

Programsko inženjerstvo

Ak. god. 2023./2024.

DentAll

Dokumentacija, Rev. 1.0

Grupa: Potplaćeni

Voditelj: Luka Kokić

Datum predaje: 17.11.2023.

Nastavnik: *Goran Rajić*

Sadržaj

| | |
|---|-----------|
| 1 Dnevnik promjena dokumentacije | 3 |
| 2 Opis projektnog zadatka | 5 |
| 3 Specifikacija programske potpore | 9 |
| 3.1 Funkcionalni zahtjevi | 9 |
| 3.1.1 Obrasci uporabe | 11 |
| 3.1.2 Sekvencijski dijagrami | 28 |
| 3.2 Ostali zahtjevi | 32 |
| 4 Arhitektura i dizajn sustava | 33 |
| 4.1 Baza podataka | 34 |
| 4.1.1 Opis tablica | 35 |
| 4.1.2 Dijagram baze podataka | 44 |
| 4.2 Dijagram razreda | 45 |
| 4.3 Dijagram stanja | 47 |
| 4.4 Dijagram aktivnosti | 48 |
| 4.5 Dijagram komponenti | 49 |
| 5 Implementacija i korisničko sučelje | 50 |
| 5.1 Korištene tehnologije i alati | 50 |
| 5.2 Ispitivanje programskog rješenja | 52 |
| 5.2.1 Ispitivanje komponenti | 52 |
| 5.2.2 Ispitni slučaj 1 - funkcionalnost prijave | 53 |
| 5.2.3 Ispitni slučaj 2 - funkcionalnost dodavanja novog adminis- tratora | 58 |
| 5.2.4 Ispitni slučaj 3 - funkcionalnost brisanja administratora | 67 |
| 5.2.5 Ispitni slučaj 4 - funkcionalnost dodavanja smještaja | 73 |
| 5.2.6 Ispitni slučaj 5 - funkcionalnost brisanja smještaja | 79 |
| 5.2.7 Ispitni slučaj 6 - funkcionalnost dohvaćanja posljednjeg une- senog realestateid-a | 84 |

| | | |
|----------|---|-----------|
| 5.3 | Dijagram razmještaja | 88 |
| 5.4 | Upute za puštanje u pogon | 89 |
| 6 | Zaključak i budući rad | 90 |
| | Popis literature | 91 |
| | Indeks slika i dijagrama | 93 |
| | Dodatak: Prikaz aktivnosti grupe | 94 |

1. Dnevnik promjena dokumentacije

| Rev. | Opis promjene/dodatka | Autori | Datum |
|------|--|-----------------------------|-------------|
| 0.1 | Napravljen predložak | Luka Kokić | 25.10.2023. |
| 0.2 | Opis projektnog zadatka + (partial) specifikacija programske potpore | Ian Marković | 31.10.2023. |
| 0.3 | Dodani opis arhitekture baze podataka kao i relacijska shema | Karlo Baljak | 31.10.2023. |
| 0.4 | Početna verzija obrazaca uporabe | Ian Marković, Teo Musa | 9.11.2023. |
| 0.5 | Konačna verzija obrazaca uporabe i ostali zahtjevi | Ian Marković, Teo Musa | 10.11.2023. |
| 0.6 | Početak dijagrama obrazaca uporabe | Mislav Matić | 11.11.2023. |
| 0.7 | Dijagrami razreda | Karlo Baljak | 11.11.2023. |
| 0.8 | Dovršeni dijagrami obrazaca uporabe | Mislav Matić, Luka Kokić | 15.11.2023. |
| 0.9 | Početak sekvencijskih dijagrama | Bruno Milaković | 15.11.2023. |
| 0.10 | Dovršeni sekvencijski dijagrami | Bruno Milaković, Luka Kokić | 16.11.2023. |
| 1.0 | Konačna verzija za prvu predaju | Luka Kokić | 17.11.2023. |
| 1.1 | Započeto poglavlje 5.1 i 6 | Luka Kokić | 16.1.2024. |
| 1.2 | Ažurirao dnevnik sastajanja | Luka Kokić | 16.1.2024. |

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

| Rev. | Opis promjene/dodatka | Autori | Datum |
|------|------------------------|--------------|------------|
| 1.3 | Napisano poglavlje 5.2 | Karlo Baljak | 17.1.2024. |

2. Opis projektnog zadatka

Cilj ovog projekta je razviti web aplikaciju „DentAll“ kojom će pružatelji usluga zdravstvenog turizma moći evidentirati i koordinirati lokalni smještaj i prijevoz korisnika usluga. Time bi se zdravstveni turizam učinio privlačnijim rješavanjem brige i cijene smještaja i prijevoza korisnika pri njihovom obitavanju u mjestu gdje koriste spomenute usluge. Uz to bi klijenti pružatelja usluga bili u mogućnosti unaprijed vidjeti geografsku sliku smještaja i okolice te moguće turističke opcije i kretanja tijekom privremenog obitavanja .

