

Programsko inženjerstvo

Ak. god. 2022./2023.

Svjetski putnik

Dokumentacija, Rev. 1

Grupa: *DJUNGELSKOG*

Voditelj: *Ian Golob*

Datum predaje: *18. 11. 2022.*

Nastavnik: *Hrvoje Nuić*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
2.1 Korisničke uloge	5
2.2 Korist, opseg te mogući skup korisnika projekta	6
2.2.1 Korist projekta	6
2.2.2 Opseg projekta	6
3 Specifikacija programske potpore	7
3.1 Funkcionalni zahtjevi	7
3.1.1 Obrasci uporabe	9
3.1.2 Sekvencijski dijagrami	20
3.2 Ostali zahtjevi	24
4 Arhitektura i dizajn sustava	25
4.1 Programske jezice, razvojni okviri, alati i biblioteke koda	26
4.1.1 <i>Backend</i> i baza podataka	26
4.1.2 <i>Frontend</i>	26
4.2 Baza podataka	27
4.2.1 Opis tablica	27
4.2.2 Dijagram baze podataka	34
4.3 Dijagram razreda	35
4.4 Dijagram stanja	44
4.5 Dijagram aktivnosti	45
4.6 Dijagram komponenti	46
5 Implementacija i korisničko sučelje	47
5.1 Korištene tehnologije i alati	47
5.2 Ispitivanje programskog rješenja	48
5.2.1 Ispitivanje komponenti	48
5.2.2 Ispitivanje sustava	53

5.3 Dijagram razmještaja	58
5.4 Upute za puštanje u pogon	59
6 Zaključak i budući rad	61
Popis literature	62
Indeks slika i dijagrama	64
Dodatak: Prikaz aktivnosti grupe	65

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napisan opis projektnog zadatka, funkcionalni zahtjevi projekta	Ivan Kolar	31.10.2022.
0.4	Dodani <i>Use Case</i> dijagrami , i dio nefunkcionalnih zahtjeva	Ivan Kolar	03.11.2022
0.5	Arhitektura i dizajn sustava, baza podataka	Ivan Kolar	07.11.2022.
0.7	Opisi obrazaca uporabe	Ivan Kolar	08.11.2022
0.9	Sekvencijski dijagrami	Ivan Kolar	09.11.2022
0.10	Opis sekvencijskih dijagrama i opis relacija baze	Ivan Kolar	15.11.2022
0.12	Započeo dijagrame razreda	Ivan Kolar	17.11.2022.
0.14	Dovršeni dijagrami razreda	Ivan Golob	17.11.2022.
1.0	Završena dokumentacija za 1. ciklus	DJUNGEL -SKOG tim	18.11.2022
1.2	Dodani UML dijagrami 2. ciklusa	Ivan Kolar	5.1.2023
1.6	Upute za puštanje u pogon	Matej Košćec	11.1.2023

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.8	Ispitivanje programskog rješenja	Ian Golob	13.1.2023
2.0	Završena dokumentacija i projekt	DJUNGEL -SKOG tim	13.1.2023

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje aplikacije "Svjetski putnik" koja se ponaša kao društvena mreža na kojoj korisnik osvaja bedževe svojim putovanjima po svijetu, osvajanje što više teritorija na karti te što više bedževa je glavni cilj korisniku.

Primjer osvajanja bedža:

- Bedž države - *posjeti glavni grad i dvije lokacije van glavnog grada*
- Bedž grada - *posjeti minimalno 5 lokacija unutar grada*

Pokretanjem aplikacije prikazuje se ekran za prijavu, odnosno registraciju korisnika. Za kreiranje novog računa (registraciju korisnika) potrebni su podatci:

- korisničko ime
- ime korisnika
- prezime korisnika
- email adresa
- lozinka

Ovakvom registracijom korisnik dobiva prava regularnog korisnika no naknadno mu od strane administratora mogu biti dodijeljene korisničke uloge administratora i/ili kartografa. Prijavom u aplikaciju se dolazi na početnu stranicu (engl. Home page).

Postoje 3 tipa korisničkih uloga:

- regularni korisnik
- administrator
- kartograf

2.1 Korisničke uloge

Regularni korisnik ima pregled svojih prošlih putovanja na karti tako da su države koje je obišao označene pinom te obojene a one koje još nije su sive boje. Korisnik može dodavati putovanja na wishlistu uz definiranje roka do kada mora obići

mjesto. Regularni korisnik može pregledavati profil i osvojene bedževe drugih korisnika, te definirati prikazuje li javno svoja putovanja ili samo bedževe koje je osvojio.

Obilaskom lokacije korisnik treba unijeti kao dokaz sliku sebe na toj lokaciji, te ispuniti formu u kojoj se bilježi:

- datum posjeta
- vrsta prijevoza do mjesta
- ocjena gužve
- samostalno putovanje ili u društvu
- recenzija putovanja od 1 do 5
- komentar na putovanje

Kartograf ima pravo definiranja pravila za osvajanje bedževa, ali ne i ovlast za uređivanje ili uklanjanje postojećih.

Administrator ima pravo unositi države, gradove i lokacije unutar grada (objekti poput muzeja, crkvi...), odobravati prijedloge korisnika za unos novih lokacija te brisanje korisnika iz sustava.

2.2 Korist, opseg te mogući skup korisnika projekta

2.2.1 Korist projekta

Korist ovog projekta bi bila motivirati ljude zanimljivim bedževima da putuju više i posjete nove lokacije za koje možda nisu ni znali da postoje a saznali su preko bedževa, također promovira povezanost ljudi tako da omogućuje pregled i reagiranje na putovanja prijatelja. Skup mogućih korisnika su mladi ljudi koji često putuju i žele dijeliti svoja putovanja sa svojim prijateljima

2.2.2 Opseg projekta

Projekt obuhvaća ostvarenje karte po kojoj je moguće navigirati, stavljati pinove i predlagati nove lokacije, te ostvarenje društvenog aspekta aplikacije nalik društvene mreže u kojem je moguće dodavati prijatelje, pretraživati korisnike i reagirati na sadržaj (putovanja) prijatelja. U aplikaciji se osvajaju bedževi za ispunjavanje određenih zahtjeva (pravila) svakog bedža koje kreira kartograf. Osim bedževa koje je kartograf izradio, korisnik samostalno može dodavati određene lokacije na wishlist kako bi ih kasnije posjetio.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj
2. Regularni korisnik
3. Kartograf
4. Administrator
5. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) Registrirati se u sustav za što je potrebno unijeti email adresu, ime, prezime, korisničko ime i lozinku
 - (b) Prijaviti se u sustav za što je potrebno unijeti samo korisničko ime i lozinku
2. Regularni korisnik (inicijator) može:
 - (a) pregledavati osobne podatke
 - (b) mijenjati osobne podatke
 - (c) mijenjati postavke privatnosti vlastitog profila
 - (d) pretraživati druge korisničke profile
 - (e) objavljivati putovanja
 - (f) pregledavati vlastite bedževe
 - (g) dodavati putovanja na wishlistu
 - (h) objavljivati prijedloge novih lokacija
 - (i) osvajati bedževe na putovanjima
3. Kartograf (inicijator) može:
 - (a) definirati nova pravila za bedževe

4. Administrator (inicijator) može:

- (a) uklanjati korisnike iz sustava
- (b) odobriti ili odbiti prijedloge novih lokacija
- (c) unositi nove lokacije, gradove i države

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o korisnikovim putovanjima, bedževima i prijateljima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - "Registracija"

- **Glavni sudionik:** Neregistrirani korisnik, regularni korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi potrebne korisničke podatke
 3. Korisnik prima obavijest o uspješnoj registraciji
 4. Korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Odabir već zauzetog korisničkog imena ili email adrese ili unos korisničkog podatka u nedozvoljenom formatu
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC2 - "Prijava u sustav"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju (početnoj stranici)
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Korisnik unosi lozinku i korisničko ime
 2. Potvrda o ispravnosti unesenih podataka
 3. Korisnik se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za prijavu

UC3 - "Pregled osobnih podataka"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moj profil"
 2. Aplikacija prikazuje osobne podatke korisnika

UC4 - "Promjena osobnih podataka"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Izmijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici "Moj profil"
- **Opis osnovnog tijeka:**
 1. Korisnik odabire "Uredi profil"
 2. Korisnik mijenja svoje osobne podatke
 3. Korisnik spremi ili odbacuje promjene
 4. Baza podataka se ažurira u slučaju spremanja

UC5 - "Pregled osvojenih bedževa"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pregledati osvojeni bedževe
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moji bedževi"
 2. Aplikacija prikazuje osvojene bedževe korisnika

UC6 - "Pregled vlastite wishliste"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pregledati wishlistu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Lista želja"
 2. Aplikacija prikazuje listu želja

UC7 - "Pregled vlastitih putovanja"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pregledati vlastita putovanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moja putovanja"
 2. Aplikacija prikazuje putovanja korisnika u obliku objava

UC8 - "Pregled profila drugih korisnika"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pregledati objave drugih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Društvo"
 2. Korisnik pretražuje korisnika čije profil želi pogledati

UC9 - "Označavanje putovanja drugog korisnika sa "Sviđa mi se"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Označiti putovanje drugog korisnika sa "Sviđa mi se"
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i pregledava račun korisnika čije postavke privatnosti su postavljene na "javno"
- **Opis osnovnog tijeka:**
 1. Korisnik reagira na putovanje
 2. U bazi podataka se ažurira broj oznaka sa "Sviđa mi se"

UC10 - "Dodavanje stavki na wishlistu"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Dodati željenu stavku (npr. lokaciju) na wishlistu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na "Lista želja" lokaciji u aplikaciji
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Nova želja"
2. Korisnik ispunjava formu za dodavanje novih želja
3. U bazi podataka se ažurira lista želja korisnika s novom željom

UC11 - "Pregled karte, osvojenih država i pinova"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Prikazati naslovnicu (početnu stranicu) na kojoj se nalazi karta s obojenim državama i pinovima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Naslovnica"
 2. Prikazuje se stranica "Naslovnica" s kartom i pinovima

UC12 - "Izmjena postavki privatnosti"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Promijeniti postavke privatnosti, specifično prikazuju li se i putovanja ili samo bedževi
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici "Moj profil"
- **Opis osnovnog tijeka:**
 1. Korisnik odabire "Uredi profil"
 2. Korisnik izmjenjuje postavke privatnosti
 3. U bazi podataka se ažuriraju postavke privatnosti korisnika

UC13 - "Objavljivanje putovanja"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Podijeliti putovanje na profilu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici "Moja putovanja"
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Nova objava"
 2. Korisnik ispunjava formu za dodavanje novog putovanja
 3. Korisnik odabire jednu od postojećih lokacija ili šalje prijedlog za dodavanje nove lokacije
 4. Korisnik u formi također mora ispuniti polja:

- datum posjeta
 - vrsta prijevoza do mjesta
 - ocjena gužve
 - oznaka putuje li samostalno ili u društvu
 - recenzija putovanja - ocjena i komentar
5. Provjerava se ispunjava li korisnik uvjete za osvajanje bedža
 6. U bazu podataka se sprema putovanje korisnika te je vidljivo na korisnikovom profilu
- **Opis mogućih odstupanja:**
 - 13.a Korisnik je odlučio poslati prijedlog za dodavanje nove lokacije
 1. Ako administrator odobri korisnikov zahtjev dodat će se nova lokacija u skup postojećih lokacija u bazi podataka
 - 13.b Korisnik ispunjava uvjete za osvajanje bedža
 1. Korisniku se dodjeljuje bedž
 2. U bazi podataka se ažurira lista bedževa korisnika
 - 13.c Korisnik ispunjava uvjete za ispunjavanje želje s wishlista
 1. Korisniku se dodjeljuje bedž s wishlista
 2. U bazi podataka se ažurira lista bedževa korisnika
 3. Uklanja se želja s liste želja

UC14 - "Pretraživanje lokacija na karti"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Pronaći traženu lokaciju na karti
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici "Moj profil"
- **Opis osnovnog tijeka:**
 1. Korisnik odabire "Pretraživanje"
 2. Korisnik upisuje lokaciju koju želi pronaći
 3. Prikazuje se tražena lokacija na karti
- **Opis mogućih odstupanja:**
 - 14.a Ne postoji lokacija koju korisnik traži
 1. Korisniku se ispisuje odgovarajuća poruka da lokacija koju traži nije pronađena

UC15 - "Izrada novog bedža"

- **Glavni sudionik:** Kartograf

- **Cilj:** Kreirati novi bedž
- **Sudionici:** Baza podataka
- **Preduvjet:** Kartograf je prijavljen
- **Opis osnovnog tijeka:**
 1. Kartograf odabire opciju "Bedževi"
 2. Kartograf ispunjava formu za dodavanje novih bedževa:
 - ime bedža
 - opis bedža
 - slika bedža (sami bedž)
 - tip bedža
 - pravila bedža
 3. Ako kartograf odabere opciju "Spremi" novi bedž se kreira i ažurira se lista bedževa u bazi podataka

UC16 - "Pretraživanje svih bedževa"

- **Glavni sudionik:** Kartograf
- **Cilj:** Pronaći bedž
- **Sudionici:** Baza podataka
- **Preduvjet:** Kartograf je prijavljen i nalazi se na lokaciji "Bedževi" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Kartograf odabire opciju "Pretraživanje"
 2. Kartograf upisuje ime bedža koji ga zanima
 3. Prikazuju se bedževi s takvim imenom

UC17 - "Uređivanje bedževa"

- **Glavni sudionik:** Kartograf
- **Cilj:** Promijeniti podatke nekog bedža
- **Sudionici:** Baza podataka
- **Preduvjet:** Kartograf je prijavljen i nalazi se na lokaciji "Bedževi" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Kartograf odabire opciju "Uredi"
 2. Kartograf uređuje određene stavke bedža
 3. Kartograf dodaje nova pravila po potrebi
 4. Ako kartograf odabere opciju "Spremi" ažuriraju se podatci o bedžu u bazi podataka

