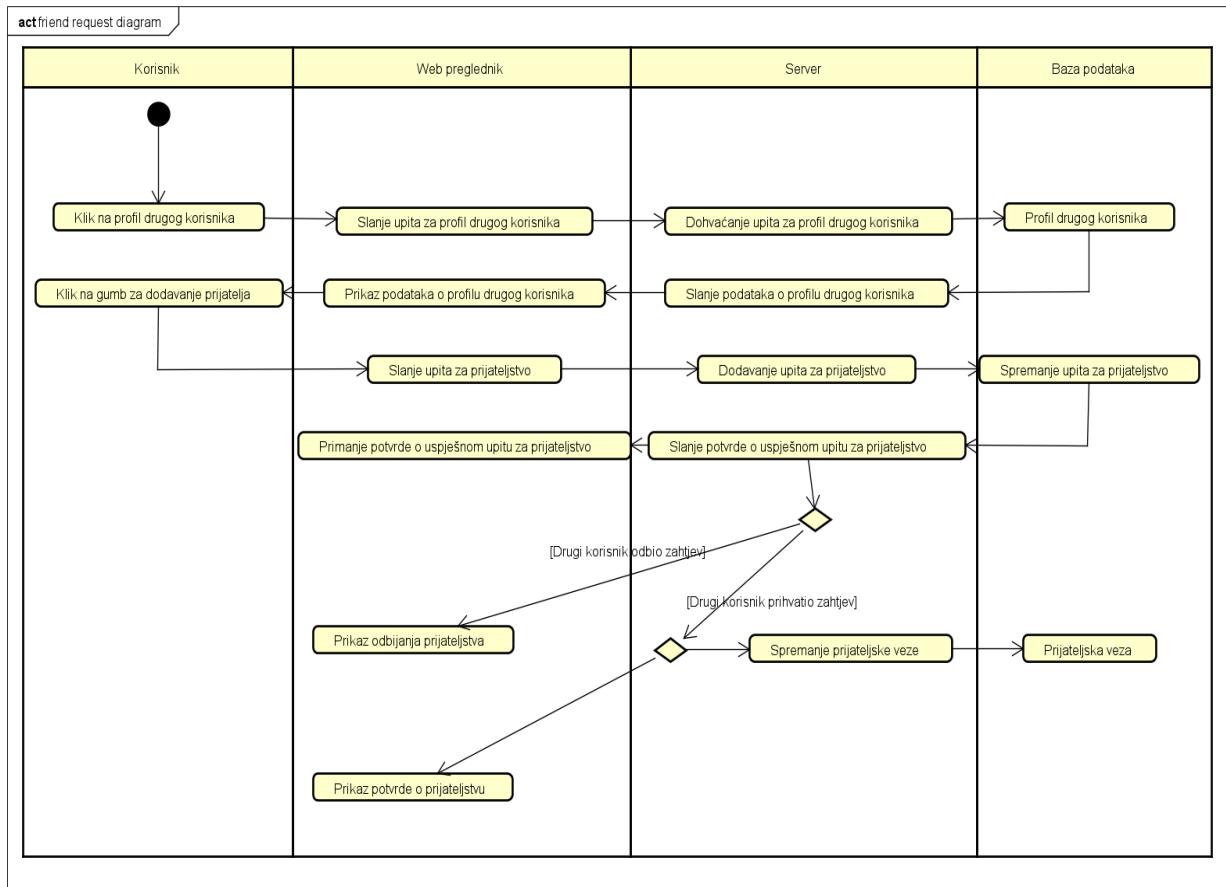
4.4 Dijagram aktivnosti

Dijagram aktivnosti se koristi za modeliranje poslovnih procesa, upravljačkog i podatkovnog toka. Pogodni su za opisivanje sinkronizacije i konkurentnosti. Ne primjenjuju se za modeliranje događajima poticanog ponašanja. Na dijagramu 4.13 prikazan je dijagram aktivnosti pretrage i arhiviranja staze. U našoj aplikaciji, jedna od funkcionalnosti koju dozvoljavamo korisniku je pretraga planinarskih staza po nekim parametrima, prikaz detalja o toj stazi i mogućnost arhiviranja te staze kako bi korisnik mogao čuvati informaciju o stazama koje je prešao. Ako je arhivirao određeni broj staza, dobiva bedž na svome profilu koji to označava.

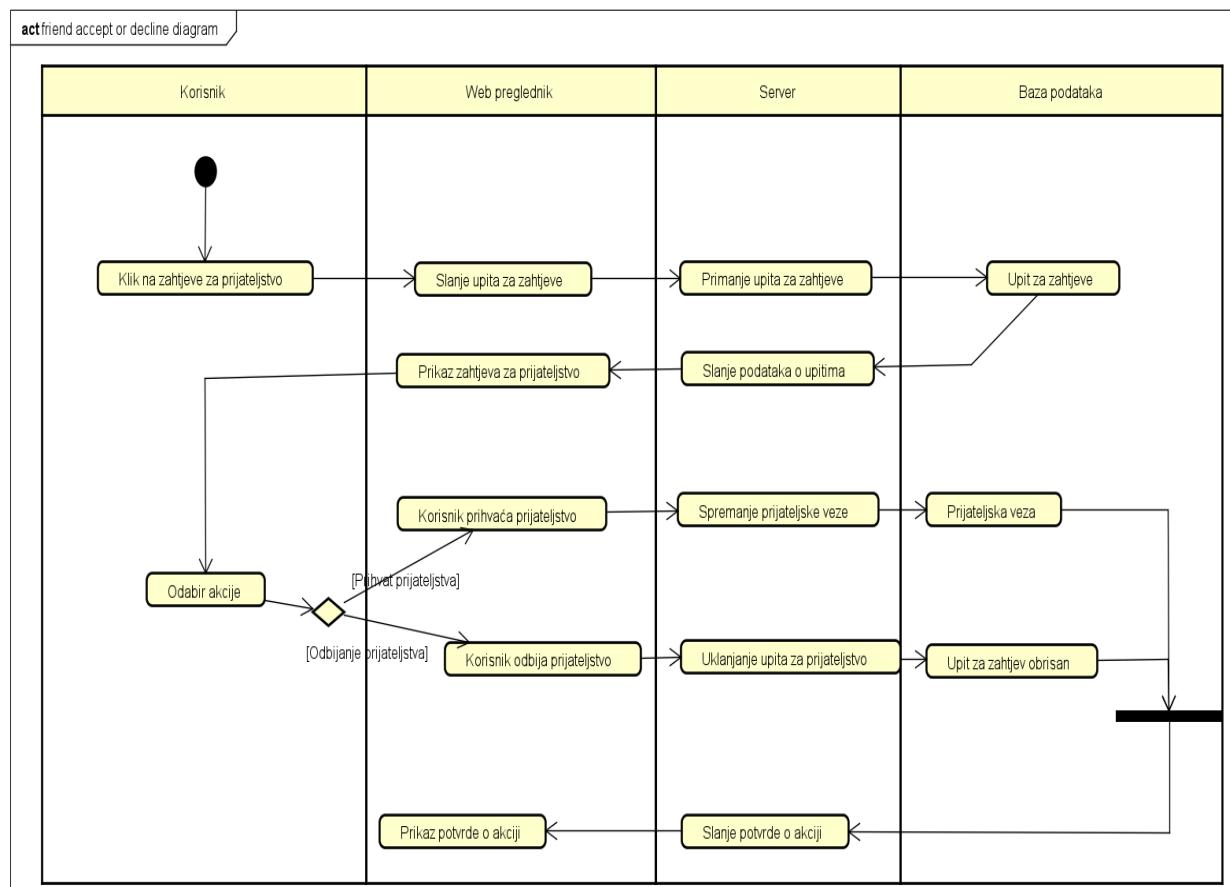


Slika 4.20: Dijagram aktivnosti - pretraga i arhiviranje staze

Još jedna od funkcionalnosti je dodavanje prijatelja. Korisnik može poslati zahtjev za prijateljstvo drugom korisniku. Taj drugi korisnik nakon toga može prihvati ili odbiti zahtjev za prijateljstvo