Aplikacija bi olakšala brige vlasnika usluga zdravstvenog turizma evidentiranjem, organizacijom i praćenjem smještaja i prijevoza klijenata uz programsku podršku interneta. Time se nastoji nadomjestiti nedostatak ovisnosti pružatelja zdravstvenih usluga o raznim i višestrukim pružateljima smještajnih usluga. Takvim pristupom kroz javnu i lagano dostupnu uslugu smanjio bi se broj posrednika u organizaciji i administraciji pružanja usluga zdravstvenog turizma. Također bi klijenti pronalazili sve potrebne informacije oko odabranog zdravstvenog turizma, tj. uz same zdravstvene usluge i o smještaju te prijevozu za svaki termin, čineći sam odabir prisustvovanja u inozemnim zdravstvenim uslugama jednostavniji i privlačniji.

Trenutna postojeća rješenja uključuju korištenje hotela i motela, što čini cijeli proces zdravstvenog turizma skupljim i kompliciranijim. Dugoročno bi pružateljima bilo isplativije iznajmljivati vlastite nekretnine u sklopu pružanja cjelokupnih usluga. Također je u trenutnim rješenjima prijevoz ostavljen na odgovornost klijenata, što otežava korištenje cijele usluge i njezinu privlačnost. Spajanjem oba problema u jedan sustav bi učinilo uslugu zdravstvenog turizma puno efektivnijom i jeftinijom, a time i više privlačnom mogućim budućim klijentima.

Postojeće vrste konkurentnih rješenja:

- javno dostupne informacije o načinima sudjelovanja u zdravstvenom turizmu i pripadnim članovima
- udruge sa svrhom spajanja svih dionika pružanja usluga zdravstvenog turizma

Primjer za usluge pružanja informacija je web stranica „MedicalTourism.com“ koja pruža korisnicima podatke o pružateljima zdravstvenih usluga, smještaja i mogućih tretmana koje spomenute usluge pružaju. Također sadrži i mnoge druge informacije kao vodiče za destinacije, usporedbe cijena i slično. No ipak je spomenuta web aplikacija napravljena za pružanje laganog pristupa svim potrebnim informacijama za moguće klijente, pružatelje zdravstvenih usluga, pružatelje smještaja te osiguravajuća društva na jednom mjestu te se time ne sukobljava do konkurentske razine sa svrhom kreiranja projektne aplikacije.

Dok je primjer druge vrste rješenja, čak konkurentnije ideji projekta, udruga Medical Tourism Association, koja spaja sve potrebne članove zdravstvenog turizma kroz program članstva. Udruga operira po cijelom svijetu i služi svrhu omogućavanja pružanja usluga zdravstvenog turizma kroz povezivanje potrebnih članova te svrhu informiranja javnosti o mogućnostima korištenja tih usluga. Također podržava edukaciju budućih pružatelja sa sveukupnim ciljem povećanja učestalosti zdravstvenog turizma u svijetu. Ovdje opisan projekt ipak sadrži određene funkcionalnosti koje nedostaju navedenoj konkurenciji, kao automatiziran proces organizacije i administracije, te lagano dostupni pregled informacija o obitavanju za klijente zdravstvenih usluga.



Slika 2.1: Mogućnosti povezivanja klijenata i pružatelja koje nudi web stranica "MedicalTourism.com"

Ciljani klijenti opisanih usluga su pružatelji usluga zdravstvenog turizma po cijelom svijetu, uključujući Hrvatsku i slične države sa jeftinom cijenom boravka. Optimalno bi bilo započeti pružanje usluga aplikacije najprije Europskim državama i okolici, a zatim, uz dovoljnu uspješnost i isplativost projekta, raširiti dostupnost usluge ostatku svijeta. Ciljani klijenti bi također bili i pružatelji lokalnih prijevoznih usluga, od privatnih firmi do javnih taksija, koje bi aplikacija spajala sa pružateljima zdravstvenih usluga za dogovor o prijevozu njihovih klijenata. Također bi, u manjoj perspektivi, evidencija većine korisnika zdravstvenog turizma na jednom mjestu olakšala statističke analize razvoja zdravstvenog turizma po cijelom svijetu.

U aplikaciji postoje tri uloge:

- smještajni administrator
- administrator prijevoznih usluga
- korisnički administrator

Jedan korisnik može imati više administratorskih uloga, dok uloga smještajnog administratora sadrži najveće ovlasti kojima mogu definirati druge korisnike i dodjeljivati uloge. Svaka uloga administratora sadrži određene posebne mogućnosti dodavanja i promjene informacija.

Smještajni administratori upravljaju podacima o smještaju; unose podatke o raspoloživom smještaju te definiraju smještajne kapacitete i unose ili brišu osnovne podatke o smještaju. Podatci smještaja se sastoje od tipa stana, kategorije opremljenosti, adrese i vremenskog perioda dostupnosti za korištenje. Uz svaki smještaj je dostupan grafički prikaz geografskog položaja uz Google Maps usluge.