UC18 - "Uklanjanje korisnika iz sustava"

- **Glavni sudionik:** Administrator
- **Cilj:** Ukloniti korisnika iz sustava
- **Preduvjet:** Administrator je prijavljen i nalazi se na lokaciji "Admin panel" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Administrator pronađe korisnika kojeg želi ukloniti iz sustava
 2. Administrator odabire opciju "Ukloni"
 3. Iz baze podataka briše se zapis o korisniku

UC19 - "Promjena statusa korisnika"

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti status korisnika
- **Preduvjet:** Administrator je prijavljen i nalazi se na lokaciji "Admin panel" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Administrator pronađe korisnika kojem želi promijeniti iz sustava
 2. Administrator odabire status (omogućen, onemogućen) računa korisnika
 3. U baze podataka se ažurira zapis o korisniku

UC20 - "Odbijanje ili prihvatanje prijedloga nove lokacije"

- **Glavni sudionik:** Administrator
- **Cilj:** Prihvati ili odbiti prijedlog nove lokacije
- **Preduvjet:** Administrator je prijavljen i nalazi se na lokaciji "Admin panel" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Preporuke/predložene lokacije"
 2. Administrator odlučuje je li predložena lokacija prihvatljiva
 3. Ako administrator odabere da je prihvatljiva u bazi podataka se dodaje nova lokacija
 4. Iz baze podataka se uklanja prijedlog lokacije

UC21 - "Dodavanje nove države, grada ili lokacije"

- **Glavni sudionik:** Administrator

- **Cilj:** Dodati novu lokaciju na kartu
- **Preduvjet:** Administrator je prijavljen i nalazi se na lokaciji "Admin panel" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Lokacije"
 2. Administrator ispunjava formu za dodavanje nove države, grada ili lokacije
 3. Odabirom "Spremi" ažurira se lista lokacija u bazi podataka

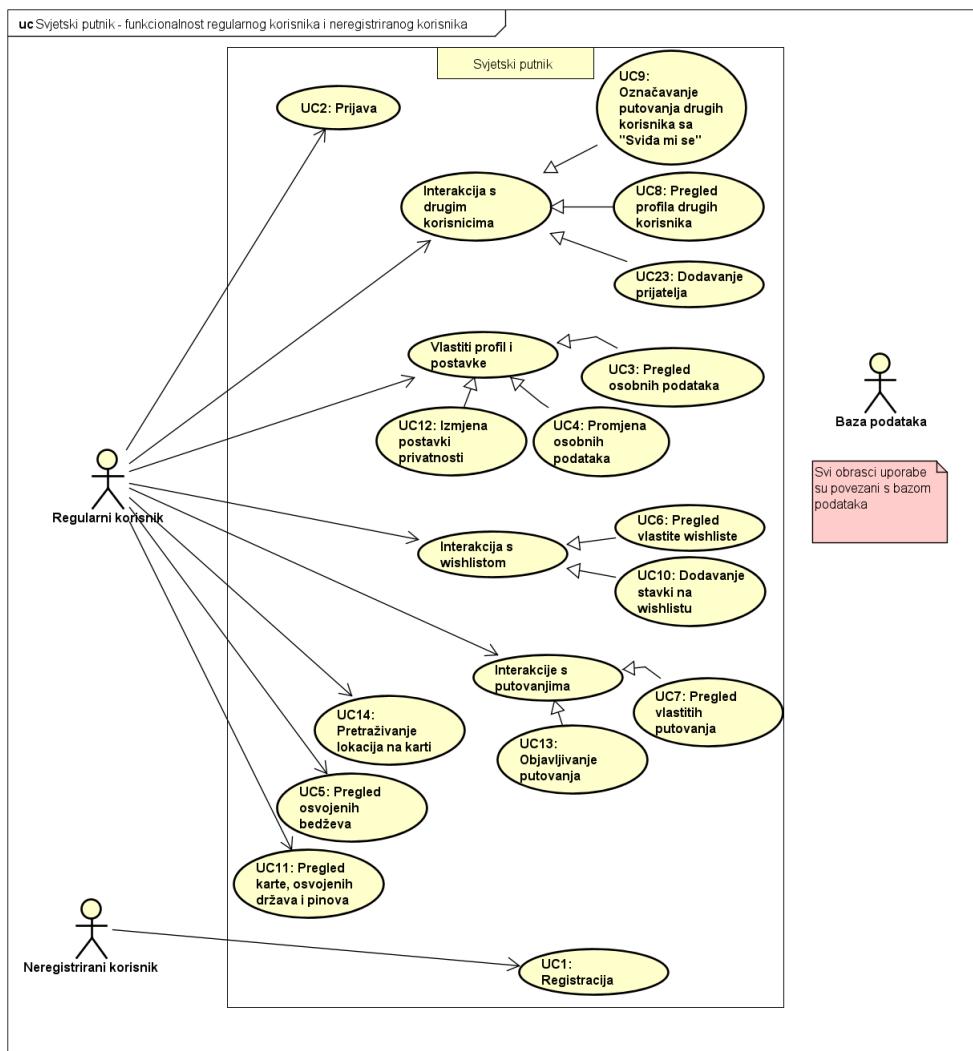
UC22 - "Promjena uloge korisnika"

- **Glavni sudionik:** Administrator
- **Cilj:** Dodijeliti novu ulogu korisniku
- **Preduvjet:** Administrator je prijavljen i nalazi se na lokaciji "Admin panel" u aplikaciji
- **Opis osnovnog tijeka:**
 1. Administrator pronađe korisniku čiju ulogu želi promijeniti
 2. Administrator mijenja ulogu korisniku, na primjer iz regularnog korisnika u kartografa
 3. U bazi podataka se ažurira zapis o ulozi korisnika

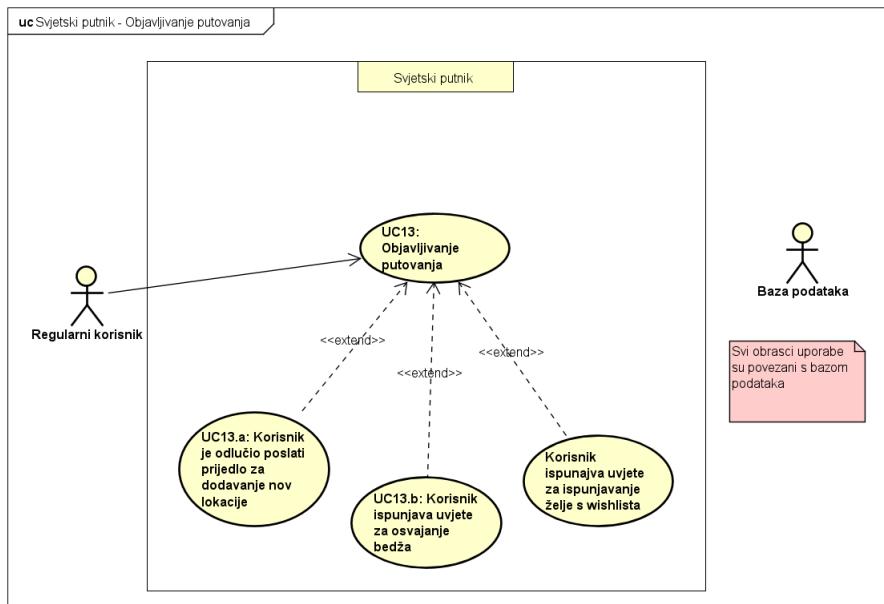
UC23 - "Dodavanje prijatelja"

- **Glavni sudionik:** Regularni korisnik
- **Cilj:** Dodati korisnika u skup prijatelja
- **Preduvjet:** Korisnik je prijavljen i pregledava profil drugog korisnika
- **Opis osnovnog tijeka:**
 1. Korisnik odabire "Dodaj prijatelja"
 2. U bazi podataka se ažurira lista prijatelja korisnika

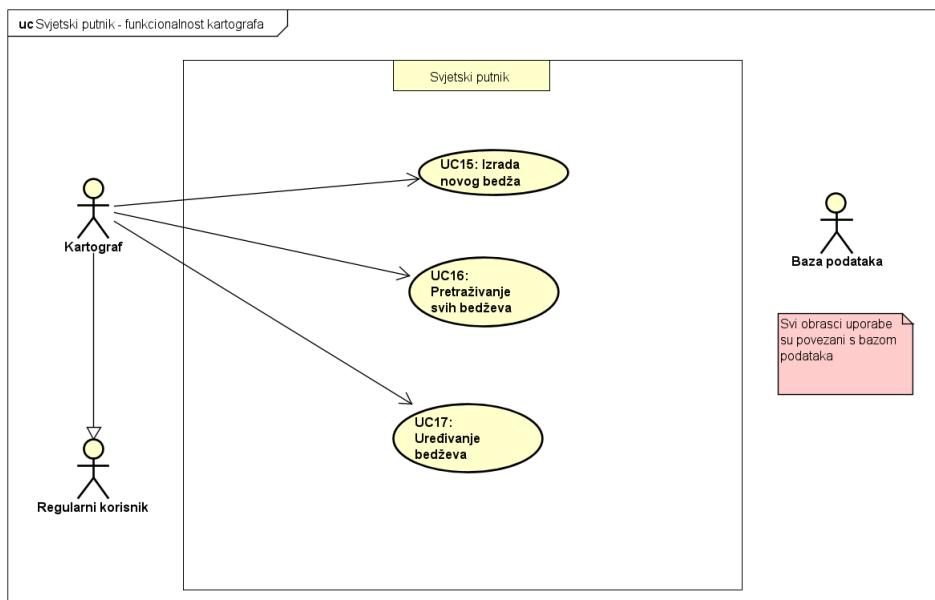
Dijagrami obrazaca uporabe



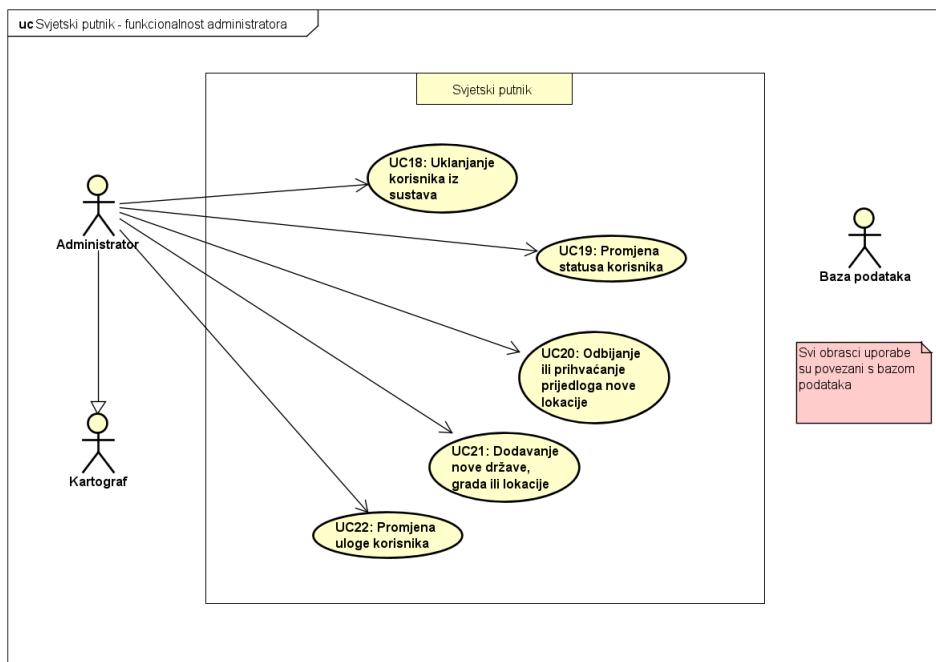
Slika 3.1: UML dijagram obrazaca uporabe - regularni korisnik



Slika 3.2: UML dijagram obrazaca uporabe - objavljivanje putovanja



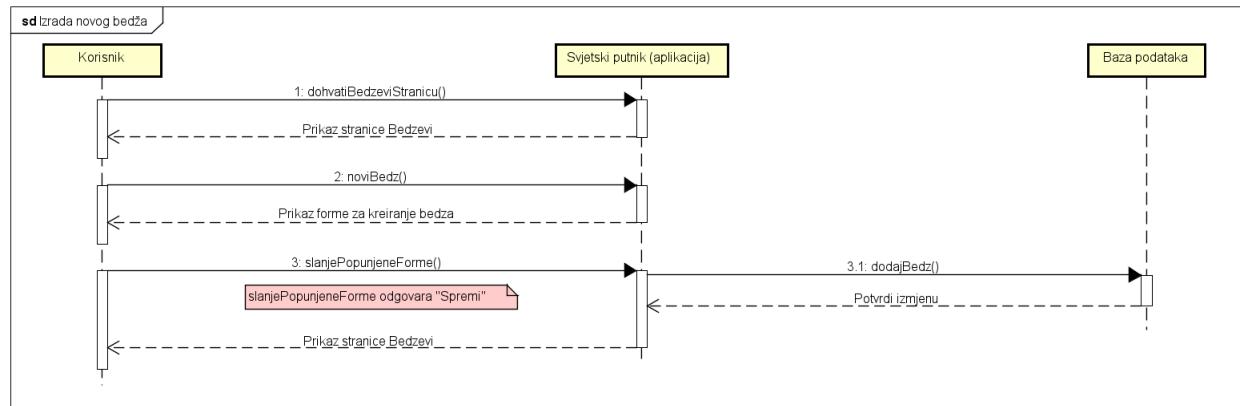
Slika 3.3: UML dijagram obrazaca uporabe - kartograf



Slika 3.4: UML dijagram obrazaca uporabe - administrator

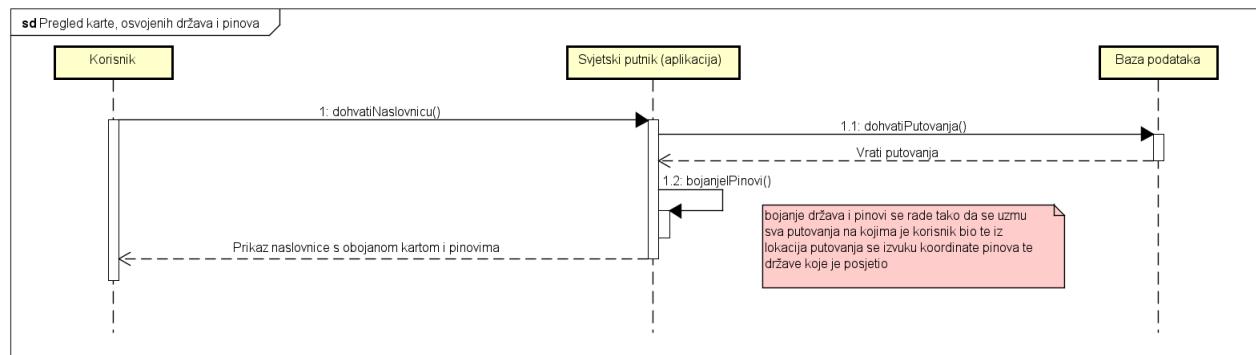
3.1.2 Sekvencijski dijagrami

Kartograf pristupa stranici "Bedževi", aplikacija prikazuje navedenu stranicu, Kartograf odabire "Novi Bedž", aplikacija prikazuje formu za kreiranje bedža. Kartograf ispunjava formu i šalje ispunjenu formu, aplikacija dodaje bedž u bazu podataka i kartografu prikazuje stranicu "Bedževi" (s vidljivim novim bedžem).