Slika 4.21: Dijagram aktivnosti - dodavanje prijatelja

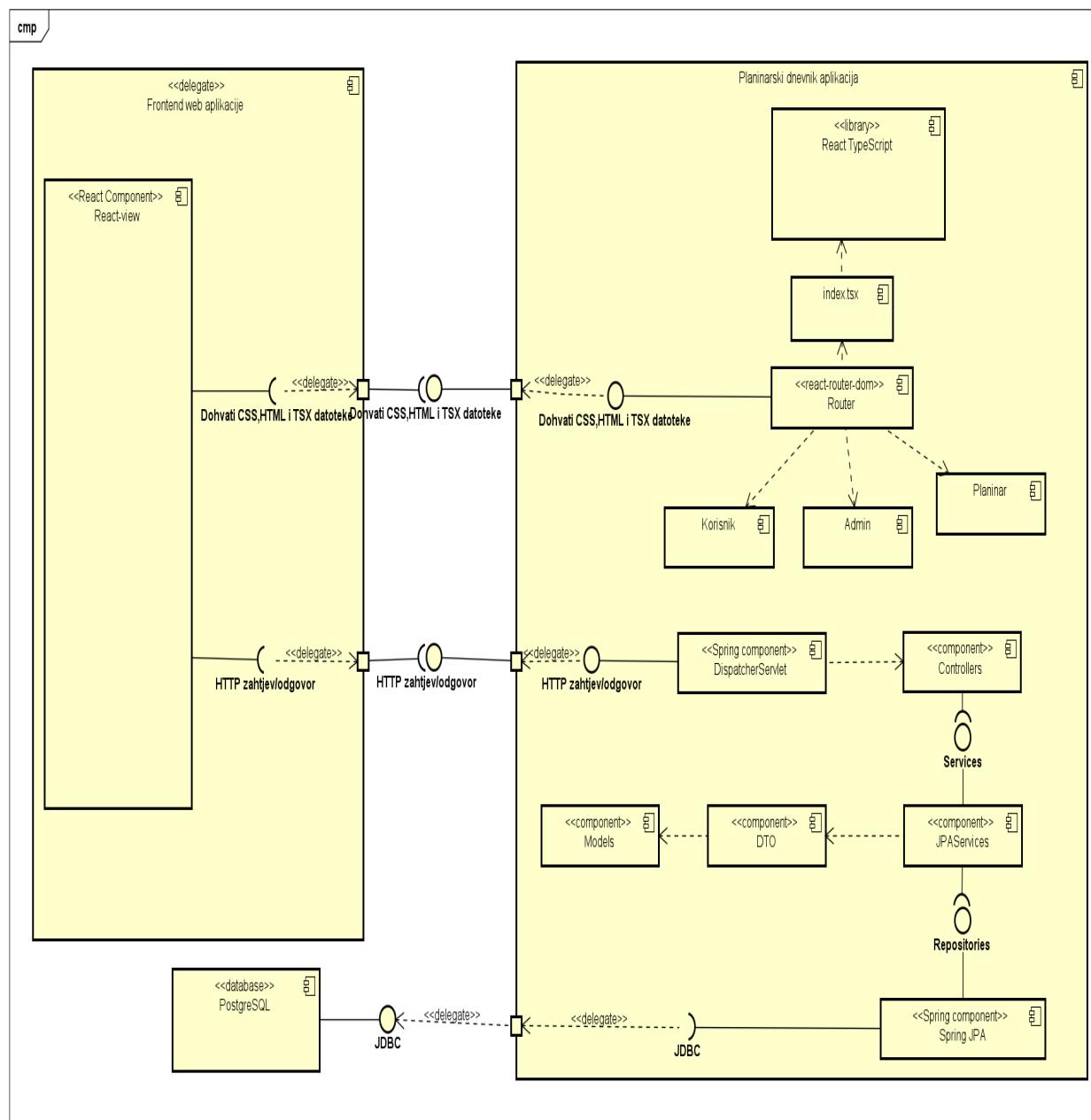


Slika 4.22: Dijagram aktivnosti - prihvaćanje ili odbijanje prijateljstva

4.5 Dijagram komponenti

Dijagrami komponenti prikazuju komponente sustava i njihove međusobne odnose. Komponenta je fizička i stvarna implementacija logičkih elemenata sustava, npr. razreda i sučelja. Često se kaže da su u UML-u sve fizičke "stvari" modelirane kao komponente. Sustavu pristupamo preko dva različita sučelja. Preko jednog dohvaćamo HTML, CSS I TSX datoteke, pomoću kojih poslužujemo datoteke koje pripadaju front end-u aplikacije, a preko drugog sučelja dohvaćamo JSON podatke kojima pristupamo preko REST API komponente, a pomoću nje poslužujemo datoteke koje pripadaju back end-u aplikacije. Frontend sadrži komponentu Router koja pomoći zatraženog url-a na sučelje poslužuje odgovarajuću datoteku. Također sadrži i niz TypeScript datoteka koje su podijeljene u logičke cjeline, a naziv im je dodijeljen po tipu aktora koji pristupaju tim datotekama. React biblioteka je temeljna za sve TypeScript datoteke jer se preko nje dohvaćaju gotove

komponente koje su ključne za prikaz podataka, primjerice gumbi i forme. Za dohvata podataka iz baze podataka pomoću SQL upita zadužen je *Spring JPA* koji u pozadini koristi *JDBC*. Kada se podaci dohvate iz baze, šalju se MVC arhitekturi u obliku DTO. Aplikacija Planinarski dnevnik preko gore navedenih sučelja komunicira sa Reactovim komponentama i ovisno o korisnikovim željama prikazuje i dohvaća zatražene datoteke.



Slika 4.23: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Tijekom rada na projektu najveći dio komunikacije ostvaren je redovnim sastancima putem aplikacije *Microsoft Teams*¹, aplikaciji namijenjenoj za komunikaciju članova tima. Za olakšan timski rad na projektu veliku zaslugu imao je i *Git*², alat za verzioniranje inačica i praćenje promjena, a kao udaljena platforma koristio se *Gitlab*³.

Kao razvojno okruženje koristio se *IntelliJIDEA*⁴, a predstavlja integrirano razvojno okruženje napisano u Javi za razvoj računalnog softvera. Tehnologija na poslužiteljskoj strani bio je *Spring Boot*⁵, radni okvir otvorenog koda korišten za stvaranje mikroservisa, a kod na poslužiteljskoj strani pisan je u programskom jeziku *Java 11*. *Spring Boot* je odabran zato što nudi veliku količinu gotovih funkcionalnosti visoke razine apstrakcije. Uz njega koristimo i poznatu knjižnicu *Spring JPA*⁶ te ORM biblioteku *Hibernate*⁷ koja služi za mapiranje objektno orijentiranog modela domena u relacijsku bazu podataka. Za vrijeme razvoja aplikacije korištena je privremena baza podataka *H2*⁸, a ona predstavlja neperzistentnu bazu podataka napisanu u programskom jeziku *Java*. Kao produkcijska baza korištena je relacijska perzistentna baza podataka *PostgreSQL*⁹, koja je besplatna i otvorenog koda, u *Springu* dobro podržana. Kao alat za praćenje promjena, tzv. migracija baze podataka korišten je *Liquibase*¹⁰. Za testiranje poslužiteljske strane korištena je knjižnica *JUnit*¹¹, namijenjena za testiranje komponenti.

¹<https://www.microsoft.com/hr-hr/microsoft-365/microsoft-teams/>

²<https://git-scm.com/>

³<https://gitlab.com/>

⁴<https://www.jetbrains.com/idea/>

⁵<https://spring.io/projects/spring-boot>

⁶<https://spring.io/projects/spring-data-jpa>

⁷<https://hibernate.org/>

⁸<https://www.h2database.com/>

⁹<https://www.postgresql.org/>

¹⁰<https://www.liquibase.org/>

¹¹<https://junit.org/junit5/>

Što se tiče klijentske strane korišten je *React*¹², a predstavlja *Javascript*¹³ knjižnicu za izgradnju korisničkog sučelja ili UI komponenti. Kod na klijentskoj strani pisan je u programskom jeziku *TypeScript*¹⁴, a to je programski jezik koji je razvio *Microsoft*, te u klasični *Javascript* dodaje navođenje tipova podataka. Prilikom izvršavanja, prevodi se u *Javascript* te se izvodi unutar preglednika. Osim toga korišten je standardni *CSS* i *HTML*, a neke gotove komponente preuzete su iz knjižnice gotovih *React* komponenti, *Material-UI*¹⁵ te *Semantic UI*¹⁶.

Za pisanje dokumentacije korišten je *LaTeX*¹⁷, jezik za pisanje strukturiranih tekstova, a pisan unutar integracijskog radnog okruženja za pisanje *LaTeX* dokumenata *TexStudio*¹⁸. Svi potrebni dijagrami modelirani su pomoću alata *Astah*¹⁹.

¹²<https://reactjs.org/>

¹³<https://www.javascript.com/>

¹⁴<https://www.typescriptlang.org/>

¹⁵<https://material-ui.com/>

¹⁶<https://semantic-ui.com/>

¹⁷<https://www.latex-project.org/>

¹⁸<https://www.texstudio.org/>

¹⁹<https://astah.net/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

U 7 testova proveli smo ispitivanje sloja nadglednika za pretraživanje domova i stvaranje novog planinarskog doma od strane administratora, te 4 testa za pretragu planinarskih domova na servisnom sloju. Kako bismo testirali servisni sloj, napravili smo SQL skriptu koja nam bazu podataka puni testnim podacima. Navedena skripta će biti prikazana prije navođena testova servisnog sloja. Za testiranje sloja nadglednika koristili smo *MockMvc*, odnosno nema stvarne komunikacije s bazom podataka prilikom testiranja sloja nadglednika. Prilikom testiranja neke pomoćne metode nisu navede zbog prevelikog prostora koji zauzima cijeli kod.

Testiranje sloja nadglednika

Ispitni slučaj 1: Pretraga planinarskih domova kada su sva polja null vrijednosti.

Očekivani rezultat ovog testa je povratak svih domova koji se nalaze u bazi podataka, to je podrazumijevano pretraživanje domova.

```
@Test
public void
    GivenRequestAllFieldsNull_When_MountainLodgeSearch_Should_ReturnAllResults()
throws Exception {

    List<MountainLodge> mountainLodges = MountainLodgeGeneratingUtil
        .generateLodgesEmptyUtilitiesAndImageNull(3);
    given(mountainLodgeQueryService
        .findAllMountainLodgeBySearchCriteria(any()))
        .willReturn(mountainLodges);

    mvc.perform(post("/mountain-lodges/search")
        .content("{}")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$", hasSize(3)))
}
```

```

.andExpect(jsonPath("$.hillName", is("hill1")))
.andExpect(jsonPath("$.hillName", is("hill2")))
.andExpect(jsonPath("$.hillName", is("hill3")))
.andExpect(jsonPath("$.name", is("lodge1")))
.andExpect(jsonPath("$.name", is("lodge2")))
.andExpect(jsonPath("$.name", is("lodge3")))
.andExpect(jsonPath("$.id", is(1)))
.andExpect(jsonPath("$.id", is(2)))
.andExpect(jsonPath("$.id", is(3)));
}

```

Ispitni slučaj 2: Pretraga planinarskih domova, polje za pretragu sadrži prazan znakovni niz.

Očekivani rezultat ovog testa je povratak svih domova koji se nalaze u bazi podataka.

```

@Test
public void
    GivenRequestSearchTextEmpty_When_MountainLodgeSearch_Should_ReturnAllResults()
throws Exception {

    List<MountainLodge> mountainLodges = MountainLodgeGeneratingUtil
        .generateLodgesEmptyUtilitiesAndImageNull(3);
    given(mountainLodgeQueryService
        .findAllMountainLodgeBySearchCriteria(any())).willReturn(mountainLodges);

    mvc.perform(post("/mountain-lodges/search")
        .content("{\"searchText\":\"\"}")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$", hasSize(3)))
        .andExpect(jsonPath("$.hillName", is("hill1")))
        .andExpect(jsonPath("$.hillName", is("hill2")))
        .andExpect(jsonPath("$.hillName", is("hill3")))
        .andExpect(jsonPath("$.name", is("lodge1")))
        .andExpect(jsonPath("$.name", is("lodge2")))
        .andExpect(jsonPath("$.name", is("lodge3")))
}

```

```
.andExpect(jsonPath("$.id", is(1)))
.andExpect(jsonPath("$.id", is(2)))
.andExpect(jsonPath("$.id", is(3)));
}
```

Ispitni slučaj 3: Pretraga planinarskih domova, sva polja sadržavaju dozvoljene vrijednosti.