Administratori prijevoznih usluga upravljaju podacima o prijevoznicima. To uključuje osnovne osobne i kontaktne podatke prijevoznika, vrstu i kapacitet prijevoznog sredstva te radno vrijeme raspoloživosti. Osnovni podatci prijevoznika se ne mogu mijenjati.

Korisnički administratori definiraju korisnike medicinskih usluga uz unos njihovih osnovnih podataka. Podatci korisnika se sastoje od imena, prezimena, kontaktnih podataka, vremena dolaska i odlaska u/iz zemlje te preferencije o veličini i kvaliteti smještaja. Detalji njihovih tretmana se preuzimaju iz vanjske aplikacije o evidenciji medicinskih usluga.

Aplikacija periodički provjerava status unesenih podataka i pridjeljuje adekvatni smještaj upisanim korisnicima te po zaključenju plana medicinskih usluga određenog korisnika im pridjeljuje prijevoznike od raspoloživih za svaki od termina. Sa završetkom spomenutog zaključenja sustav šalje poruku elektroničke pošte korisniku medicinskih usluga sa svim potrebnim informacijama te također šalje poruke elektroničke pošte prijevoznicima pridijeljenima terminima korisnika sa svim njima potrebnim informacijama.

Moguće dodatne funkcionalnosti za aplikaciju koje se mogu nadodati nakon izvršavanja jezgrenih funkcionalnosti su:

- prijavljivanje samog korisnika medicinskih usluga u aplikaciju čime mogu provjeriti osobne podatke, smještaj, prijevoznike i termine
- proširenje aplikacije u dodatnu svrhu prikaza općenitih podataka o medicinskom turizmu za privlačenje dodatnih klijenata
- mogućnost pridjeljivanja istog smještaja većem broju pacijenata u slučaju da je smještaj dovoljno velik.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Pružatelji zdravstvenih usluga (klinike)
2. Pružatelji prijevoznih usluga (prijevoznici)
3. Klijenti zdravstvenog turizma (pacijenti)
4. Korisnici (administratori)
 - a) administratori smještaja
 - b) administratori prijevoza
 - c) korisnički administratori
5. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neprijavljeni korisnik (inicijator) može:
 - (a) se prijaviti na postojeći korisnički račun upisivanjem korisničkog imena i lozinke
2. Administrator smještaja (inicijator) može:
 - (a) unijeti, modificirati i brisati podatke o smještaju
 - (b) vidjeti postojeće smještaje, njihove podatke i raspoloživost (uz grafički prikaz)
 - (c) unijeti i brisati podatke o prijavljenim klinikama
 - (d) vidjeti postojeće prijavljene klinike
 - (e) vidjeti postojeće korisnike
 - (f) registrirati nove korisnike te dodjeljivati uloge i brisati postojeće
3. Administrator prijevoznih usluga (inicijator) može:
 - (a) unijeti i brisati podatke o prijevoznicima
 - (b) modificirati podatke raspoloživosti prijevoznika
 - (c) vidjeti postojeće podatke prijevoznika i njihove raspoloživosti

4. Korisnički administrator (inicijator) može:

- (a) unijeti i brisati podatke pacijenata
- (b) vidjeti postojeće pacijente i njihove podatke
- (c) preuzeti i vidjeti detalje tretmana klinika u kontekstu određenog pacijenta

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o smještajima, raspoloživosti i smještajnim kapacitetima
- (c) pohranjuje sve podatke o prijevoznicima
- (d) pohranjuje sve podatke o pacijentima
- (e) pohranjuje sve podatke o dogovorenim terminima boravka pacijenta i prijevoza tijekom boravka

3.1.1 Obrasci uporabe

UC1 - Prijava u sustav

- **Glavni sudionik:** Neprijavljeni korisnik
- **Cilj:** Dobiti pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Postojanje korisničkog računa u bazi
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Sustav potvrđuje ispravnost unesenih podataka
 3. Sustav omogućava pristup funkcijama definirane ulogom korisničkog računa
- **Opis mogućih odstupanja:**
 - 2.a Neispravni podaci
 1. Aplikacija obavještava korisnika o neuspjeloj prijavi prikazivanjem poruke: "Incorrect username or password"

UC2 - Dodavanje novog korisnika

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Kreiranje i dodavanje novog korisnika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju "Add new user"
 2. Aplikacija ponuđuje formu za popunjavanje informacija o novom korisniku
 3. Korisnik unosi sve tražene podatke: osobne podatke (*PIN, name, surname, phone i e-mail*) i korisničko-specifične podatke (*username, password i role*)
 4. Sustav u bazi stvara novog korisnika sa predanim podacima
- **Opis mogućih odstupanja:**
 - 3.a Unos postojećeg *username*-a
 1. Aplikacija obavještava korisnika o zauzetosti ponuđenog *username*-a porukom „Username already in use“