Slika 3.5: UML sekvenčni dijagram - Izrada novog bedža

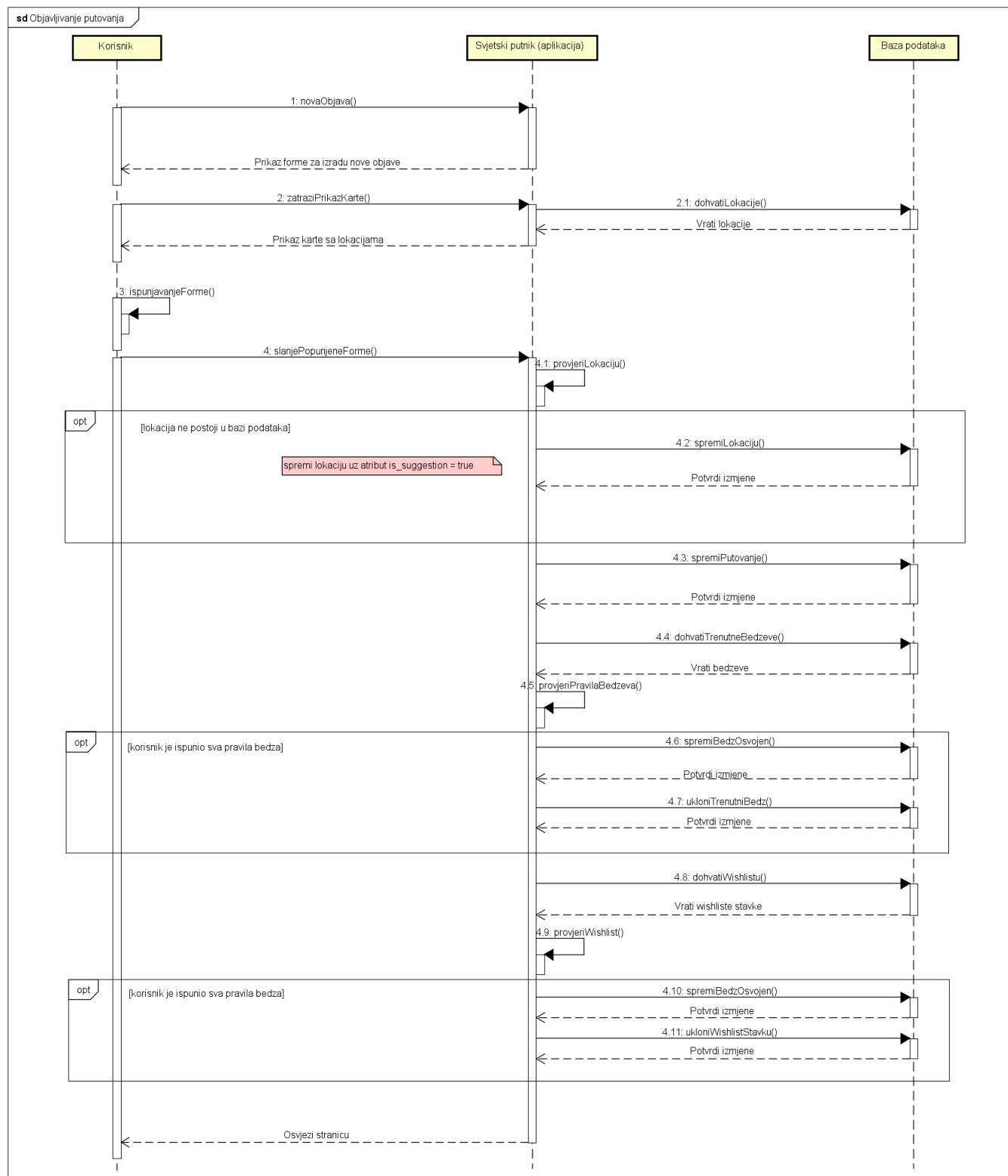
Korisnik pristupa naslovnoj stranici, aplikacija dohvaća putovanja korisnika iz baze podataka te boja države i postavlja pinove na kartu. Aplikacija korisniku prikazuje naslovnicu sa obojenom kartom i pinovima.



Slika 3.6: UML sekvenčni dijagram - Pregled karte, osvojenih država i pinova

Korisnik odabire opciju "Nova objava", aplikacija korisniku prikazuje formu za izradu nove objave. Korisnik zatraži prikaz karte, aplikacija iz baze podataka dohvaća lokacije te prikazuje korisniku kartu s lokacijama. Korisnik ispunjava formu i označuje lokaciju. Korisnik šalje formu aplikaciji, aplikacija pro-

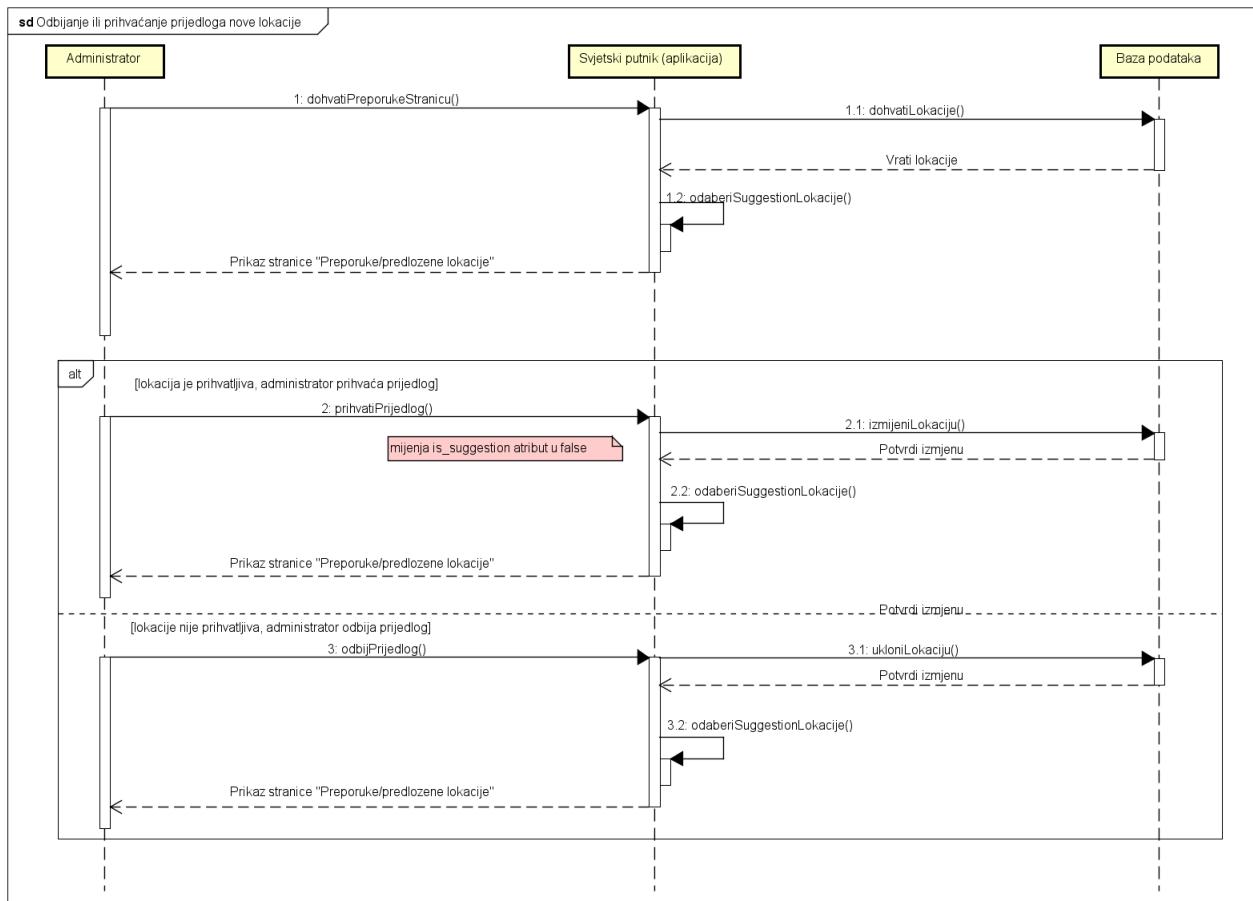
vjerava formu i ako lokacija ne postoji u bazi podataka sprema lokaciju (s flagom `is_suggestion`). Aplikacija u bazu podataka sprema putovanje. Aplikacija dohvaća trenutne bedževe korisnika iz baze podataka i provjerava je li korisnik ispunio sva pravila za bilo koji bedž, ako je u bazi podataka se sprema osvojen bedž i uklanja trenutni bedž. Aplikacija dohvaća trenutnu wishlistu korisnika iz baze podataka i provjerava je li korisnik ispunjava uvjete bilo koje wishlist stavke, ako ispunjava u bazi podataka se sprema osvojen bedž i uklanja trenutni bedž. Aplikacija ponovno prikazuje stranicu bez forme s novim putovanjem dodanim.



Slika 3.7: UML sekvencijski dijagram - Objavljivanje putovanja

Administrator odabire "Preporuke/prijedlozi (lokacija)", aplikacija dohvaća lo-

kacije iz baze podataka, odabire one s flagom `is_suggestion` i prikazuje ih Administratoru. Ako administrator smatra da je lokacija prihvatljiva onda prihvata prijedlog te aplikacija ažurira u bazi podataka da je to prava lokacija, aplikacija prikazuje stranicu "Preporuke/prijedlozi (lokacija)". Ako administrator smatra da lokacije nije prihvatljiva onda odbija prijedlog te aplikacija uklanja lokaciju iz baze podataka te aplikacija prikazuje stranicu "Preporuke/prijedlozi (lokacija)".



Slika 3.8: UML sekvencijski dijagram - Odbijanje ili prihvatanje prijedloga nove lokacije

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Neispravno korištenje korisničkog sučelja ne smije naručiti funkcionalnost sustava
- Sustav mora biti jednostavan za korištenje
- Veza prema bazi podataka treba biti zaštićena i brza
- Korisničko sučelje mora podržavati hrvatsku abecedu pri unosu i prikazu tekstualnog sadržaja

4. Arhitektura i dizajn sustava

- Struktura:
 1. Web poslužitelj
 2. *Frontend* aplikacija
 3. *Backend* aplikacija
 4. Baza podataka
- REST stil web servisa
- ORM (objektno-relacijsko preslikavanje)
- Relacijska baza podataka
- *Clean arhitektura Backend-a*

Web preglednik je program pomoću kojeg korisnik može pristupati resursima na internetu. Ima ulogu klijenta koji traži resurse od dostupnih poslužitelja. U ovom slučaju, korisnik putem preglednika šalje zahtjev za dohvaćanje (*Frontend*) *web aplikacije*, koja je implementirana kao *SPA* (engl. *Single Page Application*) - statički servirana web stranica s dinamičkim kretanjem po sadržaju ovisno o putanji, bez obaveznog osvježavanja stranice.

Frontend (korisnička) aplikacija služi za komunikaciju s *Backend* aplikacijom na poslužitelju, gdje se sva bitna poslovna logika aplikacije nalazi, a to omogućuje kroz korisničko sučelje.

Backend aplikacija prima korisničke zahtjeve i obrađuje ih, ovisno o zahtjevu korisnika pristupa bazi podataka te šalje potvrđne ili neuspješne odgovore prema korisničkoj aplikaciji, po potrebi i s dodatnim podacima u tijelu odgovora.

4.1 Programski jezici, razvojni okviri, alati i biblioteke koda

4.1.1 *Backend* i baza podataka

U *Backend* aplikaciji koristi se Java, Spring boot, JPA, Liquibase, PostgreSQL, XML, YAML, Gradle te JWT autentikacija/autorizacija.

Spring boot je Java *Framework* koji olakšava izradu web aplikacije kroz automatsku konfiguraciju ovisnih dijelova aplikacije unutar projekta te pruža razne implementacije tih dijelova koje apstrahiraju korištenje kroz intuitivna sučelja.