Očekivani rezultat ovog testa je povratak prazne liste domova, jer dom koji zadovoljava parametre ne postoji.

```
@Test
public void
    GivenRequestWithAllFields_When_MountainLodgeSearch_Should_ReturnEmptyList()
throws Exception {

    List<MountainLodge> mountainLodges = Collections.emptyList();
    given(mountainLodgeQueryService
        .findAllMountainLodgeBySearchCriteria(any()))
        .willReturn(mountainLodges);

    mvc.perform(post("/mountain-lodges/search")
        .content(
            "{\"searchText\":\"planinarski dom graficar\",
            \"hillId\":\"1\",
            \"utilities\":[\"1\"]}")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$", hasSize(0)));
}
```

Ispitni slučaj 4: Pretraga planinarskih domova, sva polja sadržavaju dozvoljene vrijednosti.

Očekivani rezultat ovog testa je povratak liste koja sadržava jedan dom, jedini dom koji zadovoljava parametre pretrage.

```
@Test
public void
```

```
GivenRequestWithAllFields_When_MountainLodgeSearch_Should_ReturnOneResultsGoodParam
throws Exception {

List<MountainLodge> mountainLodges = MountainLodgeGeneratingUtil
.generateLodgesEmptyUtilitiesAndImageNull(2);

List<Utility> utilities = getUtilities();

mountainLodges.get(0).setUtilities(utilities);
mountainLodges.get(1).setUtilities(utilities);
mountainLodges.get(1).setHill(mountainLodges.get(0).getHill());

given(mountainLodgeQueryService
.findAllMountainLodgeBySearchCriteria(any())).willReturn(mountainLodges);

mvc.perform(post("/mountain-lodges/search")
.content("{\"searchText\":\"lodge\", \"hillId\":\"1\",
\"utilities\":[\"1\"]}")
.contentType(MediaType.APPLICATION_JSON))
.andExpect(status().isOk())
.andExpect(jsonPath("$.size()", hasSize(2)))
.andExpect(jsonPath("$.[0].hillName", is("hill1")))
.andExpect(jsonPath("$.[1].hillName", is("hill1")))
.andExpect(jsonPath("$.[0].name", is("lodge1")))
.andExpect(jsonPath("$.[1].name", is("lodge2")))
.andExpect(jsonPath("$.[0].id", is(1)))
.andExpect(jsonPath("$.[1].id", is(2)))
.andExpect(jsonPath("$.[0].utilities", hasSize(1)))
.andExpect(jsonPath("$.[1].utilities", hasSize(1)));

}
}
```

Ispitni slučaj 5: Stvaranje planinarskog doma, korisnik nije administrator.
Očekivani rezultat ovog testa je povratak statusa 403, običan korisnik nema ovlasti stvaranja planinarskog doma.

@Test

```
public void
    Given_PrincipalNotAdmin_When_createMountainLodge_ShouldReturnForbiddenStatus()
throws Exception {

    mvc.perform(post("/mountain-lodges/create")
        .contentType(MediaType.APPLICATION_JSON)
        .content("{\"name\":\"Graficar\", \"hillId\":\"1\",
                  \"elevation\":1000}")
        .header("Authorization", NON_ADMIN_AUTH))
        .andExpect(status().isForbidden());

}
```

Ispitni slučaj 6: Stvaranje planinarskog doma, korisnik je admin

Očekivani rezultat ovog testa je uspješno stvaranje novog planinarskog doma.

```
@Test
public void
    Given_PrincipalAdmin_When_createMountainLodge_ShouldSuccess()
throws Exception {

    MountainLodge lodge = getDefaultLodge();
    Principal adminPrincipal = getAdminPrincipal();

    given(userService.getRole(any())).willReturn("ADMIN");
    given(mountainLodgeQueryService.createMountainLodge(any()))
        .willReturn(lodge);

    mvc.perform(post("/mountain-lodges/create")
        .contentType(MediaType.APPLICATION_JSON)
        .content("{\"name\":\"Graficar\", \"hillId\":\"1\",
                  \"elevation\":1000}")
        .principal(adminPrincipal))
        .andExpect(status().isCreated());

}
```

Ispitni slučaj 7: Stvaranje planinarskog doma, korisnik je admin ali su polja pogrešno zadana.

Ovaj test provjerava nemogućnost stvaranja planinarskog doma ukoliko nedostaje parametar koji govori na kojemu se visočju planinarski dom nalazi. Očekivani rezultat ovog testa je status *404*, pogrešan zahtjev.

```
@Test
public void
    Given_RequestWithoutRequiredField_When_createMountainLodge_ShouldReturnBadRequest()
        throws Exception {

    MountainLodge lodge = getDefaultLodge();
    Principal adminPrincipal = getAdminPrincipal();

    given(userService.getRole(any())).willReturn("ADMIN");
    given(mountainLodgeQueryService.createMountainLodge(any())).willReturn(lodge);

    mvc.perform(post("/mountain-lodges/create")
        .contentType(MediaType.APPLICATION_JSON)
        .content("{\"name\":\"Graficar\", \"elevation\":\"1000\"}")
        .principal(adminPrincipal))
        .andExpect(status().isBadRequest());
}

}
```

Testiranje sloja servisa

U nastavku je prikazana prije navedena SQL skripta, kao i testovi sloja servisa.

```
INSERT INTO HILL(ID, NAME) VALUES(1000, 'Blidinje');
INSERT INTO HILL(ID, NAME) VALUES(1001, 'Velez');
INSERT INTO HILL(ID, NAME) VALUES(1002, 'Cvrsnica');

INSERT INTO UTILITY(UTILITY_ID, NAME) VALUES(1000, 'Voda');
INSERT INTO UTILITY(UTILITY_ID, NAME) VALUES(1001, 'Hrana');
INSERT INTO UTILITY(UTILITY_ID, NAME) VALUES(1002, 'Smjestaj');
INSERT INTO UTILITY(UTILITY_ID, NAME) VALUES(1003, 'Spavanje');

INSERT INTO MOUNTAIN_LODGE(LODGE_ID, NAME, IMAGE, HILL_ID, elevation) VALUES(1000, 'Hajducke vrleti', '', 1000, 1500);
INSERT INTO MOUNTAIN_LODGE(LODGE_ID, NAME, IMAGE, HILL_ID, elevation) VALUES(1001, 'Veleske vrleti', '', 1001, 1600);
INSERT INTO MOUNTAIN_LODGE(LODGE_ID, NAME, IMAGE, HILL_ID, elevation) VALUES(1002, 'Cvrsti san', '', 1002, 1700);
INSERT INTO MOUNTAIN_LODGE(LODGE_ID, NAME, IMAGE, HILL_ID, elevation) VALUES(1003, 'Cvrsci san', '', 1002, 1800);

INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1000, 1000);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1000, 1001);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1000, 1002);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1000, 1003);

INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1001, 1000);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1001, 1001);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1002, 1002);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1002, 1003);

INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1003, 1000);
INSERT INTO MOUNTAIN_LODGE_UTILITY(LODGE_ID, UTILITY_ID) VALUES(1003, 1001);
```

Slika 5.1: SQL skripta za testiranje servisnog sloja

Ispitni slučaj 8: Pretraga planinarskih domova prema nazivu.

Pretražujemo sve planinarske domove koji sadržavaju slovo "v", a infrastrukturne pogodnosti su voda i hrana, predstavljene svojim identifikatorima 1000 odnosno 1001. Očekivano ponašanje je povratak 3 planinarska doma.

```
@Test
public void
    Given_ValidRequestWithUtilities_When_SearchMountainLodgesBySearchCriteria_
Should_ReturnResultsFromDb(){

    MountainLodgeSearchRequest request = new
        MountainLodgeSearchRequest();

    request.setSearchText("v");
    request.setUtilities(List.of(1000L, 1001L));

    List<MountainLodge> response =
        mountainLodgeQueryService.findAllMountainLodgeBySearchCriteria(request);

    Assert.assertEquals(3, response.size());
    Assert.assertEquals("Hajducke vrleti", response.get(0).getName());
    Assert.assertEquals("Veleske vrleti", response.get(1).getName());
    Assert.assertEquals("Cvrsci san", response.get(2).getName());

    Assert.assertEquals(Long.valueOf(1000), response.get(0).getId());
    Assert.assertEquals(Long.valueOf(1001), response.get(1).getId());
    Assert.assertEquals(Long.valueOf(1003), response.get(2).getId());
}
```

Ispitni slučaj 9: Pretraga planinarskih domova, sva polja sadržavaju dozvoljene vrijednosti.

Očekivani rezultat ovog testa je povratak liste planinarskih domova, čije ime sadržava riječ "vrleti", a nalaze se na planini "Blidinje", identifikatora 1000.

```
@Test
public void
    Given_ValidRequest_When_SearchMountainLodgesBySearchCriteria
```

```

_should_ReturnResultsFromDb(){

    MountainLodgeSearchRequest request = new
        MountainLodgeSearchRequest();
    request.setHillId(1000L);
    request.setSearchText("vrleti");
    request.setUtilities(Collections.emptyList());

    List<MountainLodge> response =
        mountainLodgeQueryService.findAllMountainLodgeBySearchCriteria(request);

    Assert.assertEquals(1, response.size());
    Assert.assertEquals("Hajducke vrleti", response.get(0).getName());
    Assert.assertEquals(Long.valueOf(1000), response.get(0).getId());
    Assert.assertEquals(4, response.get(0).getUtilities().size());

}

```

Ispitni slučaj 10: Pretraga planinarskih domova, sva polja sadržavaju dozvoljene vrijednosti, nema rezultata.

Očekivani rezultat ovog testa je povratak prazne liste domova, jer dom koji zadovoljava parametre ne postoji u bazi podataka. Parametar koji nije zadovoljen je naziv planinarskog doma "Planinarski dom graficar".

```

@Test
public void
    Given_ValidRequestNoMatchingLodgeName_When_SearchMountainLodgesBySearchCriteria_
Should_ReturnEmptyList(){

    MountainLodgeSearchRequest request = new
        MountainLodgeSearchRequest();
    request.setSearchText("Planinarski dom graficar");
    request.setUtilities(List.of(1000L, 1001L));

    List<MountainLodge> response = mountainLodgeQueryService
        .findAllMountainLodgeBySearchCriteria(request);
    Assert.assertEquals(0, response.size());
}

```

 }

Ispitni slučaj 11: Pretraga planinarskih domova, tekst pretrage velikim slovima.