3.b Krivi format danog osobnog identifikacijskog broja (*PIN*), broja mobitela (*phone number*) ili adrese elektroničke pošte (*e-mail*)

1. Aplikacija obavještava korisnika o neispravnom formatu i sustav ne ažurira pripadajuće podatke u bazi podataka

UC3 - Pregled korisnika

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Pregled postojećih korisnika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „View existing users“
 2. Aplikacija prikazuje podatke postojećih korisnika

UC4 - Modificiranje podataka korisnika

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Modificiranje podataka ciljanog korisnika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
 2. Baza sadrži podatke o ciljanom korisniku
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Modify user info“ pored podataka ciljanog korisnika
 2. Aplikacija ponuđuje formu za popunjavanje i izmjenjivanje informacija o korisniku
 3. Korisnik može izmijeniti osobne podatke (*phone number* i *e-mail*) i korisničko-specifične podatke (*password* i *role*)
 4. Sustav u bazi ažurira podatke ciljanog korisnika

UC5 - Brisanje postojeće korisnika

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Brisanje ciljanog korisnika
- **Sudionici:** Baza podataka
- **Preduvjeti:**

1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
2. Baza sadrži podatke o ciljanom korisniku
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Delete“ pokraj podataka ciljanog korisnika
 2. Korisnik potvrđuje odabir nakon upita aplikacije
 3. Sustav briše podatke odabranog korisnika iz baze
- **Opis mogućih odstupanja:**
 - 2.a Korisnik odustane od brisanja tijekom procesa brisanja
 1. Aplikacija obavještava korisnika o prekidu brisanja

UC6 - Dodavanje novog smještaja

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Kreiranje i dodavanje novog smještaja
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Add new accomodation“
 2. Sustav ponuđuje formu za popunjavanje informacija o novom smještaju
 3. Korisnik unosi tražene podatke: osnovne podatke (*address, latitude, longitude, accomodation type* i *equipment category*) i da li je smještaj raspoloživ (*active*)
 4. Sustav u bazi stvara novi smještaj sa predanim podacima
- **Opis mogućih odstupanja:**
 - 3.a Unos postojeće adrese (*address*) ili geografske pozicije (*latitude* i *longitude*)
 1. Aplikacija obavještava korisnika o već postojećim unesenim podacima u bazi

UC7 - Pregled smještaja

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Pregled unesenih smještaja
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
- **Opis osnovnog tijeka:**

1. Korisnik odabere opciju „View accomodations“
2. Aplikacija prikazuje podatke unesenih smještaja, uključujući prikaz geografske lokacije smještaja na karti

UC8 - Postavljanje raspoloživosti smještaja

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Postavljanje željene raspoloživosti smještaja
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik postavlja raspoloživost smještaja pomoću potvrdnog okvira
 2. Sustav u bazi ažurira raspoloživost smještaja

UC9 - Brisanje postojećeg smještaja

- **Glavni sudionik:** Smještajni korisnik
- **Cilj:** Brisanje ciljanog smještaja
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom smještajnog administratora
 2. Baza sadrži podatke o ciljanom smještaju
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Remove accomodation“ pokraj podataka ciljanog smještaja
 2. Korisnik potvrđuje odabir nakon upita aplikacije
 3. Sustav briše podatke odabranog smještaja iz baze
- **Opis mogućih odstupanja:**
 - 2.a Korisnik odustane od brisanja tijekom procesa brisanja
 1. Aplikacija obavještava korisnika o prekidu brisanja

UC10 - Dodavanje prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Kreiranje i dodavanje novog prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**

1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Add new transporter“
 2. Aplikacija ponuđuje formu za popunjavanje informacija o novom prijevozniku
 3. Korisnik unosi tražene osobne i kontaktne podatke prijevoznika (*phone number, organization name, address* i *active*)
 4. Sustav u bazu sprema podatke o prijevozniku
- **Opis mogućih odstupanja:**
 - 3.a Krivi format danog broja mobitela (*phone number*)
 1. Aplikacija obavještava korisnika o neispravnom formatu i ne ažurira pripadajuće podatke u bazi podataka

UC11 - Pregled prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Pregled unesenih prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „View transporters“
 2. Aplikacija prikazuje podatke unesenih prijevoznika

UC12 - Brisanje prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Brisanje ciljanog prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
 2. Baza sadrži podatke ciljanog prijevoznika
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Delete“ pokraj podataka ciljanog prijevoznika
 2. Korisnik potvrđuje odabir nakon upita aplikacije

3. Sustav briše podatke odabranog smještaja iz baze
- **Opis mogućih odstupanja:**
 - 2.a Korisnik odustane od brisanja tijekom procesa brisanja
 1. Aplikacija obavještava korisnika o prekidu brisanja