Liquibase se koristi za definiranje sheme baze podataka. JPA služi za preslikavanje entiteta iz PostgreSQL baze podataka na klase u *Backend* aplikaciji, a time se postiže lakše generiranje upita ovisno o pozivu metode na klasi entiteta.

YAML je format datoteke korišten za konfiguracijske datoteke aplikacije, a XML za datoteke dnevnika promjena (engl. changelog). Za upravljanje bazom podataka koristi se DBeaver, a za realizaciju *Backend* aplikacije Jetbrains IntelliJ Idea (Java). Za izgradnju cijele aplikacije koristi se Gradle, a za sigurnost se koristi JWT (engl. Json Web Token) standard za generiranje tokena koji istovremeno služe za autorizaciju i autentikaciju korisnika.

4.1.2 *Frontend*

Za *Frontend* aplikaciju koristi se Typescript, React, TailwindCSS, Vite, React Query, Formik, HeadlessUI te Figma.

React je *Framework* koji olakšava izradu web stranica kroz razne implementacije za navigaciju, dohvaćanje te prikaz podataka. Za definiranje sadržaja stranice koristi označni kod sličan HTML-u, proširen mogućnostima Typescript-a. Za definiranje stila tj. izgleda stranice koristi se TailwindCSS.

React Query koristi se za raspodjelu podataka u aplikaciji, za izradu formi koristi se biblioteka Formik, a za ponašanje kompleksnijih komponenti na stranici HeadlessUI. Vite je alat korišten za izgradnju cijele aplikacije.

Dizajn ekrana aplikacije odrđen je pomoću alata Figma, a pisanje Typescripta u Visual Studio Code-u.

4.2 Baza podataka

Baza podataka korištena u projektu je PostgreSQL.

4.2.1 Opis tablica

Trip - opisuje jedno objavljeno putovanje korisnika. U *one to many* vezi s Location, *many to one* sa User_profile.

Trip		
id	INT	primarni ključ relacije Trip
user_id	INT	strani ključ od relacije User_profile (User_profile.user_id)
location_id	INT	strani ključ od relacije Location (Location.id)
date_visited	TIMESTAMP	datum putovanja
upload_timestamp	TIMESTAMP	datum objave putovanja
transportation_type	VARCHAR	tip prijevoza
traffic_rating	VARCHAR	ocjena gužve na putovanju
is_solo	BOOLEAN	samostalno ili putovanje u društvu
trip_rating	VARCHAR	ocjena putovanja
description	VARCHAR	komentar na putovanje
image	BYTEA	slika putovanja

Trip_like je veza *many to many* između User_profile i Trip.

Trip_like		
user_id	INT	primarni ključ relacije Trip_like i strani ključ od relacije User_profile (User_profile.user_id)

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Trip_like		
trip_id	INT	primarni ključ relacije Trip_like i strani ključ od relacije Trip (Trip.id)

Location - opisuje jednu lokaciju na karti. U *one to many* vezi s Trip, *many to one* vezi s City, *one to many* sa Wishlist_entry i *many to one* s Account.

Location		
id	INT	primarni ključ relacije Location
name	VARCHAR	ime lokacije
x_coordinate	INT	x koordinata lokacije
y_coordinate	INT	y koordinata lokacije
type	VARCHAR	tip lokacije (muzej, crkva...)
city_id	INT	strani ključ od relacije City (City.id)
is_suggestion	BOOLEAN	je li lokacija prijedlog regularnog korisnika
suggested_by_user_id	INT	strani ključ od relacije Account (Account.id)

City - opisuje jedan grad. U *one to many* vezi s Location i *many to one* vezi s Country.

City		
id	INT	primarni ključ relacije City
name	VARCHAR	ime grada
country_code	INT	strani ključ od relacije Country (Country.code)

Country - opisuje jednu državu. U *one to many* vezi s City.

Country		
id	INT	primarni ključ relacije Country
name	VARCHAR	ime države

Wishlist_entry - jedna želja na popisu želja. U *many to one* vezi s Location, *many to one* sa User_profile te *one to one* vezi s Badge.

Wishlist_entry		
id	INT	primarni ključ relacije Wishlist_entry
user_id	INT	strani ključ od relacije User_profile (User_profile.user_id)
location_id	INT	strani ključ od relacije Location (Location.id)
visit_before	TIMESTAMP	deadline kada želja treba biti ispunjena
state	VARCHAR	stanje

Wishlist_badge je veza između entiteta. *One to one* veza Wishlist_entry i Badge.

Wishlist_badge		
id	INT	primarni ključ relacije Wishlist_badge i strani ključ od relacije Badge
wishlist_entry_id	INT	strani ključ od relacije Wishlist_entry (Wishlist_entry.id)

Won_badge je veza *many to many to many* Location, User_profile, Badge.

Won_badge		
user_id	INT	primarni ključ relacije Won_badge i strani ključ od relacije User_profile (User_profile.user_id)
badge_id	INT	primarni i strani ključ od relacije Badge (Badge.id)
last_location_id	INT	primarni i strani ključ od relacije Location (Location.id)
won_timestamp	TIMESTAMP	datum osvajanja bedža

Badge - opisuje jedan bedž. U *one to one* vezi s City_badge i *one to one* vezi s Country_badge.

Badge		
id	INT	primarni ključ relacije Badge
name	VARCHAR	ime bedža
image	INT	slika bedža
type	VARCHAR	tip bedža
image	BYTEA	slika bedža

City_badge - opisuje pravila za gradove. U *one to one* vezi s Badge i *one to many* vezi s City_badge_requirement.

City_badge		
id	INT	primarni relacije City i strani ključ od relacije Badge (Badge.id)
required_locations	INT	broj koliko minimalno lokacija unutar grada treba posjetiti

City_badge_requirement - opisuje pravila lokacija unutar grada. U *many to one* vezi s City_badge.

City_badge_requirement		
id	INT	primarni ključ relacije City_badge.requirement
badge_id	INT	strani ključ od relacije City_badge (City_badge.id)
required.locations	INT	broj koliko minimalno lokacija tipa location_type unutar grada treba posjetiti
location_type	VARCHAR	tip lokacije

Country_badge - opisuje pravila za države. U *one to one* vezi s Badge.

Country_badge		
id	INT	primarni ključ relacije Country_badge i strani ključ relacije od relacije Badge (Badge.id)
visit_capital_city	BOOLEAN	je li potrebno posjetiti glavni grad
required_number	INT	ako je potrebno posjetiti glavni grad, broj type koje je potrebno posjetiti izvan glavnog grad. ako nije potrebno posjetiti glavni grad, broj type koje je potrebno posjetiti unutar države
type	VARCHAR	enumeracija: grad, lokacija

Account - opisuje jedan korisnički račun. U *one to one* vezi sa User_profile i *one to many* vezi s Location.

Account		
id	INT	primarni ključ relacije Account
name	VARCHAR	ime korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Account		
surname	VARCHAR	prezime korisnika
username	VARCHAR	korisničko ime korisnika
email	VARCHAR	email korisnika
password	VARCHAR	lozinka korisnika
is_active	BOOLEAN	je li korisnikov račun aktivan

User_profile - opisuje jedan korisnički profil. U *one to one* vezi s Account, *one to many* vezi sa Trip i *one to many* vezi sa Wishlist_entry.

User_profile		
user_id	INT	primarni ključ relacije User_profile i strani ključ relacije od relacije Account (Account.id)
is_public	BOOLEAN	je li profil javan, prikazuju li se putovanja
profile_image	BYTEA	slika korisničkog profila

Friend je veza *many to many* između User_profile i User_profile.

Friend		
from_user_id	INT	primarni ključ relacije Friend i strani ključ od relacije User_profile (User_profile.user_id)
to_user_id	INT	primarni ključ relacije Friend i strani ključ od relacije User_profile (User_profile.user_id)
is_trip_friend	BOOLEAN	je li prijatelj označeno kao prijatelj s putovanja

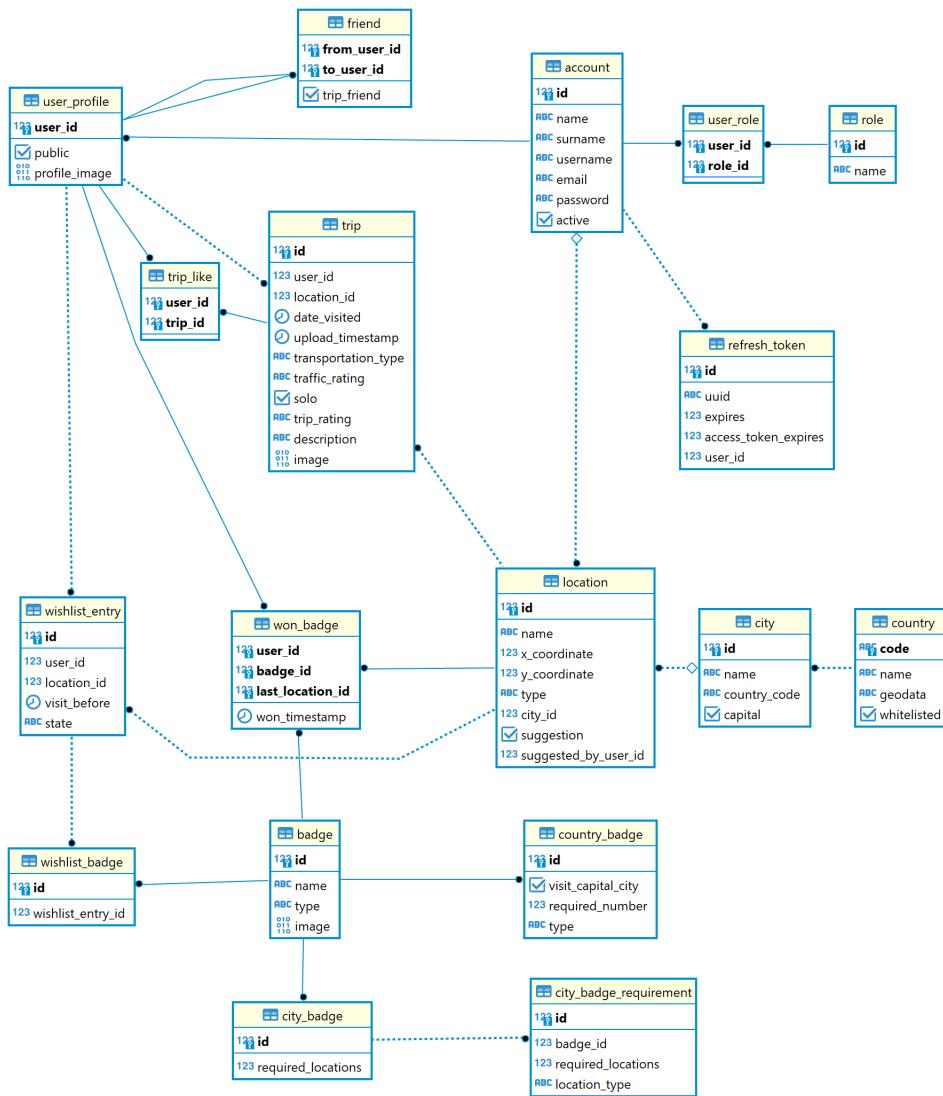
User_role je veza *many to many* između Account i Role.

User_role		
user_id	INT	primarni ključ relacije User_role i strani ključ od relacije Account
role_id	INT	primarni ključ relacije User_role i strani ključ od relacije Role

Role - opisuje jednu korisničku ulogu.

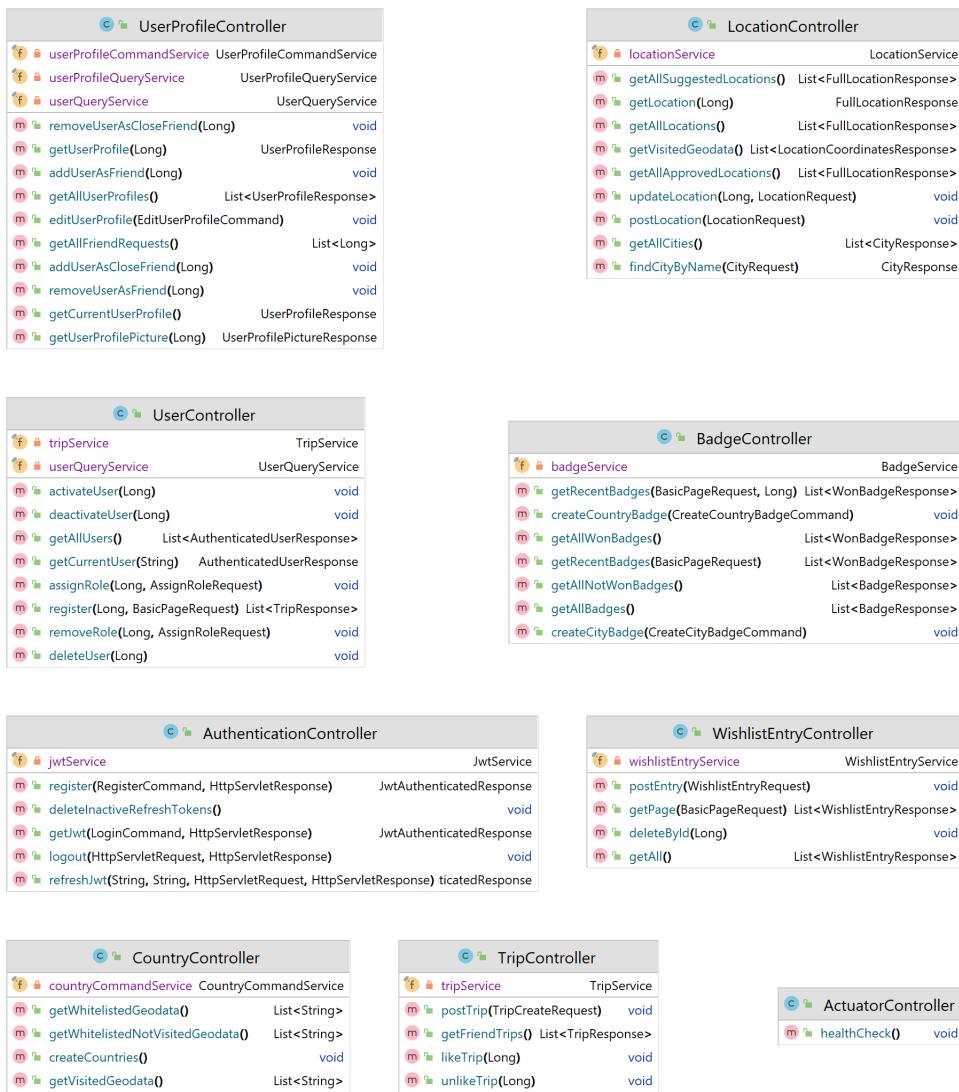
Role		
id	INT	primarni ključ relacije Role
name	VARCHAR	ime uloge