Očekivani rezultat ovog testa je povratak liste koja sadrži jedan dom naziva "Hajducke vrleti".

```

@Test
public void
    Given_ValidRequestUppercase_When_SearchMountainLodgesBySearchCriteria
    _Should_ReturnOneResult(){

    MountainLodgeSearchRequest request = new
        MountainLodgeSearchRequest();
    request.setSearchText("HAJDUCKE vR1ETI");

    List<MountainLodge> response =
        mountainLodgeQueryService.findAllMountainLodgeBySearchCriteria(request);
    Assert.assertEquals(1, response.size());
    Assert.assertEquals("Hajducke vrleti", response.get(0).getName());
}

```

Prikaz rezultata testova

✓	hr.fer.pi.planinarskidnevnik.controllers.MountainLodgeControllerTest	544 ms
✓	GivenRequestWithAllFields_When_MountainLodgeSearch_Should_ReturnOneResultsGoodParams()	343 ms
✓	Given_RequestWithoutRequiredField_When_createMountainLodge_ShouldReturnBadRequest()	131 ms
✓	Given_PrincipalAdmin_When_createMountainLodge_ShouldSuccess()	18 ms
✓	GivenRequestSearchTextEmpty_When_MountainLodgeSearch_Should_ReturnAllResults()	16 ms
✓	GivenRequestAllFieldsNull_When_MountainLodgeSearch_Should_ReturnAllResults()	12 ms
✓	GivenRequestWithAllFields_When_MountainLodgeSearch_Should_ReturnEmptyList()	12 ms
✓	Given_PrincipalNotAdmin_When_createMountainLodge_ShouldReturnForbiddenStatus()	12 ms
✓	hr.fer.pi.planinarskidnevnik.services.MountainLodgeQueryServiceTest	481 ms
✓	Given_ValidRequestUppercase_When_SearchMountainLodgesBySearchCriteria_Should_ReturnOneResult()	393 ms
✓	Given_ValidRequestWithUtilities_When_SearchMountainLodgesBySearchCriteria_Should_ReturnResultsFromDb()	42 ms
✓	Given_ValidRequest_When_SearchMountainLodgesBySearchCriteria_Should_ReturnResultsFromDb()	28 ms
✓	Given_ValidRequestNoMatchingLodgeName_When_SearchMountainLodgesBySearchCriteria_Should_ReturnEmptyList()	18 ms

Slika 5.2: Rezultati JUnit testova

5.2.2 Ispitivanje sustava

Ispitivanje smo proveli koristeći Selenium WebDriver unutar JUnit testova. Cilj ispitivanje bio je provjera nekih glavnih funkcionalnosti sustava i pronalazak pogrešaka. WebDriver potrebno je instalirati lokalno na računalo i prilikom pokretanja postaviti putanju koja se nalazi u varijablama okruženja sustava. Prilikom testiranja aplikacija se pokretala na lokalnom računalu zato što je Heroku servis na kojem je aplikacija puštena u pogon dosta sporiji, pa bi testovi trajali nešto dulje vremena. Slika testova koji su se uspješno izvršili i na aplikaciji puštenoj u pogon na servisu Heroku će biti priložena.

Ispitni slučaj 1: Prijava planinara u sustav s ispravnim/neispravnim podacima

```
public void testLoginBadAndGoodCreds() throws InterruptedException {
    //Driver setup...
    System.setProperty("webdriver.chrome.driver", "/path...");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(Duration.of(20,
        ChronoUnit.SECONDS));

    driver.get("http://localhost:3000/home");
    WebElement loginButton = driver.findElement(By.id("log-button"));
    loginButton.click();
    String loginUrl = driver.getCurrentUrl();
    assertTrue(loginUrl.contains("/login"));
    WebElement Emailelement = driver.findElement(By.id("email"));
    Emailelement.sendKeys("luka.ravenscak");
    WebElement passElement = driver.findElement(By.id("password"));
    passElement.sendKeys("password");
    driver.findElement(By.className("submitButton")).click();
    WebElement element = driver.findElement(By.className("errorText"));
    assertEquals("Neispravan oblik mail-a.", element.getText());
    Emailelement.clear();
    Emailelement.sendKeys("luka.ravenscak@fer.hr");
    passElement.clear();
    passElement.sendKeys("mypassword");
    driver.findElement(By.className("submitButton")).click();
```

```
element = driver.findElement(By.id("error-span"));
assertEquals("Neispravan e-mail ili lozinka.", element.getText());
passElement.clear();
passElement.sendKeys("password");
driver.findElement(By.className("submitButton")).click();
Thread.sleep(1000);
assertTrue(driver.getCurrentUrl().contains("/mountaineering-community"));

driver.quit();
}
```

U 1. ispitnom slučaju ispitana je funkcionalnost prijave u sustav. Prvo se dohvati početna stranica aplikacije te se odabire opcija "Prijavi se" koja se nalazi na zaglavlju stranice. Zatim se unese neispravni oblik email-a, na što aplikacija vraća obavijest "Neispravan oblik mail-a.", nakon toga se unose valjni email i neispravna lozinka, aplikacija reagira tako da vrati obavijest "Neispravan e-mail ili lozinka.". Na kraju se unesu ispravni podaci za prijavu i aplikacija reagira uspješnom prijavnom korisnika u sustav i odvede korisnika na stranicu vlastite planinarske zajednice.

Ispitni slučaj 2: Planinar pretražuje ostale korisnike

```
public void testSearchAdminOnAllUserSearchPage() throws
InterruptedException {

    //Driver setup
    driver.get("http://localhost:3000/login");

    WebElement element = driver.findElement(By.id("email"));
    element.sendKeys("luka.ravenscak@fer.hr");
    element = driver.findElement(By.id("password"));
    element.sendKeys("password");
    driver.findElement(By.className("submitButton")).click();
    Thread.sleep(2000);
    assertTrue(driver.getCurrentUrl().contains("/mountaineering-community"));

    WebElement searchAllButton =
        driver.findElement(By.className("search-all-button"));
```

```
searchAllButton.click();

assertTrue(driver.getCurrentUrl().contains("/users/search"));
element = driver.findElement(By.name("searchText"));
element.sendKeys("admin");
driver.findElement(By.name("searchText")).click();
driver.findElement(By.className("all-user-photo")).click();
Thread.sleep(2000);
assertTrue(driver.getCurrentUrl().contains("/profile"));

assertEquals("admin",
    driver.findElement(By.id("name-profile")).getAttribute("value"));
driver.quit();
}
```

U 2. ispitnom slučaju ispitana je funkcionalnost pretrage svih korisnika od strane planinara. Prvo se provede prijava planinara u sustav, zatim prijavljeni korisnik odabire opciju "Pretraži sve planinare" te ga aplikacija preusmjeri na stranicu za pretragu svih korisnika. Korisnik u prostor za unos teksta za pretragu upiše "admin" i odabire opciju pretrage, aplikacija u rezultatima pretrage vrati traženog korisnika. Nakon toga korisnik odabere adminovu sličicu profila i aplikacija mu prikaže adminov profil. Testiranje je provedeno na način da se pretražuje profil admina, koji unutar aplikacije uvijek postoji.

Ispitni slučaj 3: Pretraga i arhiviranje posjećenog planinarskog doma

```
public void testMountainLodgeSearchArchiveAndBadgeAdd() throws
InterruptedException {
//Driver setup

driver.get("http://localhost:3000");
driver.findElement(By.className("domovi")).click();
assertTrue(driver.getCurrentUrl().contains("/mountain-lodge/search"));
driver.findElement(By.className("input-search")).sendKeys("Planinarski
dom Glavica");
driver.findElement(By.className("search-button")).click();
assertEquals("Planinarski dom Glavica",
    driver.findElement(By.className("mountain-lodge-name")).getText());
```

```
assertThrows(NoSuchElementException.class, new ThrowingRunnable() {
    public void run() throws Throwable {
        driver.findElement(By.className("archive-button-lodge"));
    }
});

WebElement loginButton = driver.findElement(By.id("log-button"));
loginButton.click();
String loginUrl = driver.getCurrentUrl();
assertTrue(loginUrl.contains("/login"));
WebElement Emailelement = driver.findElement(By.id("email"));
Emailelement.sendKeys("luka.ravenscak@fer.hr");
WebElement passElement = driver.findElement(By.id("password"));
passElement.sendKeys("password");
driver.findElement(By.className("submitButton")).click();
Thread.sleep(2000);
assertTrue(driver.getCurrentUrl().contains("/mountaineering-community"));
driver.findElement(By.className("logo-image")).click();

driver.findElement(By.className("domovi")).click();
assertTrue(driver.getCurrentUrl().contains("/mountain-lodge/search"));
driver.findElement(By.className("input-search")).sendKeys("Planinarski
    dom Glavica");
driver.findElement(By.className("search-button")).click();
assertEquals("Planinarski dom Glavica",
    driver.findElement(By.className("mountain-lodge-name")).getText());

WebElement archiveButton =
    driver.findElement(By.className("archive-button-lodge"));
assertEquals("ARHIVIRAJ",driver.findElement(By.className("MuiButton-label"))
.getText());
archiveButton.click();
Thread.sleep(3000);
assertEquals("ARHIVIRANO",driver.findElement(By.className("MuiButton-label"))
.getText());

driver.findElement(By.className("profil-image")).click();
driver.findElement(By.id("my-profile")).click();
```

```
String badgeDesc =  
    driver.findElement(By.className("badge-image")).getAttribute("title");  
assertEquals("Imam barem 1 posjecen planinarski dom!", badgeDesc);  
  
driver.quit();  
}
```

U 3. ispitnom slučaju ispitana je funkcionalnost pretraživanja i arhiviranja planinarskog doma, te je testirano da pristup arhiviranju imaju samo prijavljeni korisnici. Neprijavljen korisnik odabire opciju "Pretraga planinarskih domova", u prostor za unos teksta upisuje "Planinarski dom Glavica". Taj dom postoji u bazi podataka i aplikacija mu kao rezultat pretrage prikaže traženi planinarski dom. Zatim se testira da aplikacija neprijavljenom korisniku ne nudi mogućnost arhiviranja doma. Nakon što je taj dio testa aplikacija zadovoljen, test se nastavlja i korisnik se prijavi u sustav kao planinar te ponavlja istu radnju. Ovaj put aplikacija mu prikazuje traženi dom, ali korisnik ima mogućnost arhiviranja doma. Korisnik odabire opciju "Arhiviraj", aplikacija na tu akciju mijenja opis uz navedeni dom u "Arhivirano", te odabrani dom sprema u popis posjećenih domova planinara koji je odabrao opciju "Arhiviraj". Korisnik odabire opciju "Moj profil", aplikacija mu prikazuje profil na kojem se može vidjeti priznanje (bedž) kojeg je dobio posjetom jednog planinarskog doma, uz opis postignuća "Imam barem 1 posjećen planinarski dom!"