UC13 - Dodavanje vozila prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Kreiranje i dodavanje novog vozila prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Add new vehicle“ pod podacima ciljanog prijevoznika
 2. Aplikacija ponuđuje formu za popunjavanje informacija o novom vozilu
 3. Korisnik unosi tražene podatke o vozilu (*type of vehicle, capacity* i *active*)
 4. Sustav u bazi stvara novo vozilo i pridjeljuje ga ciljanom prijevozniku

UC14 - Pregled vozila prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Pregled unesenih vozila određenog prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „View assigned vehicles“
 2. Aplikacija prikazuje podatke unesenih vozila ciljanog prijevoznika

UC15 - Postavljanje raspoloživosti vozila prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Postavljanje raspoloživosti ciljanog vozila prijevoznik
- **Sudionici:** Baza podataka
- **Preduvjeti:**

1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Korisnik postavlja raspoloživost ciljanog vozila pomoću potvrdnog okvira
 2. Sustav u bazi ažurira raspoloživost ciljanog vozila

UC16 - Brisanje vozila prijevoznika

- **Glavni sudionik:** Administrator prijevoznih usluga
- **Cilj:** Brisanje ciljanog vozila prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom administratora prijevoznih usluga
 2. Baza sadrži podatke ciljanog vozila ciljanog prijevoznik
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Remove vehicle“ pod podacima ciljanog prijevoznika
 2. Korisnik potvrđuje odabir nakon upita aplikacije
 3. Sustav briše podatke odabranog vozila prijevoznika iz baze
- **Opis mogućih odstupanja:**
 - 2.a Korisnik odustane od brisanja tijekom procesa brisanja
 1. Aplikacija obavještava korisnika o prekidu brisanja

UC17 - Dodavanje pacijenta

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Kreiranje i dodavanje novog pacijenta
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Add new patient“
 2. Aplikacija ponuđuje formu za popunjavanje informacija o novom pacijentu
 3. Korisnik unosi tražene podatke: osobne i kontaktne podatke (*PIN, name, surname, phone, e-mail i residence address*) te podatke o tretmanu (*clinic i treatment*) i preferenciji smještaja (*accomodation preferences*)

4. Sustav u bazi stvara novog pacijenta sa predanim podacima
- **Opis mogućih odstupanja:**
 - 3.a Krivi format danog osobnog identifikacijskog broja (*PIN*), broja mobitela (*phone number*) ili adrese elektroničke pošte (*e-mail*)
 1. Aplikacija obavještava korisnika o neispravnom formatu i ne ažurira pripadajuće podatke u bazi podataka

UC18 - Pregled pacijenata

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Pregled postojećih pacijenata
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „View patients“
 2. Aplikacija prikazuje podatke postojećih pacijenata, uključujući da li im je pridijeljen smještaj i prijevoz, te podacima o smještaju i prijevozu u slučaju da jesu pridijeljeni

UC19 - Dohvaćanje informacija o tretmanu

- **Glavni sudionik:** Sustav
- **Cilj:** Dohvaćanje informacija o tretmanu pacijenta pri registraciji/dodavanju pacijenta
- **Sudionici:** Baza podataka, baza podataka pacijentove klinike
- **Preduvjeti:**
 1. Baza sadrži sve potrebne podatke o pacijentu
 2. Baza podataka klinike sadrži potrebne podatke o tretmanu pacijenta
- **Opis osnovnog tijeka:**
 1. Sustav uspostavlja vezu sa bazom podataka klinike
 2. Sustav dohvaća podatke o tretmanu iz baze podataka klinike
 3. Sustav sprema dohvaćene podatke u bazu podataka sustava
- **Opis mogućih odstupanja:**
 - 1.a Sustav ne može uspostaviti vezu sa klinikom
 1. Aplikacija obavještava korisnika o neuspješnom povezivanju

UC20 - Brisanje pacijenta

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Brisanje ciljanog vozila prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Korisnik je prijavljen na račun sa ulogom korisničkog administratora
 2. Baza sadrži podatke ciljanog pacijenta
- **Opis osnovnog tijeka:**
 1. Korisnik odabere opciju „Delete“ pokraj podataka ciljanog pacijenta
 2. Korisnik potvrđuje odabir nakon upita aplikacije
 3. Sustav briše podatke odabranog pacijenta iz baze
- **Opis mogućih odstupanja:**
 - 2.a Korisnik odustane od brisanja tijekom procesa
 1. Aplikacija obavještava korisnika o prekidu brisanja