4.2.2 Dijagram baze podataka



Slika 4.1: Dijagram baze podataka

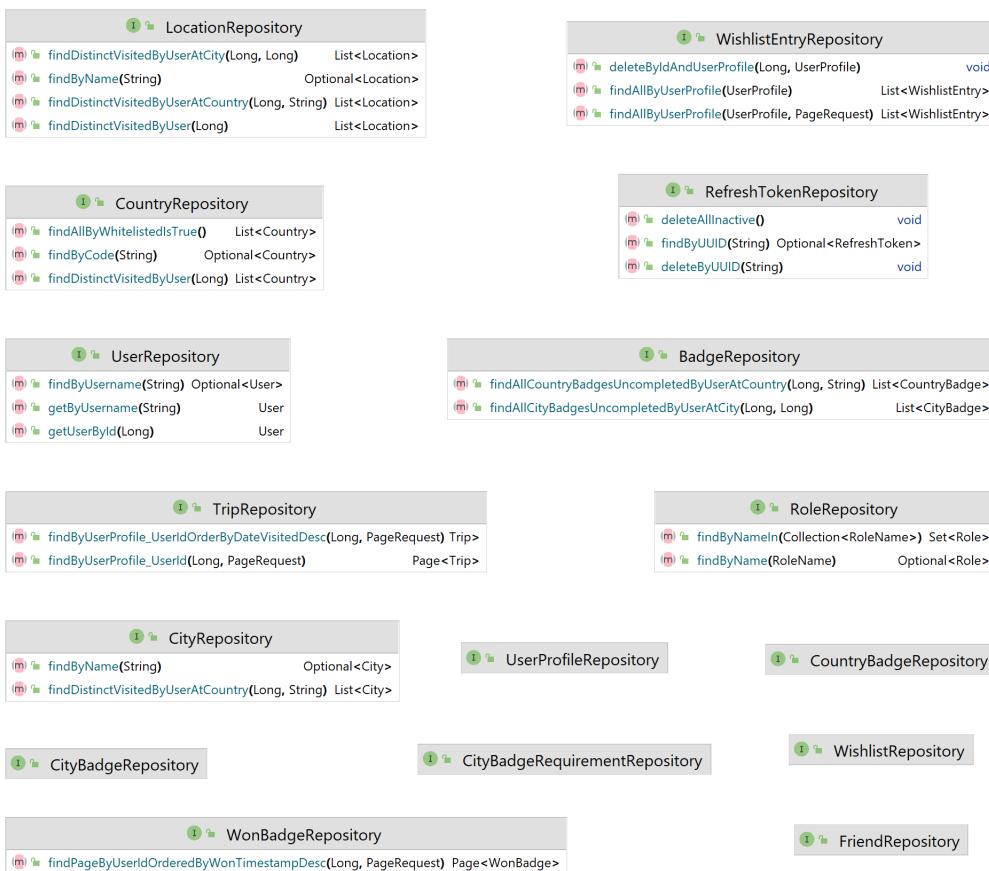
4.3 Dijagram razreda

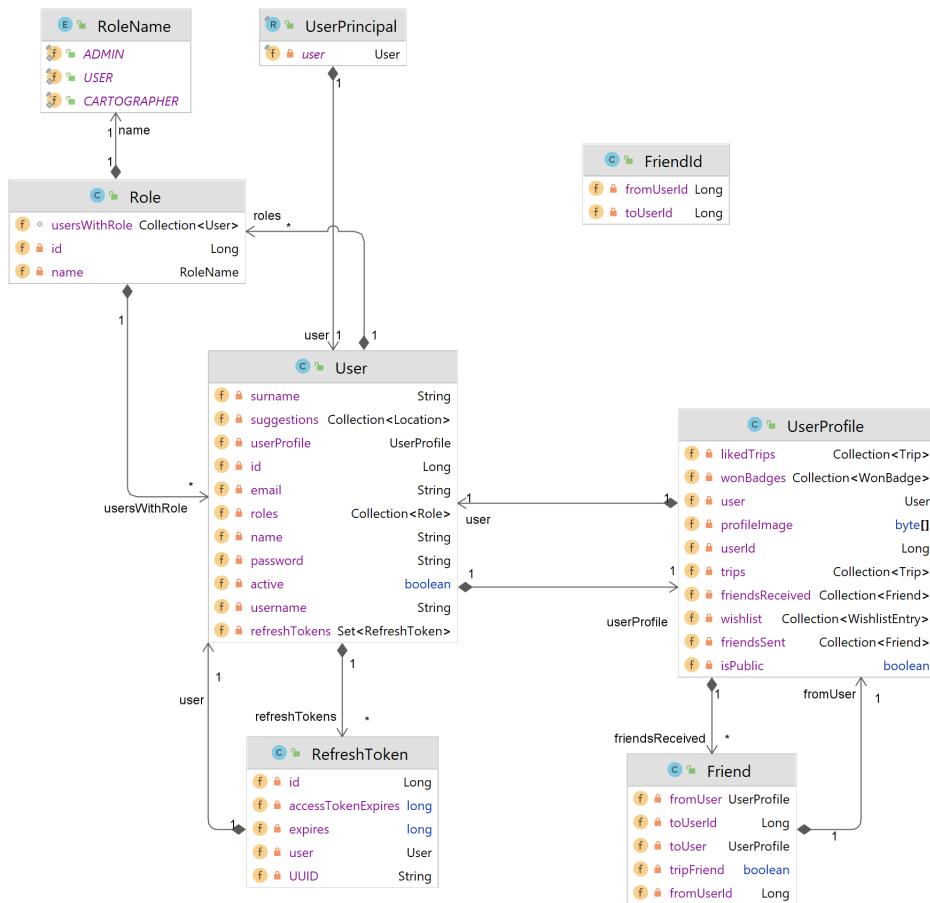


Slika 4.2: Dijagram razreda - *Controller*

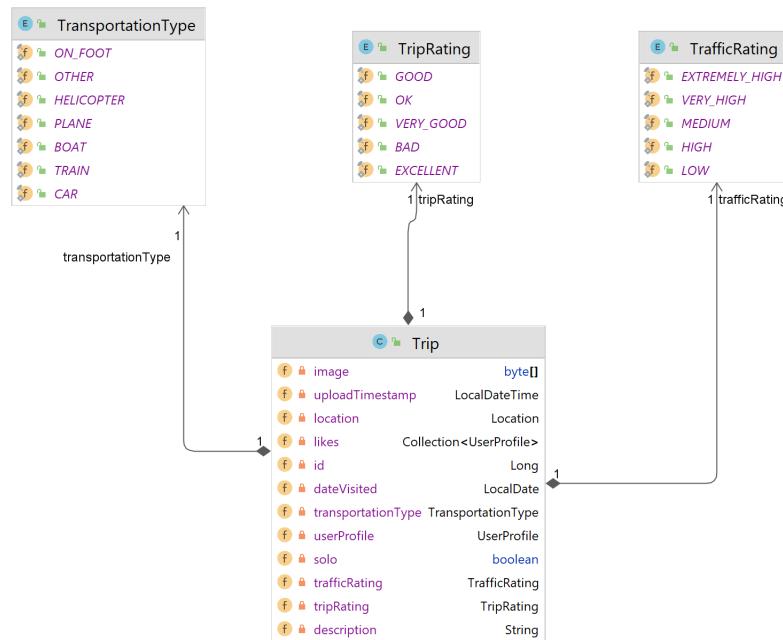


Slika 4.3: Dijagram razreda - Service

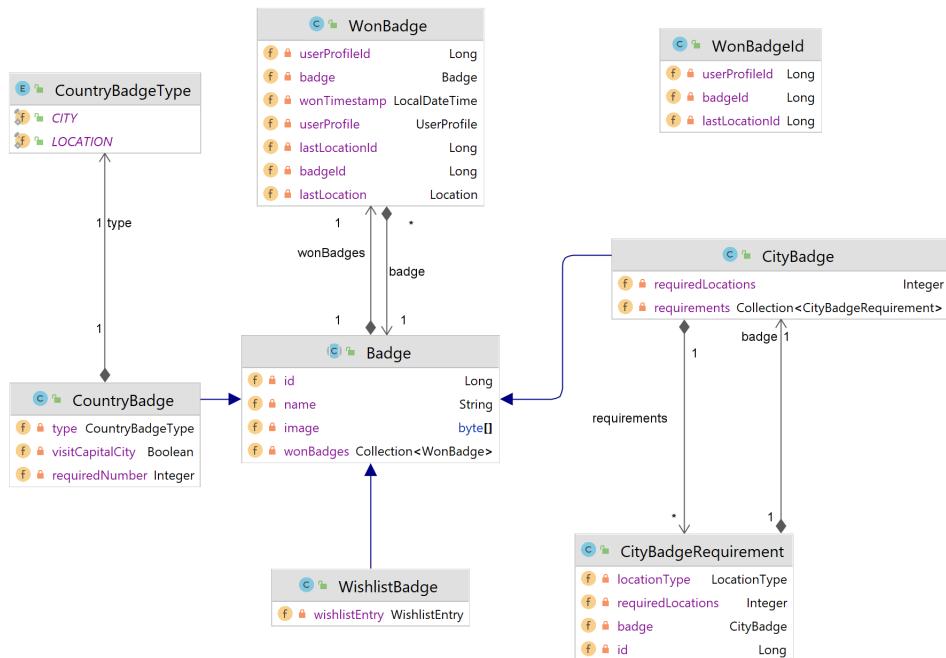
Slika 4.4: Dijagram razreda - *Repository*



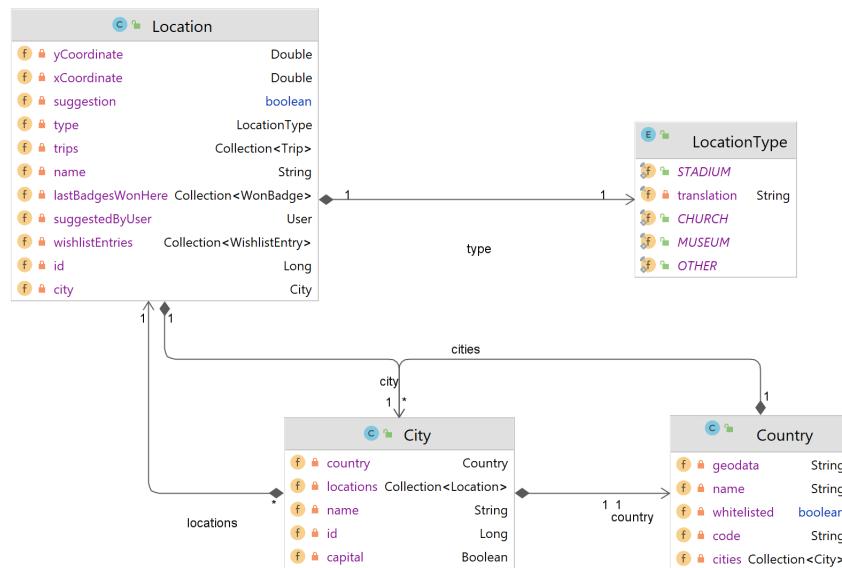
Slika 4.5: Dijagram razreda - modeli korisnika



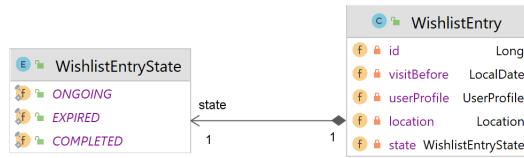
Slika 4.6: Dijagram razreda - modeli putovanja



Slika 4.7: Dijagram razreda - modeli bedževa

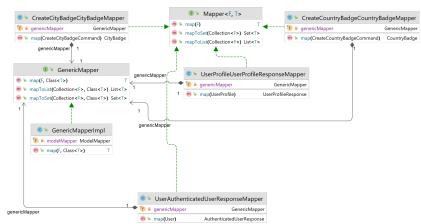


Slika 4.8: Dijagram razreda - modeli lokacija

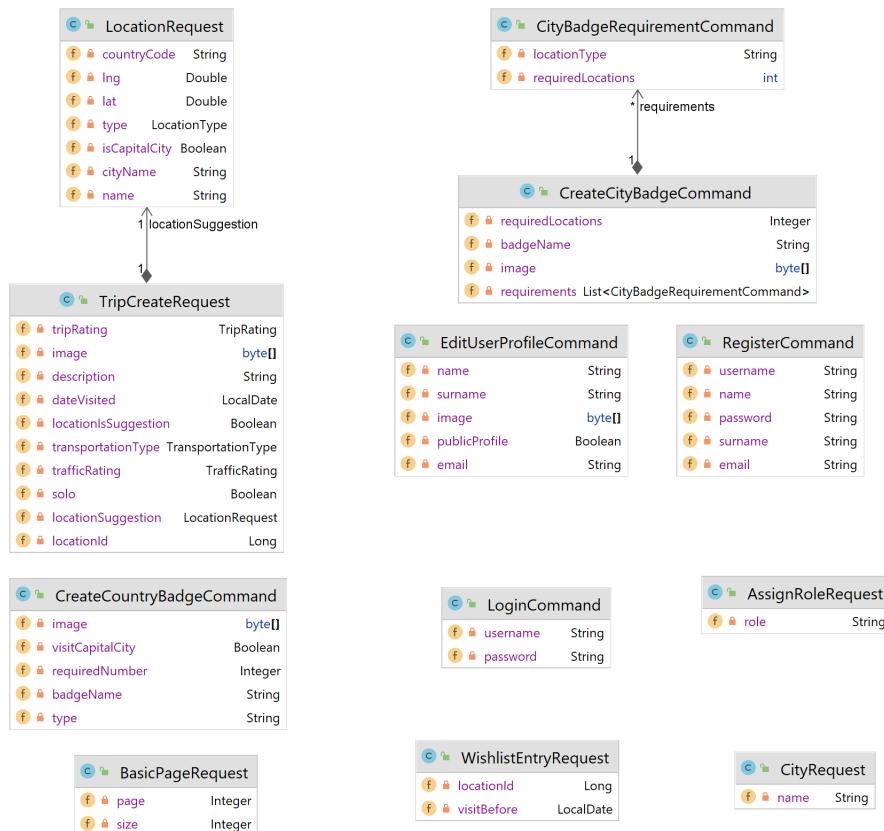
Slika 4.9: Dijagram razreda - modeli *wishlist*-a

Na dijagramima razreda modela prikazani su razredi koji reprezentiraju sve relacije koje predstavljaju neki entitet u bazi podataka, sve vezne relacije koje sadrže dodatne atribute, kompozitne ključeve te enumeracije.