Ispitni slučaj 4: Slanje zahtjeva za prijateljstvo i primetak obavijesti o prihvaćenom zahtjevu

```
public void  
testAddFriendAndFriendRequestApproveOrDeclineAndFriendNotification()  
throws InterruptedException {  
  
//Driver setup  
  
driver.get("http://localhost:3000/login");  
driver.findElement(By.id("email")).sendKeys("luka.ravenscak@fer.hr");  
driver.findElement(By.id("password")).sendKeys("password");  
driver.findElement(By.className("submitButton")).click();  
Thread.sleep(2000);
```

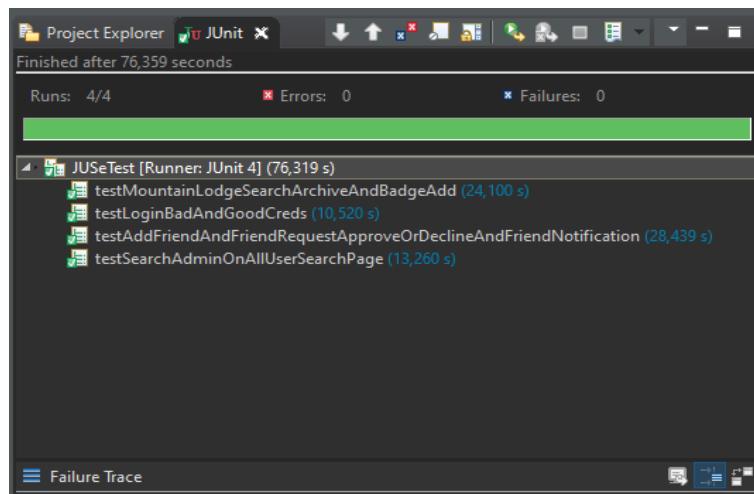
```
WebElement searchAllButton =
    driver.findElement(By.className("search-all-button"));
searchAllButton.click();
assertTrue(driver.getCurrentUrl().contains("/users/search"));
WebElement element = driver.findElement(By.name("searchText"));
element.sendKeys("admin");
driver.findElement(By.name("searchText")).click();
driver.findElement(By.className("all-user-photo")).click();
Thread.sleep(2000);
assertTrue(driver.getCurrentUrl().contains("/profile"));

driver.findElement(By.className("button-profile")).click();
assertTrue(driver.findElement(By.className("button-profile-fr"))
.getText().contains("Zahtjev poslan"));
driver.findElement(By.className("profil-image")).click();
driver.findElement(By.id("logout-b")).click();
Thread.sleep(2000);
driver.findElement(By.id("log-button")).click();
driver.findElement(By.id("email")).sendKeys("admin@fer.hr");
driver.findElement(By.id("password")).sendKeys("password");
driver.findElement(By.className("submitButton")).click();
Thread.sleep(2000);
driver.findElement(By.className("profil-image")).click();
driver.findElement(By.id("fr-requests")).click();
assertEquals("Luka Ravenscak",
    driver.findElement(By.className("user-name")).getText());
driver.findElement(By.className("submitButtonaccept")).click();

driver.findElement(By.className("profil-image")).click();
driver.findElement(By.id("logout-b")).click();
Thread.sleep(2000);
driver.findElement(By.id("log-button")).click();
driver.findElement(By.id("email")).sendKeys("luka.ravenscak@fer.hr");
driver.findElement(By.id("password")).sendKeys("password");
driver.findElement(By.className("submitButton")).click();
Thread.sleep(2000);
driver.findElement(By.className("profil-image")).click();
driver.findElement(By.id("fr-notifications")).click();
```

```
assertTrue(driver.findElement(By.className("user-name-span")))
    .getText().contains("Postali ste prijatelj s"));
assertTrue(driver.findElement(By.className("user-name")))
    .getText().contains("admin"));
driver.quit();
}
```

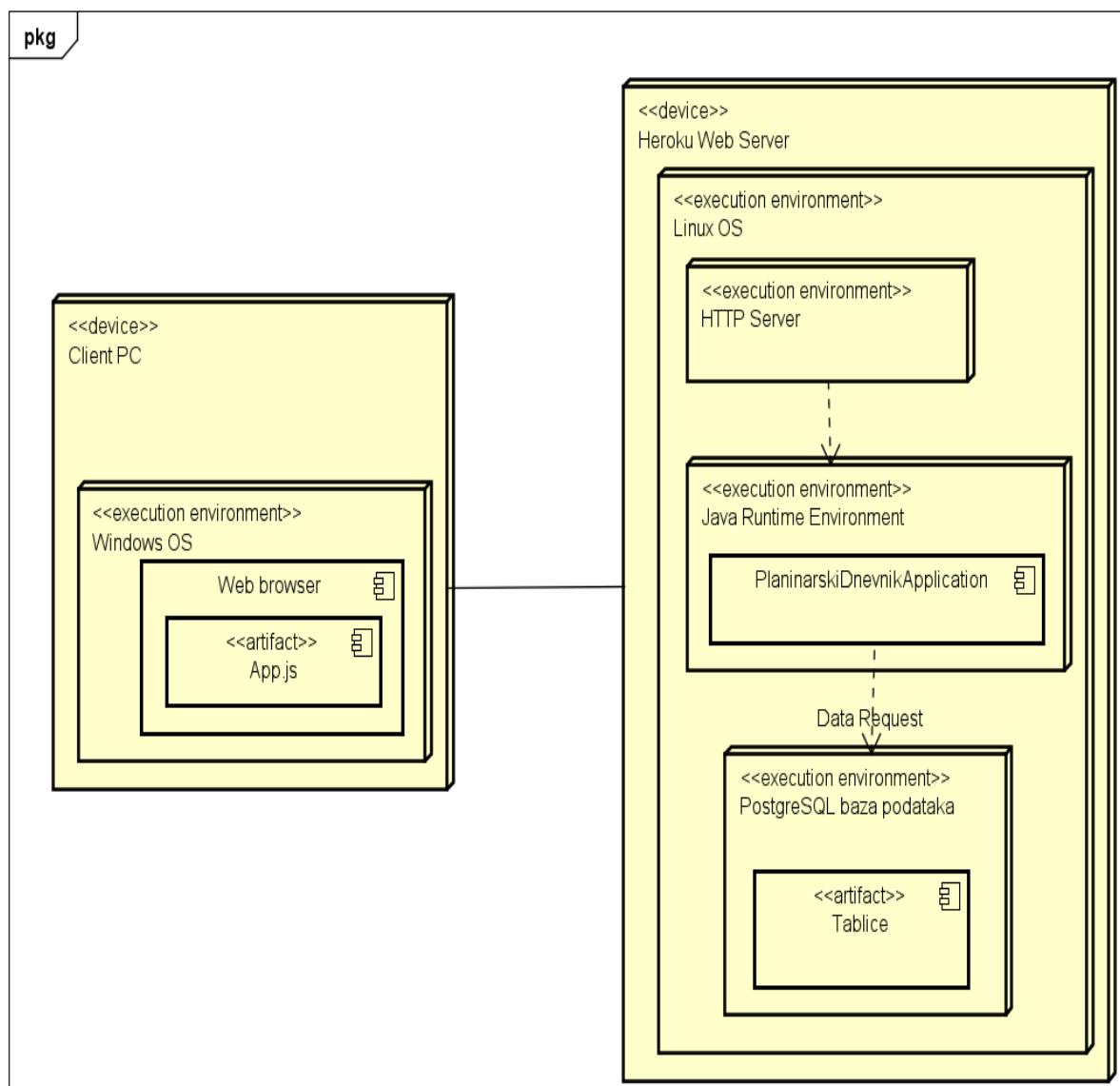
U 4. ispitnom slučaju ispitana je funkcionalnost slanja zahtjeva za prijateljstvo i primanje obavijesti o prihvaćenom zahtjevu. Prvo se korisnik prijavi kao "luka.ravenscak@fer.hr" te provede radnje kao u testu 2. Dolaskom na adminov profil odabere opciju "Dodaj prijatelja". Aplikacija promjeni status gumba u "Zah-tjev poslan" te proslijedi zahtjev za prijateljstvo adminu. Zatim se korisnik odjavi, te odlazimo na profil admina, nakon čega korisnik provjeri zahtjeve za prijateljstvo i prihvati zahtjev korisnika "luka.ravenscak@fer.hr". Nakon što je admin uspješno prihvatio zahtjev korisnik se odjavi i prijavi ponovno kao "luka.ravenscak@fer.hr". Nakon toga aplikacija će korisniku "luka.ravenscak@fer.hr" prikazati novu obavijest o prihvaćenom zahtjevu za prijateljstvo od strane admina.



Slika 5.3: Uspješno izvršeni Selenium testovi - lokalno računalo

5.3 Dijagram razmještaja

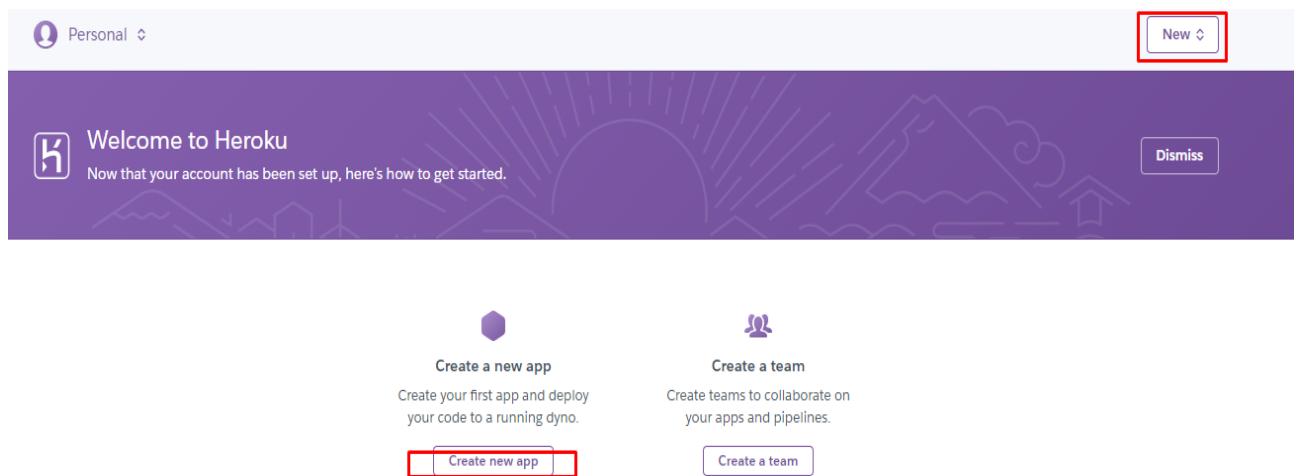
Dijagrami razmještaja opisuju topologiju fizikalnih uređaja i radnih okruženja koja se koriste u implementaciji sustava. Na korisničkom računalu se nalazi web poslužitelj kao pristupna točka aplikacije. Na samom poslužitelju imamo našu PostgreSQL bazu podataka i backend koji služi za razmjenu HTTP zahtjeva s korisnikom i obrađivanje transakcija s bazom podataka. Sustav je baziran na arhitekturi "klijent - poslužitelj".