UC21 - Periodičko pridiyeljivanje smještaja i prijevoza pacijentima

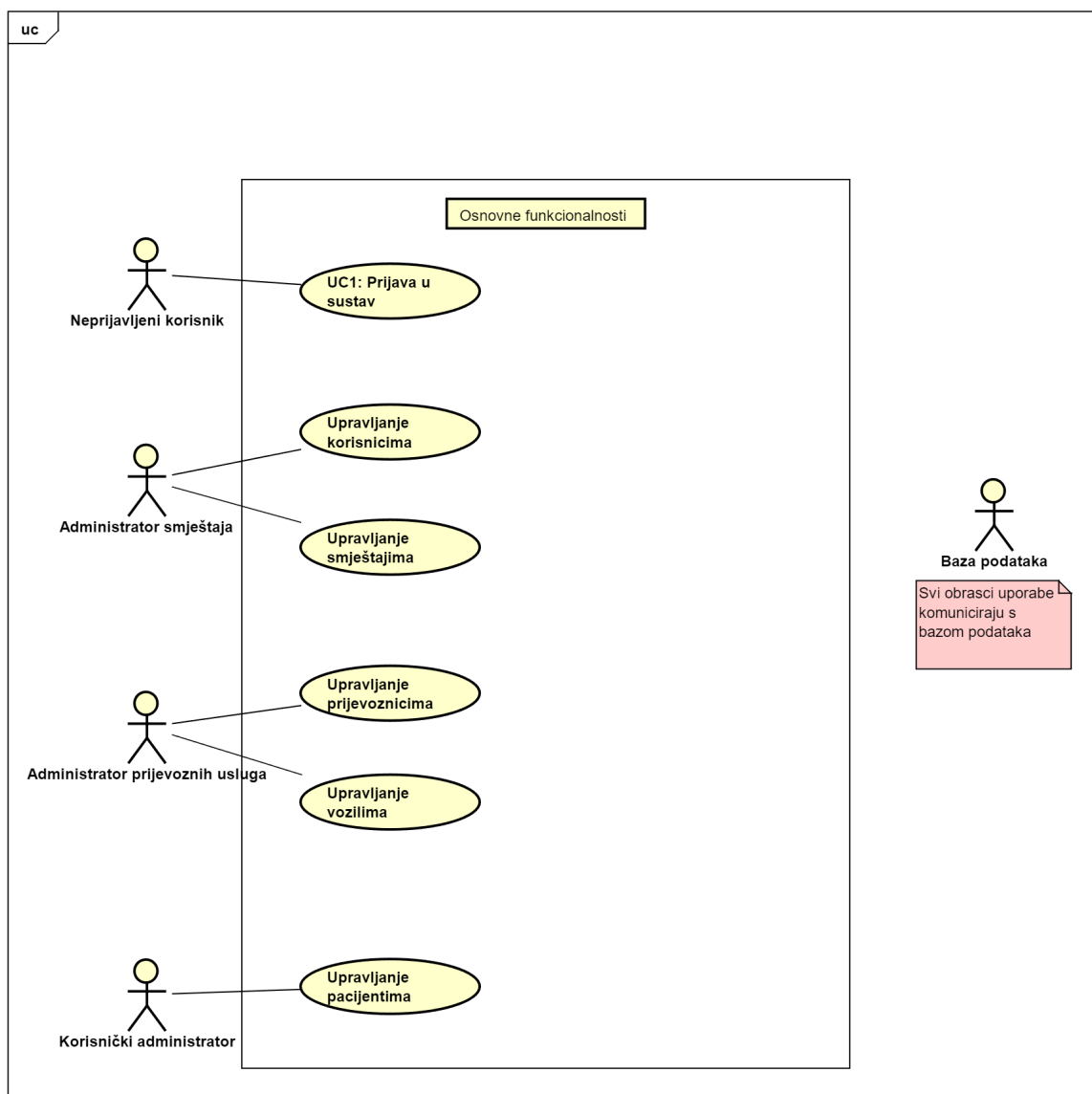
- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Pridiyeljivanje adekvatnog smještaja i prijevoza upisanim pacijentima te obavještavanje pacijenta o uspješnom pridiyeljenju
- **Sudionici:** Baza podataka
- **Preduvjeti:**
 1. Baza sadrži potrebne podatke o pacijentima, smještajima i prijevoznima
 2. Baza sadrži barem jednu instancu svakog potrebnog podatka
- **Opis osnovnog tijeka:**
 1. Sustav provjerava unesene podatke za pacijente bez smještaja
 2. Sustav (ako je moguće) pridiyeljuje adekvatni smještaj pacijentu
 3. Sustav (ako je moguće) pridiyeljuje adekvatni prijevoz pacijentu pri zaključenju medicinskog plana
- **Opis mogućih odstupanja:**
 - 2.a Sustav ne može naći adekvatni smještaj i/ili prijevoz pacijentu
 1. Sustav periodički pokušava ponovno pri unosu novih podataka