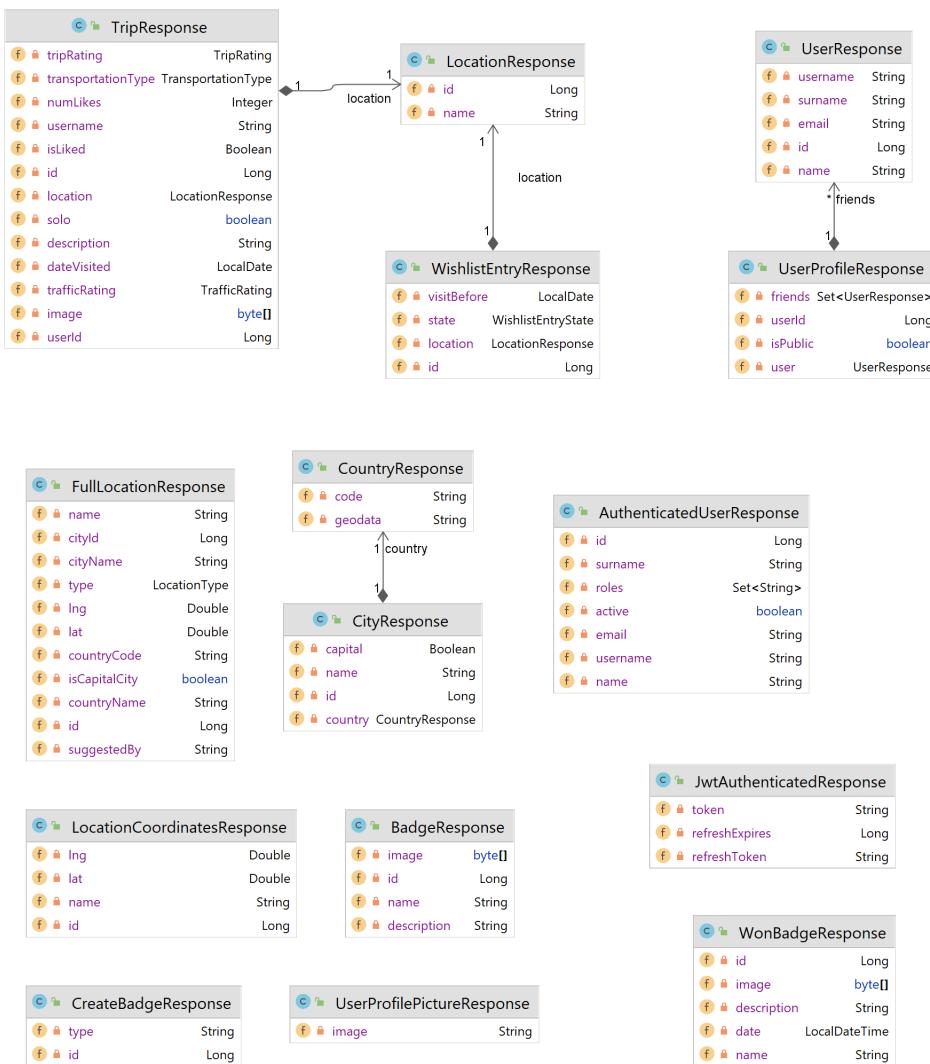
Razredi modela za svako svojstvo imaju metodu *get* i *set* te konstruktore.



Slika 4.10: Dijagram razreda - *Mapper*



Slika 4.11: Dijagram razreda - *Request*



Slika 4.12: Dijagram razreda - Response



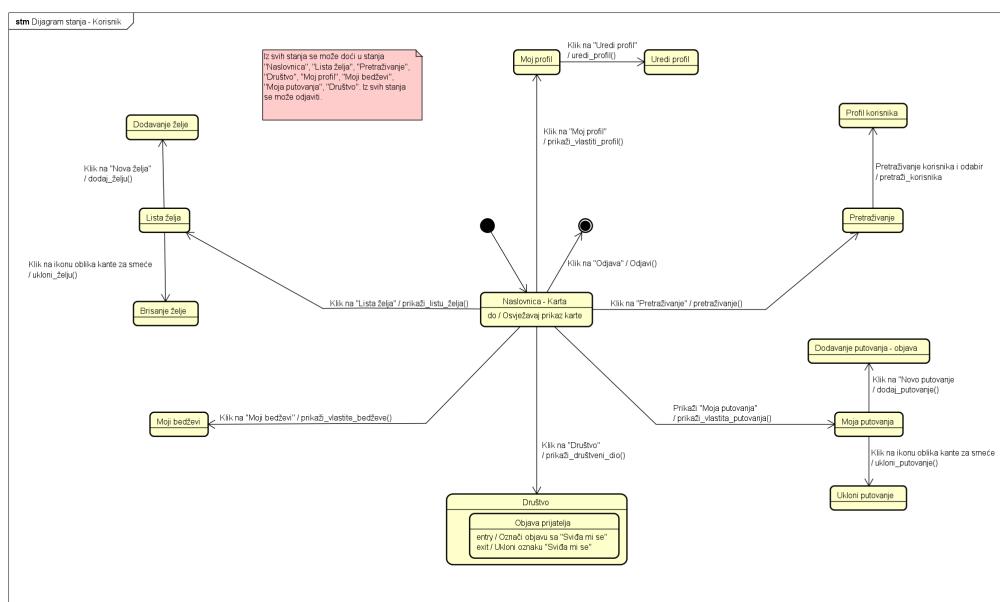
Slika 4.13: Dijagram razreda - konfiguracijski i pomoćni razredi

Na dijagramima prikazani su razredi koji se koriste za pokretanje Spring Boot-a, konfiguraciju autorizacije/autentifikacije i filtera, *Exception* i *util* razredi, generički *Mapper* i njegove implementacije koje služi za pretvaranje razreda entiteta u *DTO* (*Request* i *Response*) razred te još neki konfiguracijski razredi.

4.4 Dijagram stanja

Dijagram stanja prikazuje određenu funkcionalnost aplikacije pomoću automata. Dolje je prikazan dijagram stanja za regularnog korisnika, korisniku se prijavom prvo pokazuje naslovica odakle može pristupit ostalim dijelovima aplikacije. S obzirom na način kako je implementirano sva "glavna" stanja su međusobno povezana odnosno čine potpuno povezanu mrežu, to su uz "Naslovica", "Moji bedževi", "Moj profil", "Društvo", "Pretraživanje", "Lista želja" i "Moja putovanja". Iz svih stanja je moguće odjaviti se iz aplikacije.

Pregledavanjem vlastitog profila može se uređivati profil dodatnim klikom. Prilikom pretraživanja može se odabrati korisnik i pregledati profil navedenog korisnika. Iz stanja "Moja putovanja" moguće je dodati novo putovanje (objavu) ili ukloniti postojeću objavu. Na stranici "Lista želja" moguće je dodati novu želju na listu želja ili ukloniti postojeću želju sa liste želja. Iz stanja "Društvo" moguće je označiti objave prijatelja oznakom "Sviđa mi se".

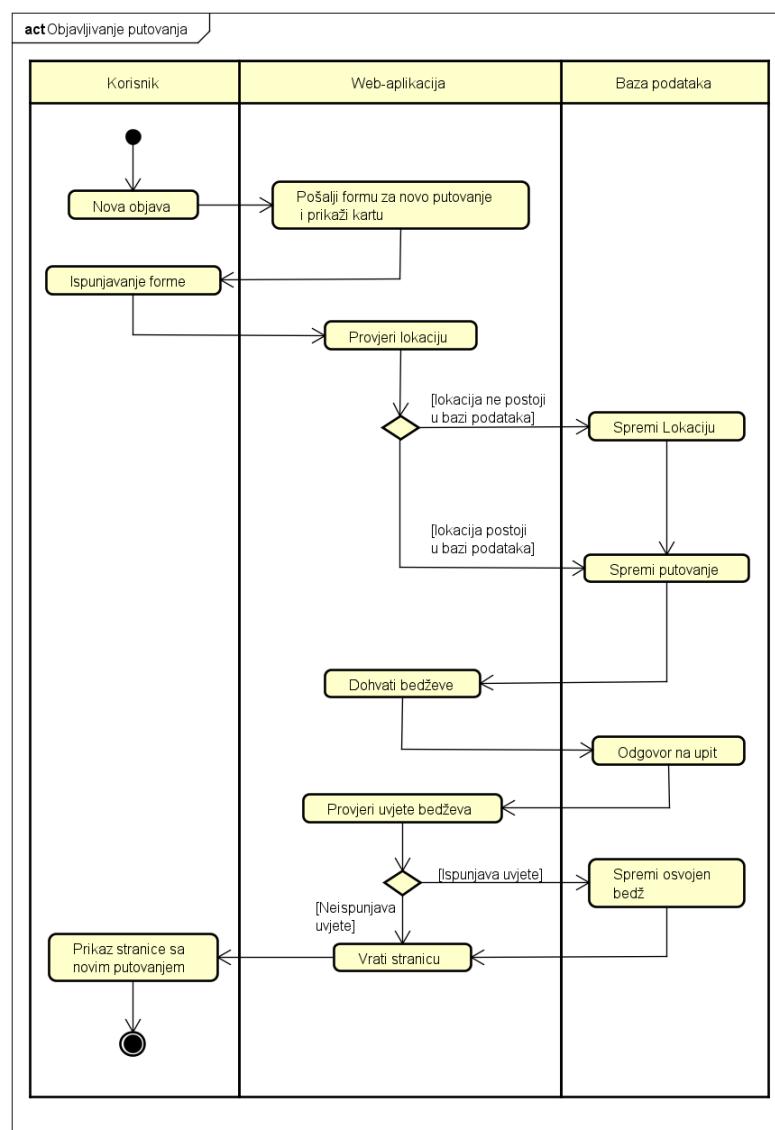


Slika 4.14: Dijagram stanja

4.5 Dijagram aktivnosti

Dijagram aktivnosti opisuje tok upravljanja određenog dijela aplikacije. Na dijagramu aktivnosti ispod prikazan je proces objavljivanja putovanja.

Korisnik odabire opciju "Nova objava" zatim mu se prikazuje forma za izradu nove objave (putovanja) te mu se prikazuje karta za označavanje mjesta. Nakon provjere označene lokacije i uvjeta bedževa spremi se putovanje (po potrebi lokacija i osvojen bedž) te se korisniku prikazuje stranica s novim putovanjem dodanim.