Slika 5.4: Dijagram razmještaja

5.4 Upute za puštanje u pogon

U ovom poglavlju će biti opisano puštanje aplikacije u pogon, odnosno jedan od načina da se od izvornog koda dođe do funkcionalne aplikacije koja odgovara na upite. Aplikacija je u pogon puštena na servis Heroku²⁰ (platforma kao usluga). Prije početka, potrebno je napraviti korisnički račun na toj platformi, pritiskom na gumb **Sign up for free**, a nakon toga i prijaviti²¹ na platformu. Nakon prijave otvara se ekran koji prikazuje popis Vaših aplikacija te mogućnost stvaranja nove aplikacije, kao na slici. Ako ste se prvi put registrirali, ovaj popis će biti prazan.

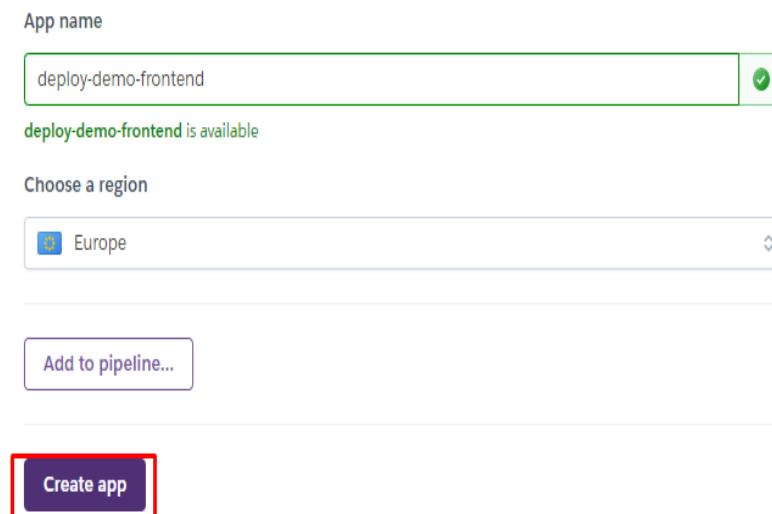


Slika 5.5: Heroku dashboard

²⁰<https://www.heroku.com/home>

²¹<https://id.heroku.com/login>

Pritiskom na gumb za stvaranje nove aplikacije, otvara se prozor kao na slici, gdje unosite željeni naziv aplikacije, u našem slučaju *deploy-demo-frontend*.



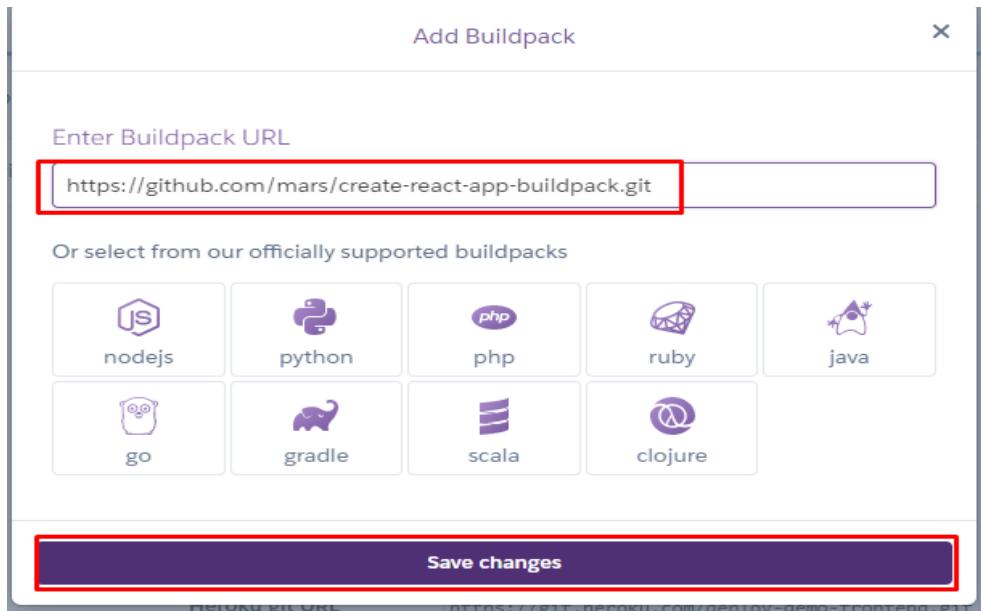
Slika 5.6: Stvaranje frontenda - Heroku

Nakon toga se otvara upravljač za Vašu aplikaciju, odaberite **Settings** te nakon toga gumb **Add buildpack**, kao na slici.

The screenshot shows the Heroku application settings interface. At the top, there's a navigation bar with a user icon, 'Personal', a back arrow, the app name 'deploy-demo-frontend', a star icon, 'Open app' button, and a 'More' dropdown. Below the navigation is a horizontal menu with tabs: Overview, Resources, Deploy, Metrics, Activity, Access, and Settings (which is underlined). The main content area is titled 'App Information'. It includes fields for 'App Name' (set to 'deploy-demo-frontend'), 'Region' (Europe), 'Stack' (heroku-18, with a red-bordered 'Upgrade Stack' button), 'Framework' (No framework detected), 'Slug size' (No slug detected), and 'Heroku git URL' (a link to <https://git.heroku.com/deploy-demo-frontend.git>). Below this is a section titled 'Config Vars' with a 'Reveal Config Vars' button. A note says: 'Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.' At the bottom is a section titled 'Buildpacks' with a large empty box and a red-bordered 'Add buildpack' button.

Slika 5.7: Dodavanje buildpacka

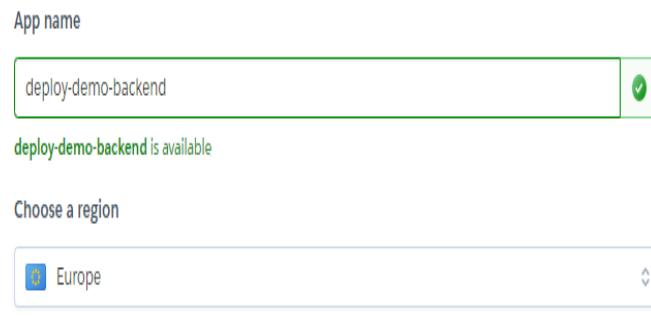
Nakon otvaranja novog prozora, upište sljedeći URL: <https://github.com/mars/create-react-app-buildpack.git>, kao na slici.



Slika 5.8: Dodavanje buildpacka

Na istom mjestu možete pronaći i domenu Vaše aplikacije, ona će uvijek biti u obliku <https://APP-NAME.herokuapp.com/>, gdje je APP-NAME Vaš odabrani naziv aplikacije. U našem slučaju to bi bilo: <https://deploy-demo-frontend.herokuapp.com/>. Na analogan način treba postupiti i s dodavanjem poslužiteljskog dijela aplikacije, samo ovoga puta nakon stvaranja aplikacije **nije potrebno dodavati buildpack** zato što je poslužiteljski dio aplikacije pisan u Javi, s *Gradle-om* kao upravljačem.

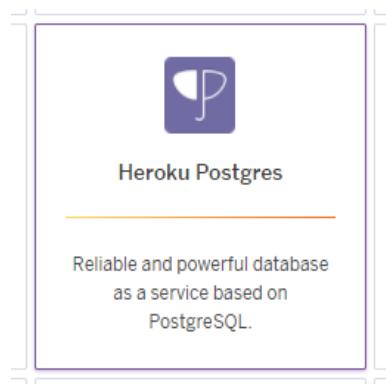
Za dodavanje PostgreSQL baze podataka na backendu, nakon stvaranja aplikacije pratite niže navedene korake. Prilikom odabira za koju aplikaciju stvarate bazu podataka, odaberite poslužiteljsku stranu, u našem slučaju *deploy-demo-backend*.



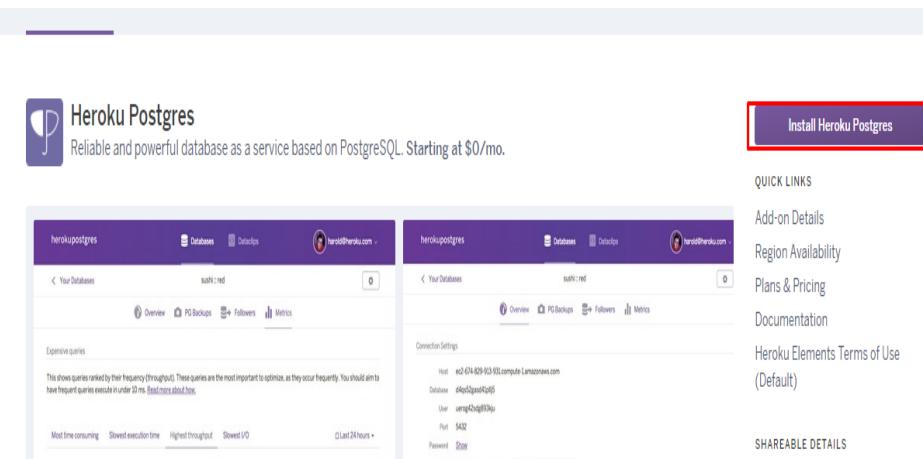
Slika 5.9: Dodavanje backenda

A screenshot of the Heroku dashboard showing the 'Add-ons' section. The top navigation bar includes 'Personal', the app name 'deploy-demo-backend', 'Open app', and 'More'. Below the navigation, there are tabs for 'Overview', 'Resources', 'Deploy', 'Metrics', 'Activity', 'Access', and 'Settings'. The 'Add-ons' section is highlighted with a red box around the 'Find more add-ons' button. A search bar below it says 'Q Quickly add add-ons from Elements'.