UC22 - Obavještavanje klijenata o uspješnom zaključenju plana

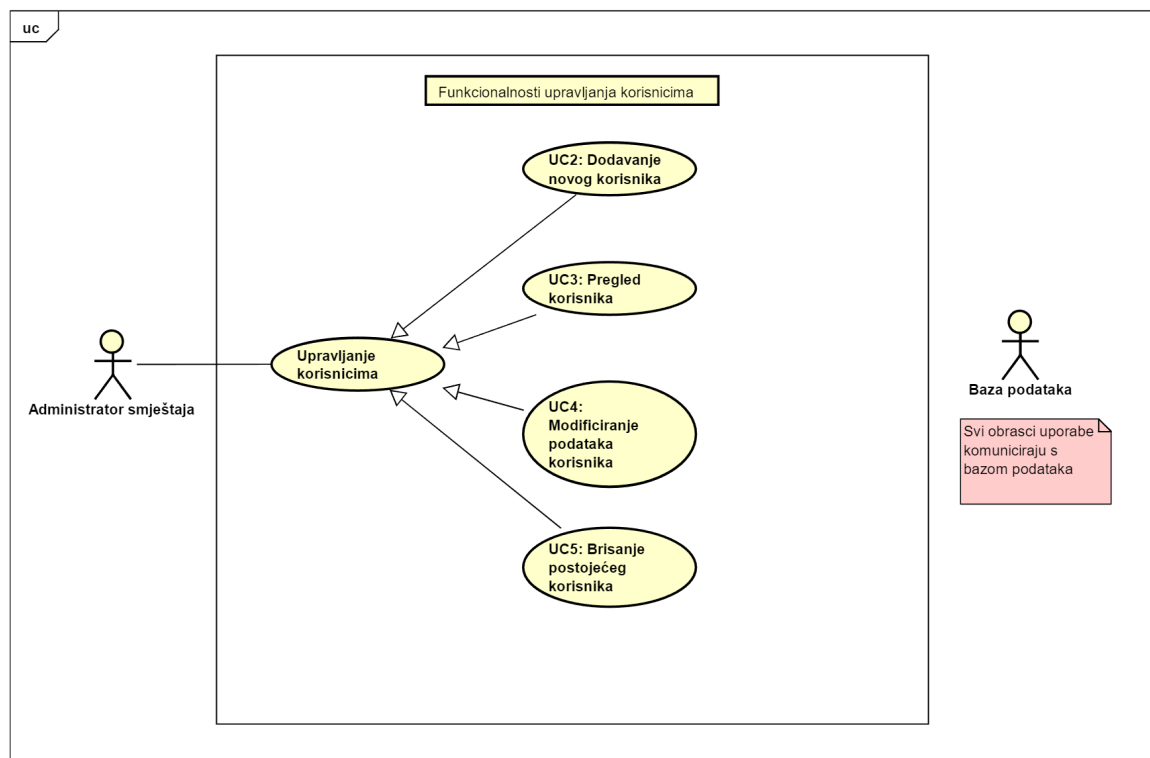
- **Glavni sudionik:** Sustav
- **Cilj:** Obavještavanje pacijenta, prijevoznika o uspješno zaključenom planu medicinskih usluga

- **Sudionici:** Baza podataka, pacijent, prijevoznik
- **Preduvjeti:**
 1. Baza sadrži potrebne podatke o pacijentima, smještajima i prijevoznima
 2. Sustav je uspješno zaključio medicinski plan pacijenta pridijeljivanjem smještaja i prijevoza pacijentu
- **Opis osnovnog tijeka:**
 1. Sustav provjerava zaključen plan te primjereno modificira podatke u bazi podataka (raspoloživost smještaja i vozila prijevoznika)
 2. Aplikacija šalje mail pacijentu sa podacima o datumu plana korištenja tretmana te smještaju i prijevozu tijekom plana tretmana
 3. Aplikacija šalje mail prijevozniku o zauzetosti vozila koje će se koristiti za pridijeljeni medicinski plan