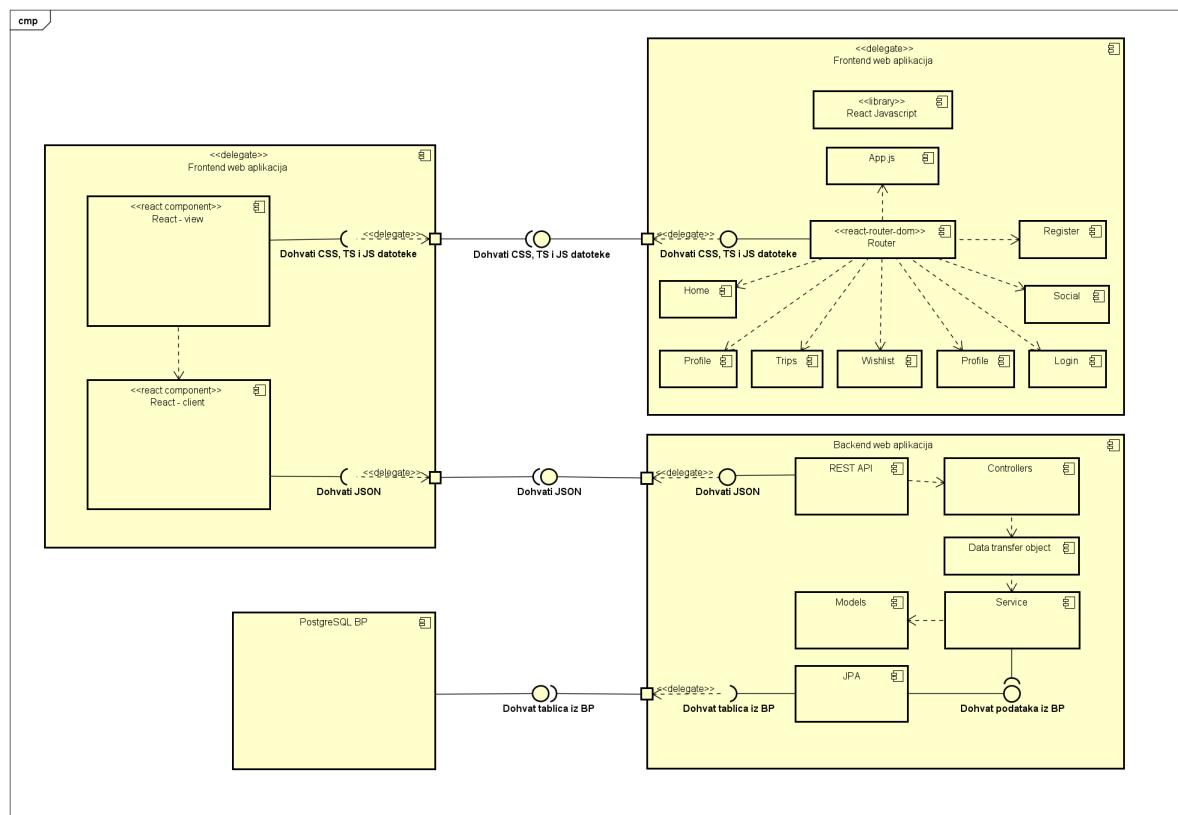


Slika 4.15: Dijagram aktivnosti

4.6 Dijagram komponenti

Dijagram komponenti prikazuje organizaciju i internu strukturu aplikacije. Na dijagramu komponenti ispod prikazana je struktura aplikacije Svjetski putnik. Sustavu se pristupa preko dva sučelja, jednim koji komunicira sa dijelom *frontenda* i jednim koji komunicira sa dijelom *backenda* (i baze podataka). Preko sučelja za dohvat CSS, JS i TS datoteka poslužuju se datoteke koje pripadaju *frontend* dijelu, a preko sučelja za dohvat JSON datoteka poslužuju se datoteke koje pripadaju *backend* dijelu.

Sve JS/TS datoteke ovise o React *library-u* iz koje dohvaćaju potrebne komponente. REST API poslužuje podatke koji pripadaju *backend* dijelu aplikacije, odnosno pruža JSON podatke. JPA je zadužen za dohvaćanje tablica iz baze podataka.



Slika 4.16: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu ostvarena je korištenjem aplikacija WhatsApp i Discord, za izradu UML dijagrama korišten je alat Astah UML, a za organizaciju i upravljanje izvornim kodom korišten je sustav Git. Kao udaljeni repozitorij projekta korištena je web platforma GitLab. Za oblikovanje dokumentacije korišten je alat Overleaf koji omogućuje prevodenje jezika LaTeX bez posebnih alata.

Razvojno okruženje korišteno za izradu *backend* dijela aplikacije je Jetbrains IntelliJ IDEA, IntelliJ olakšava razvoj aplikacija i web-stranica u programskom jeziku Java (uz Spring boot). Za izradu *frontend* dijela aplikacije korišten je Visual Studio Code razvojno okruženje s obzirom da treba koristiti više tehnologija odjednom: React, Javascript, Typescript, CSS i slično. Za bazu podataka korišten je PostgreSQL a za upravljanje bazom podataka korišten je alat DBeaver.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Provedeno je ispitivanje komponenti koristeći okvire za jedinično ispitivanje JUnit i Mockito.

Ispituju se metode razreda *BadgeService*, *LocationService* te *TripService*. Ispitni slučajevi sadrže ispitivanje regularnog ponašanja te bacanja iznimki.

Prije pokretanja ispitnih slučajeva inicijaliziraju se objekti korišteni u ispitnim slučajevima u *initData()* metodi koja je anotirana s *@BeforeEach* (zbog njene duljine, navedena metoda nije prikazana na slikama).

```
@Test
public void testGetAllWonBadges(){
    Long userId = userProfile.getUserId();

    Mockito.when(userProfileRepository.findById(userId))
        .thenReturn(Optional.ofNullable(userProfile));

    List<WonBadgeResponse> expected = wonBadgeResponses;
    List<WonBadgeResponse> received = badgeService.getAllWonBadges(userId);

    assertEquals(expected, received);
}
```

Slika 5.1: Ispitni slučaj - razred *BadgeService* - metoda *getAllWonBadges*

```
@Test
public void testGetAllWonBadges_WrongId(){
    Long userId = 1L;

    Mockito.when(userProfileRepository.findById(userId))
        .thenReturn(Optional.empty());

    assertThrows(EntityNotFoundException.class,
        () -> badgeService.getAllWonBadges(userId)
    );
}
```

Slika 5.2: Ispitni slučaj - razred *BadgeService* - metoda *getAllWonBadges* - bacanje iznimke

```
@Test
public void testSaveNewLocation(){
    LocationRequest locationRequest = locationRequest1;
    City requestCity = city1;
    Location expectedLocation = location1;
    Country requestCountry = country1;

    Mockito.when(cityRepository.findByName(requestCity.getName()))
        .thenReturn(Optional.of(requestCity));
    Mockito.when(countryRepository.findById(locationRequest.getCountryCode()))
        .thenReturn(Optional.of(requestCountry));

    locationService.saveNewLocation(locationRequest);

    verify(locationRepository, times(wantedNumberOfInvocations: 1))
        .save(eq(expectedLocation));
}
```

Slika 5.3: Ispitni slučaj - razred *LocationService* - metoda *saveNewLocation*

```
@Test
public void testUpdateLocation(){
    Long locationId = 1L;
    Location location = location1;
    LocationRequest updateRequest = locationRequest2;
    City requestCity = city2;

    Location updatedLocation = location2;

    Mockito.when(locationRepository.findById(locationId))
        .thenReturn(Optional.of(location));
    Mockito.when(cityRepository.findByName(updateRequest.getCityName()))
        .thenReturn(Optional.of(requestCity));

    locationService.updateLocation(locationId, updateRequest);

    verify(locationRepository, times(wantedNumberOfInvocations: 1))
        .save(updatedLocation);
}
```

Slika 5.4: Ispitni slučaj - razred *BadgeService* - metoda *updateLocation*

```
@Test
public void testGetRecentTrips(){
    Long user1Id = user1.getId();
    user1.setRoles(Set.of(new Role(RoleName.ADMIN)));

    Long user2Id = user2.getId();

    BasicPageRequest basicPageRequest = new BasicPageRequest( page: 0, size: 2);
    PageRequest pageRequest = PageRequest.of(
        basicPageRequest.getPage(),
        basicPageRequest.getSize()
    );

    Mockito.when(userRepository.findById(user1.getId()))
        .thenReturn(Optional.ofNullable(user1));
    Mockito.when(userRepository.findById(user2.getId()))
        .thenReturn(Optional.ofNullable(user2));

    Mockito.when(
        tripRepository.findByUserProfile_UserIdOrderByDateVisitedDesc(
            user2Id,
            pageRequest)
        .thenReturn(new PageImpl<>(List.of(trip1, trip2)));

    List<TripResponse> expected = List.of(tripResponse1, tripResponse2);
    List<TripResponse> received =
        tripService.getRecentTrips(user1Id, user2Id, basicPageRequest);

    assertEquals(expected, received);
}
```

Slika 5.5: Ispitni slučaj - razred *TripService* - metoda *getAllWonBadges* - bacanje iznimke

```
@Test
public void testGetRecentTrips_NotFriends(){
    Long user1Id = user1.getId();
    user1.setRoles(Set.of(new Role(RoleName.USER)));

    Long user2Id = user2.getId();

    BasicPageRequest basicPageRequest = new BasicPageRequest( page: 0, size: 3);

    Mockito.when(userRepository.findById(user1.getId()))
        .thenReturn(Optional.ofNullable(user1));
    Mockito.when(userRepository.findById(user2.getId()))
        .thenReturn(Optional.ofNullable(user2));

    assertThrows(AccessDeniedException.class,
        () -> tripService.getRecentTrips(user1Id, user2Id, basicPageRequest)
    );
}
```

Slika 5.6: Ispitni slučaj - razred *TripService* - metoda *getRecentTrips* - bacanje iznimke

Test Results		138 ms
✓	BadgeServiceTest	75 ms
✓	testGetAllWonBadges()	70 ms
✓	testGetAllWonBadges_WrongId()	5 ms
✓	LocationServiceTest	25 ms
✓	testSaveNewLocation()	19 ms
✓	testUpdateLocation()	6 ms
✓	TripServiceTest	38 ms
✓	testGetRecentTrips_NotFriends	30 ms
✓	testGetRecentTrips()	8 ms

Slika 5.7: Rezultati ispitivanja komponenti

5.2.2 Ispitivanje sustava

Provedeno je ispitivanje sustava koristeći Selenium WebDriver (uz Chrome-ov Web-Driver) u programskom jeziku Java uz pomoć okvira za jedinično ispitivanje JUnit.

Konstante koje završavaju na *XPATH* predstavljaju izravnu putanju do elementa na stranici. Konstanta *BASE_URL* predstavlja URL web stranice. Prije pokretanja testova pokreće se funkcija *initDriver()* koja inicijalizira WebDriver.

Kako bi se uspješno izvršili ispitni slučajevi u sustavu mora postojati registrirani korisnik s korisničkim imenom "*user*" te lozinkom "*password123*" te ne smije postojati korisnik s korisničkim imenom "*NameSurname*".

```
@Before
public void initDriver(){
    System.setProperty(
        "webdriver.chrome.driver",
        "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe"
    );
    driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
}
```

Slika 5.8: Inicijalizacija WebDriver-a

```
@Test
public void testLogin() {
    driver.get(BASE_URL + "login");

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

    element = driver.findElement(By.name("password"));
    element.sendKeys(...keysToSend: "password123");

    driver.findElement(By.xpath(LOGIN_BUTTON_XPATH)).click();

    // Checks if the home button exists and is selected
    assertTrue(driver.findElement(By.xpath(HOME_BUTTON_XPATH)) is WebElement
        .getAttribute(name: "class") String
        .contains("bg-base-bg"))
};

    driver.close();
}
```

Slika 5.9: Ispitni slučaj - prijava

```
@Test
public void testLogout(){
    driver.get(BASE_URL + "login");

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

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "password123");

    driver.findElement(By.xpath(LOGIN_BUTTON_XPATH)).click();
    driver.findElement(By.xpath(LOGOUT_BUTTON_XPATH)).click();

    assertTrue(driver.findElement(By.xpath(LOGIN_BUTTON_XPATH)).isDisplayed());

    driver.close();
}
```

Slika 5.10: Ispitni slučaj - odjava

```
@Test
public void testRegister(){
    driver.get(BASE_URL + "login");

    driver.findElement(By.xpath(REGISTER_BUTTON_XPATH)).click();

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

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

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

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

    element = driver.findElement(By.name("password"));
    element.sendKeys( ...keysToSend: "password1234");

    driver.findElement(By.xpath(REGISTER_FINISH_BUTTON_XPATH)).click();

    // Checks if the home button exists and is selected
    assertTrue(driver.findElement(By.xpath(HOME_BUTTON_XPATH)) WebElement
        .getAttribute( name: "class") String
        .contains("bg-base-bg")
    );

    driver.close();
}
```

Slika 5.11: Ispitni slučaj - registracija

```
@Test
public void testRegister_requiredFields(){
    driver.get(BASE_URL);

    driver.findElement(By.xpath(REGISTER_BUTTON_XPATH)).click();
    driver.findElement(By.xpath(REGISTER_FINISH_BUTTON_XPATH)).click();

    List<String> elementNames =
        List.of("name", "surname", "email", "username", "password");

    elementNames.forEach(name ->{
        WebElement element = driver.findElement(By.name(name));
        assertTrue(element.getAttribute("name").contains("red"));
    });

    driver.close();
}
```

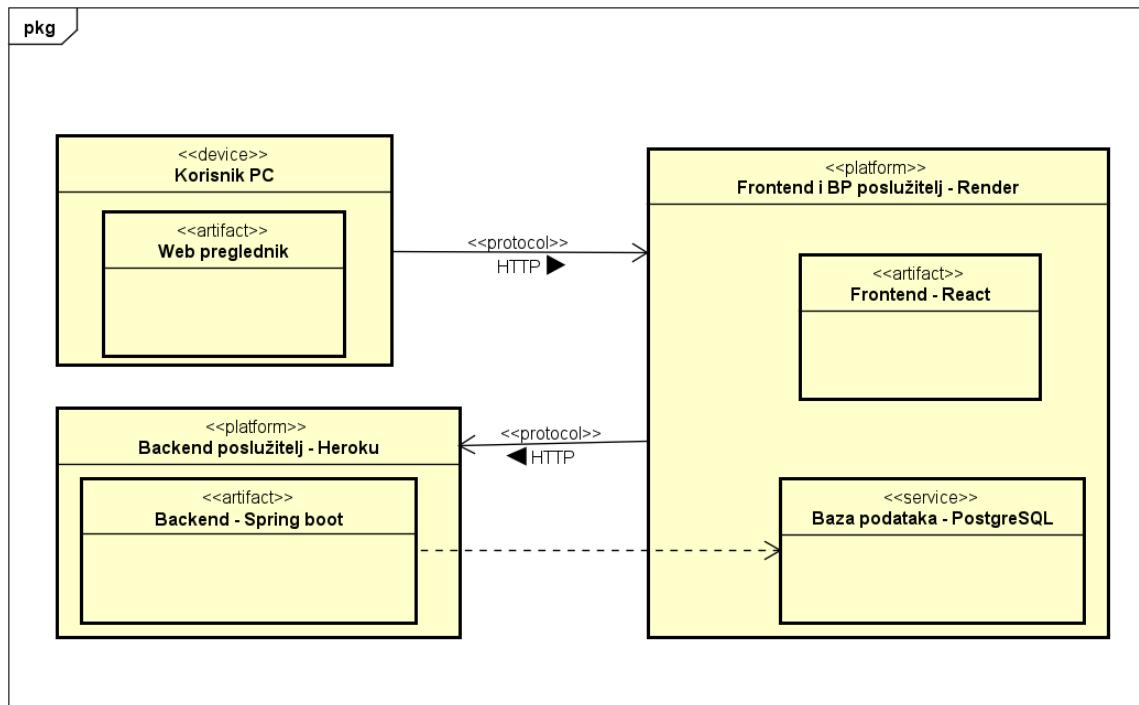
Slika 5.12: Ispitni slučaj - validacija polja tijekom registracije

▼	✓ IntegrationTest	10 sec 326 ms
	✓ testLogin	3 sec 312 ms
	✓ testRegister_requiredFields	1 sec 969 ms
	✓ testRegister	2 sec 519 ms
	✓ testLogout	2 sec 526 ms

Slika 5.13: Rezultati ispitivanja sustava

5.3 Dijagram razmještaja

Dijagram razmještaja opisuje topologiju sklopolja i programsku potporu koja se koristi u implementaciji. Korisnici koriste web preglednik kako bi pristupili web aplikaciji. Trenutna implementacija koristi dva različita poslužitelja, Heroku kao *backend* poslužitelja i Render kao *frontend* poslužitelja i poslužitelja baze podataka.