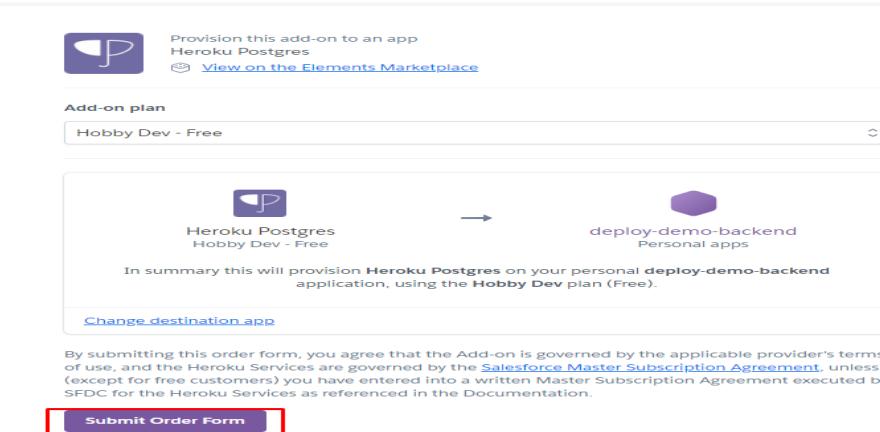
Slika 5.10: Dodavanje baze podataka prvi korak



Slika 5.11: Dodavanje baze podataka drugi korak



Slika 5.12: Dodavanje baze podataka treći korak



Slika 5.13: Dodavanje baze podataka četvrti korak

Sada je sve spremno za postavljanje našeg izvornog koda na udaljeni poslužitelj koji pruža *Heroku*.

Nakon stvaranja aplikacija na Heroku i uspostavljanja baze, sa URL-a ²² potrebno je preuzeti naš Izvorni kod, preporučljivo kao ZIP arhivu zbog monorepozitorija, te ju raspakirati na lokalnom računalu. Nakon toga potrebno je preuzeti *HerokuCLI*²³. Nakon što ste preuzeli izvorni kod, pozicionirajte se u direktorij *IzvorniKod* u Vašem terminalu. Nakon toga unesite naredbu

```
heroku login
```

te se prijavite na Vaš Heroku korisnički račun.

²²<https://gitlab.com/Pi-FER/RuntimeTerror/-/tree/final-deploy/IzvorniKod>

²³<https://devcenter.heroku.com/articles/heroku-cli>

```
D:\Downloads2\RuntimeTerror-final-deploy-IzvorniKod\RuntimeTerror-final-deploy-IzvorniKod\IzvorniKod>heroku login
» Warning: heroku update available from 7.47.0 to 7.47.7.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.herokuapp.com/auth/cli/browser/f1d67953-45e8-4dea-8282-484cc31e7969?requestor=SFMyNT
guMjE0bgYA_V7V-3YBYgABUYA.7-RbAv7SCKEVZgQ1KYy0oZG9Da2isp_Xrvxp7E9mmh8
Logging in... done
```

Slika 5.14: Dodavanje baze podataka četvrti korak

Nakon toga je potrebno otvoriti:

```
IzvorniKod\frontend\src\Util.ts
```

te zamijeniti postojeći varijablu **PROD_ENV** URL-om Vašeg poslužiteljskog dijela aplikacije. Na slici je prikazan URL u ovom demonstracijskom slučaju.

```
const PROD_ENV = "https://deploy-demo-backend.herokuapp.com";
```

Nakon ovoga klijentska strana aplikacije spremna je za puštanje u pogon. Isti postupak ćemo napraviti i na poslužiteljskoj strani, otvorite datoteku:

```
IzvorniKod\backend\src\hr\fer\src\main\java\pi\planinarskidnevnik
\PlaninarskiDnevnikApplication.java
```

te u dopuštene izvore dodajte URL Vaše klijentske strane aplikacije. U našem slučaju to bi izgledalo ovako:

```
allowedOrigins("https://deploy-demo-frontend.herokuapp.com")
```

Nakon toga se pozicionirajte u *IzvorniKod* te unesite redom naredbe:

```
cd backend
git init
heroku git:remote -a deploy-demo-backend

git add .
git commit -am "initial commit"
git push heroku master
```

gdje ćete umjesto **deploy-demo-backend** unijeti ime Vašeg poslužiteljskog dijela aplikacije. Nakon nekoliko minuta će proces unutar naredbenog retka završiti porukom uspjeha te je poslužiteljski dio aplikacije pušten u pogon.

Preostalo je još u pogon pustiti klijentski dio aplikacije. Unesite sljedeće naredbe:

```
cd ../frontend
git init
heroku git:remote -a deploy-demo-frontend

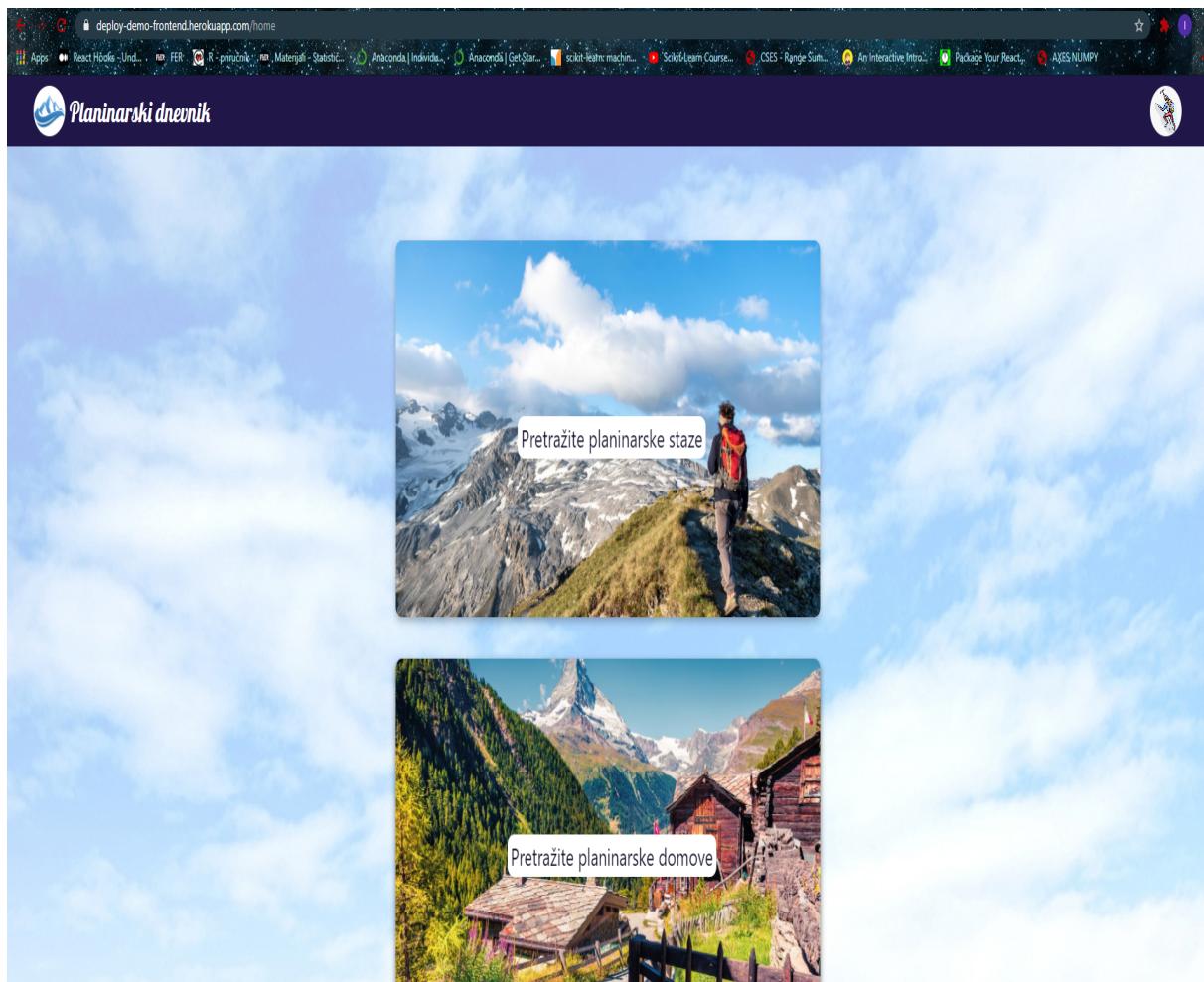
git add .
git commit -am "initial commit"
git push heroku master
```

gdje ćete umjesto **deploy-demo-frontend** unijeti ime Vašeg klijentskog dijela aplikacije. Ovaj proces će potrajati nekoliko minuta duže od poslužiteljskog dijela aplikacije. Naravno, pretpostavka je da imate instaliran *Git*²⁴. Nakon završetka, aplikacija za demonstraciju puštanja u pogon je dostupna na URL-u.²⁵ Stvarna aplikacija dostupna je na URL-u.²⁶.

²⁴<https://git-scm.com/>

²⁵<https://deploy-demo-frontend.herokuapp.com/home>

²⁶<https://pdnevnik.herokuapp.com/>



Slika 5.15: Aplikacija puštena u pogon

6. Zaključak i budući rad

Rezultat ovog projekta je izrađena web aplikacija „Planinarski dnevnik“ koja ima implementirane razne funkcionalnosti koje olakšavaju život planinarima, kao što su stvaranje novih staza, planinarskih domova i događaja, uvid u već odradene staze i događaje, pristup profilnim stranicama drugih planinara, mogućnost dodavanja prijatelja i slanja poruka te pregled popisa članova u vlastitoj planinarskoj zajednici. Aplikacija je dovršena nakon 16 tjedana, a u njezinu izradu bili su uključeni svi članovi tima.

Na početku je problem bio neiskustvo većine članova tima s odabranim alatima i programskim jezicima u kojima se izrađivala aplikacija, ali on se prebrodio ostvarenjem dobre komunikacije između članova, međusobnim pomaganjem, marljivim učenjem i trudom. Komunikacija se najvećim dijelom odvijala preko Microsoft Teamsa i WhatsAppa. Izrada aplikacije podijeljena je u dvije faze. Prva faza je veći naglasak imala na izradi dokumentacije koja ide uz aplikaciju, dok je u drugoj fazi naglasak bio na programskoj implementaciji same aplikacije.

U prvoj fazi tim se najprije upoznao, odabrao se vođa, dogovoren je okvirni plan kada bi koji dio aplikacije trebao biti dovršen i članovima su dodijeljeni prvi zadaci. U toj početnoj fazi rada također je osmišljen i izgled aplikacije i način rada većine funkcionalnosti koje aplikacija treba imati, što je članovima uvelike olakšalo kasniji rad. Izrada dokumentacije obuhvaćala je kreiranje obrazaca uporabe, sekveničkih dijagrama, modela baze podataka i dijagrama razreda. „Podignute“ su poslužiteljska i klijentska strana aplikacije i napravljene su neke jednostavnije početne funkcionalnosti poput prijave i registracije.