Slika 5.14: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Instalacija poslužitelja baze podataka

Potrebno je preuzeti Docker Desktop aplikaciju. Za operacijski sustav Windows potrebno je instalirati i WSL2 podršku. Provodi se standardna instalacija s postavljanjem korisnika.

Pokretanje poslužitelja baze podataka

Nakon instalacije potrebno je pokrenuti Docker kontejner preko ‘docker-compose.yaml’ datoteke. Pozicionira se u direktorij gdje se datoteka nalazi te se pokreće korištenjem naredbe ‘docker-compose up’. U datoteci se nalazi potrebna konfiguracija za pozizanje PostgreSQL baze podataka, sa postavljenim sučeljima i informacijama za spajanje.

Pokretanje poslužiteljske strane aplikacije

Potrebno je instalirati Gradle i Java programsku podršku. Nakon instalacije potrebno se pozicionirati u root direktorij aplikacije te pokrenuti naredbu ‘gradlew assemble’. Nakon što se aplikacija kompajlira, za pokretanje je potrebno upisati naredbu ”‘java -jar build/libs/[naziv_izvršne_datoteke].jar’.

Spajanje sa bazom podataka

Za stvaranje konekcije prema bazi podataka koristi se konfiguracija unutar aplikacije. Za aplikaciju je pri pokretanju potrebno definirati sve varijable okruženja potrebne za konfiguraciju, kao što su varijable za spajanje na bazu podataka. Primjer: ‘java -jar build/libs/app.jar -Dserver.port=8080 -DB_URL=jdbc:postgresql://db.url -DB_USERNAME=username -DB_PASSWORD=password -DB_SCHEMA=schema’.

Pokretanje klijentske strane aplikacije

Za pokretanje klijentske strane potrebno je instalirati Node.js programsku podršku. Nakon instalacije potrebno se pozicionirati u root direktorij poslužiteljske aplikacije te izvršiti naredbu ‘npm run build’. Nakon izgradnje statičke stranice sa svim potrebnim resursima, poslužiteljska aplikacija za serviranje tih resursa pokreće se naredbom ‘npm start’.

Spajanje klijentske i poslužiteljske strane aplikacije

Isto kao i kod spajanja na bazu podataka, klijentska aplikacija ima konfiguriranog posrednika (eng. “proxy”) pomoću kojeg se spaja prema poslužiteljskoj aplikaciji. Potrebno je dodati variable okruženja pri pokretanju aplikacije. Primjer: ‘PORT=3000 APP_BASE_URL=https://url.do.posluzitelja node app.js

6. Zaključak i budući rad

Zadatak grupe "DJUNGELSKOG" je bio razvoj web aplikacije za praćenje putovanja po svijetu i dijeljenje istih sa priateljima. Nakon 17 tijedana rada na aplikaciji ostvaren je cilj i aplikacija "Svjetski putnik" je u potpunosti izrađena.

Izradu projekta možemo promatrati u dvije faze, svaka odgovara jednom ciklusu predavanja semestra. U prvoj fazi primaran cilj je bio upoznat se i osmisliti kako ostvariti aplikaciju te postići funkcionalnu alfa inačicu web aplikacije. Česti sastanci tima su omogućili dobru organiziranost i koliko god moguću ravnomernu raspodijelu posla unutar tima.

Druga faza projekta je bila različita od prve, primarno je bilo jako puno intenzivnog rada kako bi se ispunili *deadline*-ovi. Dio tima se prvi put susreo sa određenim tehnologijama te je bilo potrebno uložiti dosta vremena kako bi ih se naučilo koristiti. S obzirom na dobar plan iz prve faze projekta, u drugoj fazi je tim samo trebao pratiti navedeni plan što je u konačnici dovelo do gotovog proizvoda.

Kroz obje faze su se izradivali UML dijagrami koji opisuju funkcionalnost aplikacije iz različitih perspektiva. Osim dijagrama izrađena je dokumentacija koja opisuje velik dio funkcionalnosti projekta i služi budućim korisnicima kao pomoć pri uporabi aplikacije ili izmjenama aplikacije.

Moguće proširenje postojeće inačice sustava je izrada mobilne aplikacije te dodavanje 3D modela posjećenih objekata bedževima kako bi se postigla zanimljivija estetika.

Sudjelovanje na ovakovom projektu omogućilo je svim članovima tima stjecanje vrijednog iskustva rada u timu te upoznavanja novih tehnologija i dokumentiranja većeg projekta. Kao tim zadovoljni smo postignutim rješenjem i uspjehom jer smo uspjeli ostvariti cilj uz sve druge obaveze koje svaki član tima ima.

Popis literature

Kontinuirano osvježavanje

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. The Unified Modeling Language, <https://www.uml-diagrams.org/>
4. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

3.1	UML dijagram obrazaca uporabe - regularni korisnik	17
3.2	UML dijagram obrazaca uporabe - objavljivanje putovanja	18
3.3	UML dijagram obrazaca uporabe - kartograf	18
3.4	UML dijagram obrazaca uporabe - administrator	19
3.5	UML sekvencijski dijagram - Izrada novog bedža	20
3.6	UML sekvencijski dijagram - Pregled karte, osvojenih država i pinova	20
3.7	UML sekvencijski dijagram - Objavljinje putovanja	22
3.8	UML sekvencijski dijagram - Odbijanje ili prihvatanje prijedloga nove lokacije	23
4.1	Dijagram baze podataka	34
4.2	Dijagram razreda - <i>Controller</i>	35
4.3	Dijagram razreda - <i>Service</i>	36
4.4	Dijagram razreda - <i>Repository</i>	37
4.5	Dijagram razreda - modeli korisnika	38
4.6	Dijagram razreda - modeli putovanja	39
4.7	Dijagram razreda - modeli bedževa	39
4.8	Dijagram razreda - modeli lokacija	40
4.9	Dijagram razreda - modeli <i>wishlist-a</i>	40
4.10	Dijagram razreda - <i>Mapper</i>	41
4.11	Dijagram razreda - <i>Request</i>	41
4.12	Dijagram razreda - <i>Response</i>	42
4.13	Dijagram razreda - konfiguracijski i pomoćni razredi	43
4.14	Dijagram stanja	44
4.15	Dijagram aktivnosti	45
4.16	Dijagram komponenti	46
5.1	Ispitni slučaj - razred <i>BadgeService</i> - metoda <i>getAllWonBadges</i> . . .	48
5.2	Ispitni slučaj - razred <i>BadgeService</i> - metoda <i>getAllWonBadges</i> - bacanje iznimke	49
5.3	Ispitni slučaj - razred <i>LocationService</i> - metoda <i>saveNewLocation</i> . . .	49

5.4 Ispitni slučaj - razred <i>BadgeService</i> - metoda <i>updateLocation</i>	50
5.5 Ispitni slučaj - razred <i>TripService</i> - metoda <i>getAllWonBadges</i> - bacanje iznimke	51
5.6 Ispitni slučaj - razred <i>TripService</i> - metoda <i>getRecentTrips</i> - bacanje iznimke	52
5.7 Rezultati ispitivanja komponenti	52
5.8 Inicijalizacija WebDriver-a	53
5.9 Ispitni slučaj - prijava	54
5.10 Ispitni slučaj - odjava	55
5.11 Ispitni slučaj - registracija	56
5.12 Ispitni slučaj - validacija polja tijekom registracije	57
5.13 Rezultati ispitivanja sustava	57
5.14 Dijagram razmještaja	58
6.1 Pregled promjena - <i>backend</i>	69
6.2 Pregled promjena - <i>frontend</i>	69

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 14. listopada 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - upoznavanje članova
 - raspravljanje o podjeli posla
 - inicijalni "brainstorming"

2. sastanak

- Datum: 23. listopada 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - inicijalni oblik baze podataka
 - razrađivanje projektnog zadatka

3. sastanak

- Datum: 30. listopada 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - raspored posla do idućeg sastanka
 - dizajn ekrana aplikacije - funkcionalost i *Frontend* dizajn

4. sastanak

- Datum: 08. studenog 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - raspored posla do "laboratorijske vježbe" demonstracije do sad napravljenog posla.

5. sastanak

- Datum: 15. studenog 2022.

- Prisustvovali: Svi članovi
- Teme sastanka:
 - pregled do sada napravljenog posla i preostalog posla
 - raspodjela posla do predaje projekta za kraj 1. ciklusa nastave

6. sastanak

- Datum: 10. prosinca 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - pregled i raspodjela glavnog posla za 2. ciklus
 - druženje unutar tima pomoću aplikacije Discord

7. sastanak

- Datum: 10. siječnja 2022.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - pregled napravljenog posla pred predaju

Tablica aktivnosti

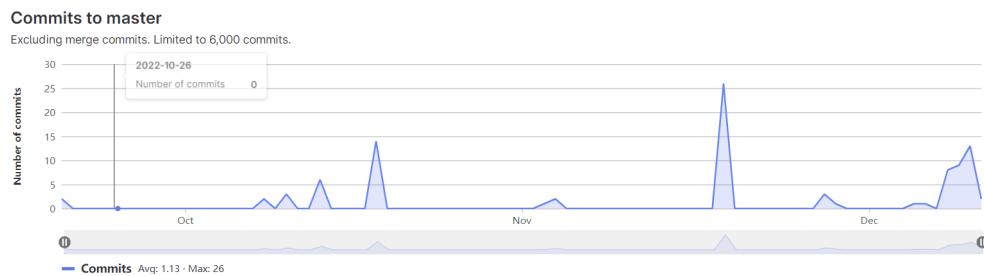
	Ian Golob	Matej Koščec	Edi Prodan	Zvonimir Žunić	Josip Goluža	Adrian Sušec	Ivan Kolar
Upravljanje projektom	4						
Opis projektnog zadatka							2
Funkcionalni zahtjevi							2
Opis pojedinih obrazaca							6
Dijagram obrazaca							2
Sekvencijski dijagrami							6
Opis ostalih zahtjeva							1
Arhitektura i dizajn sustava							5
Baza podataka	10						4
Dijagram razreda	3						1
Dijagram stanja							2
Dijagram aktivnosti							2
Dijagram komponenti							2
Korištene tehnologije i alati	1	0.5					2
Ispitivanje programskog rješenja	12						
Dijagram razmještaja							1
Upute za puštanje u pogon		1					
Dnevnik sastajanja						2	
Zaključak i budući rad							2
Popis literature							0.5

Nastavljeno na idućoj stranici

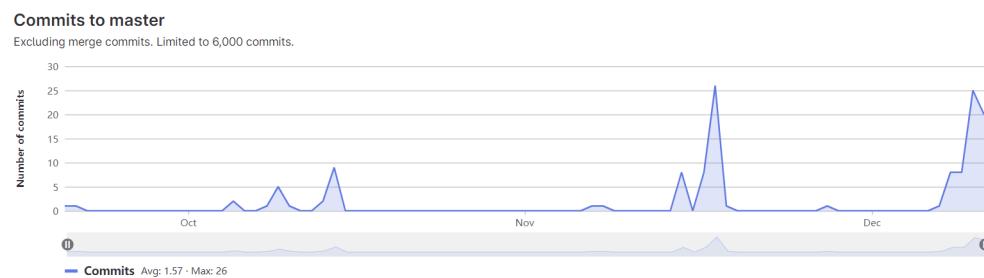
Nastavljeno od prethodne stranice

	Ian Golob	Matej Košćec	Edi Prodan	Zvonimir Žunić	Josip Goluža	Adrian Sušec	Ivan Kolar	
Izrada baze podataka	2	1						
Spajanje s bazom podataka	0.5							
Dizajn stranice	5	15	10	3		8		
<i>Backend</i>	58	65	20	9	26	35		
<i>Frontend</i>	5	60	55	50	40	50		
Puštanje aplikacije u pogon		25						

Dijagrami pregleda promjena



Slika 6.1: Pregled promjena - *backend*



Slika 6.2: Pregled promjena - *frontend*