U drugoj fazi krenulo se u implementaciju preostalih funkcionalnosti kojih je bilo poprilično mnogo, zbog čega je druga faza bila mnogo zahtjevnija i bio je potreban intenzivan rad svih članova tima. Međutim, upravo zbog prethodno dobro razrađenog plana i vizualnog osmišljavanja problemskih zadataka, nije bilo dvojbi oko načina na koji neki dio aplikacije treba raditi i izgledati pa je tim uspio implementirati veliki dio zahtjevanih funkcionalnosti na vrijeme. U dogовору с асистентом неке unaprijed dogovorene funkcionalnosti su izbačene, a to su uglavnom funkcionalnosti vezane uz ulogu *Dežurnog planinara* što bi bio i prvi sljedeći korak

ako dođe do nastavka radi na aplikaciji. Na nekim dijelovima dokumentacije su napravljene male preinake i dodani su ostali potrebni UML dijagrami kao što su dijagram stanja, dijagram komponenti i dijagram razmještaja. Na kraju je odrđeno ispitivanje programskog rješenja pomoću Unit i Selenium testova.

Rad na ovom projektu je bio odlična prilika za stjecanje novih znanja i vještina koje će sigurno biti korisne svim članovima tima u budućnosti. Sudjelovanje na projektu obogatilo nas je za iskustvo rada u timu koji nije uvijek savršen, ali upravo zbog toga nas uči koliko je važna dobra komunikacija i vremenska usklađenost između njegovih članova. Jako smo zadovoljni s krajnjim rezultatom projekta - funkcionalna i oku ugodna aplikacija unatoč početnom neiskustvu većine članova tima.

Popis literature

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

Indeks slika i dijagrama

2.1 Primjeri sličnih aplikacija - eHPS i infoHPS	8
2.2 Primjer slične aplikacije - Mountain Project	8
3.1 Prikaz funkcionalnosti dostupnih neregistriranom ili neprijavljenom korisniku	27
3.2 Dio funkcionalnosti koje obavljaju planinari	28
3.3 Drugi dio funkcionalnosti planinara i zasebne aktivnosti dežurnog planinara	29
3.4 Prikaz funkcionalnosti koje obavlja administrator	30
3.5 Ponašajni prikaz pretraživanja planinarskih domova	31
3.6 Ponašajni prikaz brisanja vlastitih planinarskih staza	33
3.7 Ponašajni prikaz UC10 - UC11 - UC11.1	35
4.1 Arhitektura sustava	38
4.2 Model - View - Controller	40
4.3 Dijagram baze podataka	52
4.4 Dijagram razreda - konceptualni model	53
4.5 Dijagram razreda - sloj modela prvi dijagram	55
4.6 Dijagram razreda - sloj modela drugi dijagram	56
4.7 Dijagram razreda - razredi za prijenos podataka	57
4.8 Dijagram razreda - razredi za preslikavanje između sloja modela i DTO	58
4.9 Dijagram razreda korisnik - sloj nadglednik - servis - repozitorij . .	60
4.10 Dijagram razreda planinarske staze - sloj nadglednik - servis - repozitorij	61
4.11 Dijagram razreda planinarski dom - sloj nadglednik - servis - repozitorij	62
4.12 Dijagram razreda planinarski događaji - sloj nadglednik - servis - repozitorij	63
4.13 Dijagram razreda, ostalo - sloj nadglednik - servis - repozitorij . . .	64
4.14 Dijagram razreda - sigurnost	66

4.15 Dijagram razreda - iznimke i upravljanje iznimkama	67
4.16 Dijagram stanja - autentikacija	68
4.17 Dijagram stanja - domovi	69
4.18 Dijagram stanja - staze	70
4.19 Dijagram stanja - profil	71
4.20 Dijagram aktivnosti - pretraga i arhiviranje staze	72
4.21 Dijagram aktivnosti - dodavanje prijatelja	73
4.22 Dijagram aktivnosti - prihvatanje ili odbijanje prijateljstva	74
4.23 Dijagram komponenti	75
 5.1 SQL skripta za testiranje servisnog sloja	84
5.2 Rezultati JUnit testova	87
5.3 Uspješno izvršeni Selenium testovi - lokalno računalo	94
5.4 Dijagram razmještaja	95
5.5 Heroku dashboard	96
5.6 Stvaranje frontenda - Heroku	97
5.7 Dodavanje buildpacka	98
5.8 Dodavanje buildpacka	99
5.9 Dodavanje backenda	100
5.10 Dodavanje baze podataka prvi korak	100
5.11 Dodavanje baze podataka drugi korak	100
5.12 Dodavanje baze podataka treći korak	101
5.13 Dodavanje baze podataka četvrti korak	101
5.14 Dodavanje baze podataka četvrti korak	102
5.15 Aplikacija puštena u pogon	104

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 7. listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, N.Kušurin, H.Ladić, L.Ravenščak, J.Kaselj, D.Konjevod
- Teme sastanka:
 - Komentiranje zadatka koji smo dobili i komentiranje nejasnih dijelova
 - Razgovor o poznavanju tehnologija (Git, Spring, React)
 - Dogovor o tutorialima koje treba pogledati na internetu
 - Razgovor o funkcioniranju platforme Gitlab

2. sastanak

- Datum: 8.listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, N.Kušurin, H.Ladić, L.Ravenščak, J.Kaselj, D.Konjevod, K.Labor, M.Bićanić, H.Šimić
- Teme sastanka:
 - Predstavljanje načina rada i uvod u projekt
 - Rješavanje nejasnoća vezanih uz zadatak s asistentom i demonstratorom

3. sastanak

- Datum: 9.listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, N.Kušurin, H.Ladić, L.Ravenščak, J.Kaselj, D.Konjevod
- Teme sastanka:
 - Inicijalizacija projekta (back end i front end) na Gitlabu
 - Dogovor oko raspodjele poslova vezanih uz dokumentaciju

4. sastanak

- Datum: 11.listopada 2020.

- Prisustvovali: I.Martinović, M.Rajnović, J.Kaselj
- Teme sastanka:
 - Izlučivanje funkcionalnih zahtjeva

5. sastanak

- Datum: 11.listopada 2020.
- Prisustvovali: N.Kušurin, D.Konjevod, H.Ladić
- Teme sastanka:
 - Opis projektnog zadatka

6. sastanak

- Datum: 15.listopada 2020.
- Prisustvovali: N.Kušurin, D.Konjevod, H.Ladić, I.Martinović, M.Rajnović, J.Kaselj, L.Ravenšćak, K.Labor
- Teme sastanka:
 - Raspravljanje o funkcionalnim zahtjevima s asistentom
 - Dogовори за будућу комуникацију

7. sastanak

- Datum: 15.listopada 2020.
- Prisustvovali: N.Kušurin, D.Konjevod, H.Ladić, I.Martinović, M.Rajnović, J.Kaselj, L.Ravenšćak
- Teme sastanka:
 - Crtanje ekrana
 - Raspodjela dalnjih poslova

8. sastanak

- Datum: 19.listopada 2020.
- Prisustvovali: N.Kušurin, D.Konjevod
- Teme sastanka:
 - Scenariji obrazaca uporabe

9. sastanak

- Datum: 20.listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenšćak
- Teme sastanka:
 - Modeliranje baze podataka

10. sastanak

- Datum: 22.listopada 2020.

- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin, K.Labor, M.Bićanić
- Teme sastanka:
 - Pregledavanje dokumentacije i upute za nastavak s asistentom i demonstratorom

11. sastanak

- Datum: 23.listopada 2020.
- Prisustvovali: I.Martinović, J.Kaselj, H.Ladić
- Teme sastanka:
 - Razrada sekvencijskih dijagrama

12. sastanak

- Datum: 27.listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - Pokazivanje Gitlaba , sekvencijski dijagrami

13. sastanak

- Datum: 29.listopada 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin, K.Labor, M.Bićanić
- Teme sastanka:
 - Sastanak s asistentom, komentari na sekvencijske dijagrame i dijagrame obrazaca uporabe

14. sastanak

- Datum: 5.studenog 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin, K.Labor, M.Bićanić
- Teme sastanka:
 - Sastanak s asistentom i demonstratorom, upute za daljnji rad i razvoj, demonstracija napravljenog i dogovor za predaju

15. sastanak

- Datum: 12.studenog 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin, K.Labor, M.Bićanić, H.Šimić
- Teme sastanka:

- Demonstracija generičkih funkcionalnosti

16. sastanak

- Datum: 29.studenog 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - Raspodjelili se u dva tima za programiranje, i napravili plan daljnjeg rada, podijelili taskove

17. sastanak

- Datum: 01.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - diskusija napretka, preraspodijela posla

18. sastanak

- Datum: 08.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - diskusija napretka i preuzimanje novih zadataka

19. sastanak

- Datum: 10.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - rješavanje problema

20. sastanak

- Datum: 12.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - komentiranje napretka

21. sastanak

- Datum: 17.prosinca 2020.

- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - podijela novih zadataka

22. sastanak

- Datum: 22.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - riješavanje bugova

23. sastanak

- Datum: 23.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - podijela zadataka i dogovor o dalnjem radu

24. sastanak

- Datum: 28.prosinca 2020.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - komentar napretka, sinkronizacija, pruzimanje zadataka

25. sastanak

- Datum: 31.prosinca 2020.
- Prisustvovali: I.Martinović, L.Ravenščak, D.Konjevod
- Teme sastanka:
 - deploy aplikacije

26. sastanak

- Datum: 02. siječnja 2021.
- Prisustvovali: I.Martinović, M.Rajnović, L.Ravenščak, J.Kaselj, H.Ladić, D.Konjevod, N.Kušurin
- Teme sastanka:
 - preraspodijela na dokumentaciju i programiranje

Tablica aktivnosti

	Ivan Martinović	Luka Ravenščak	Marko Rajnović	Josipa Kasej	Neda Kušurin	Helena Ladić	David Konjevod
Upravljanje projektom	15	1					
Opis projektnog zadatka	3			3	6	4	8
Funkcionalni zahtjevi	7	2	4	1.5			
Opis pojedinih obrazaca	8				7		7
Dijagram obrazaca	5						
Sekvencijski dijagrami	2.5			3		3	
Opis ostalih zahtjeva					1		1
Arhitektura i dizajn sustava				4			
Baza podataka	8	2	12.5				
Dijagram razreda	12	2	2	3	3		
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							
Dnevnik sastajanja	1.5						
Zaključak i budući rad							
Popis literature							
Izrada ekrana	3	6.5	3	3	3	13	3
Back end - generičke funkcionalnosti	7	10					
Front end - generičke funkcionalnosti	7	7			3	6	3
Deploy aplikacije	12						