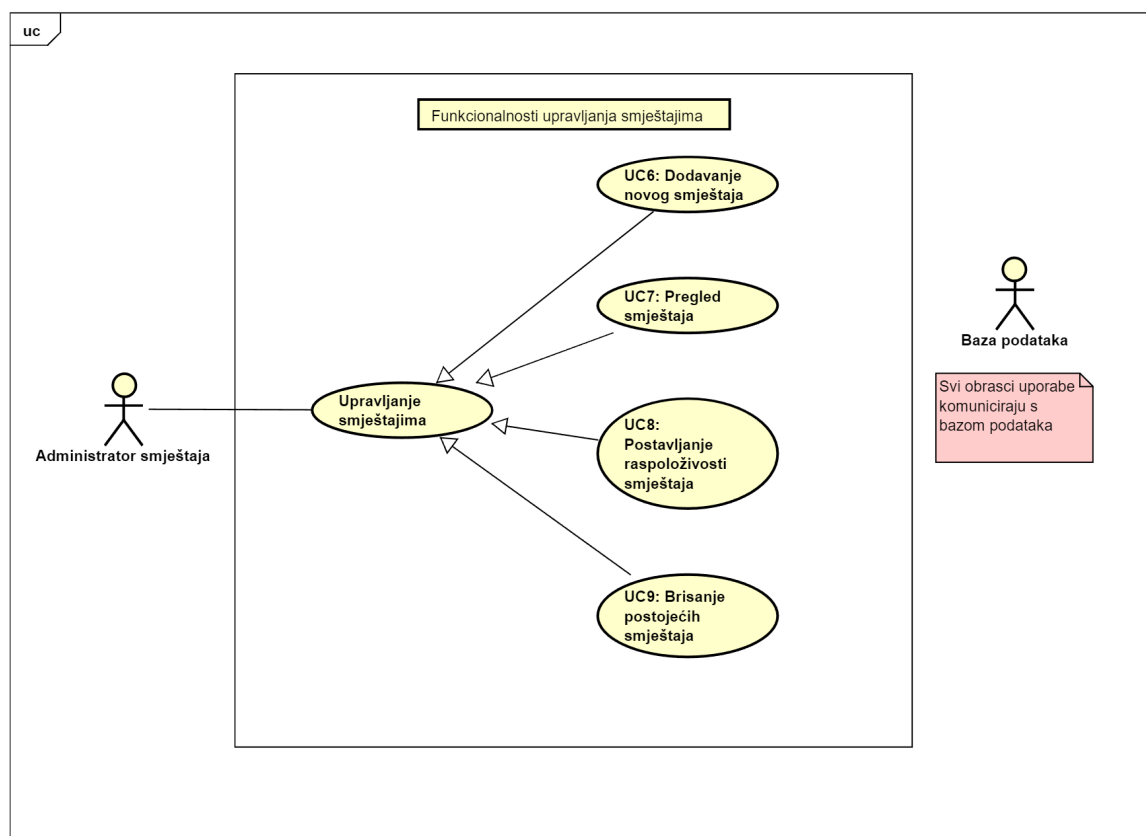
Dijagrami obrazaca uporabe



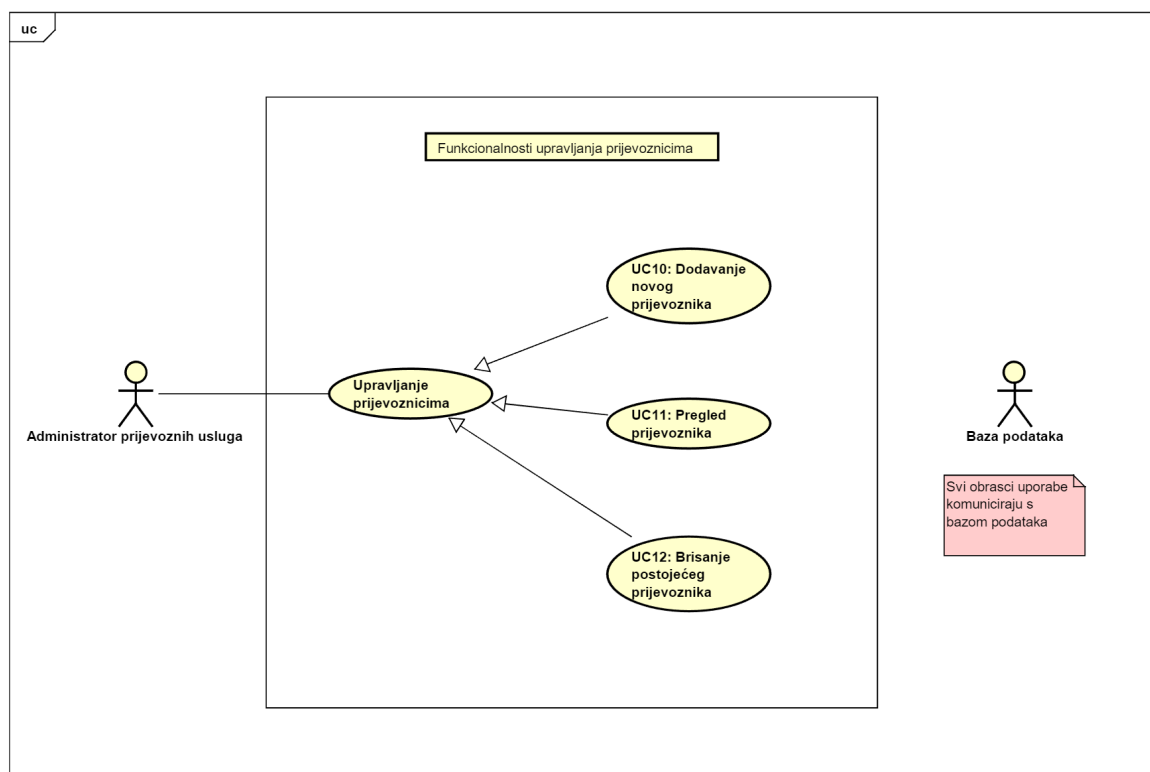
Slika 3.1: Dijagram obrasca uporabe, osnovne funkcionalnosti



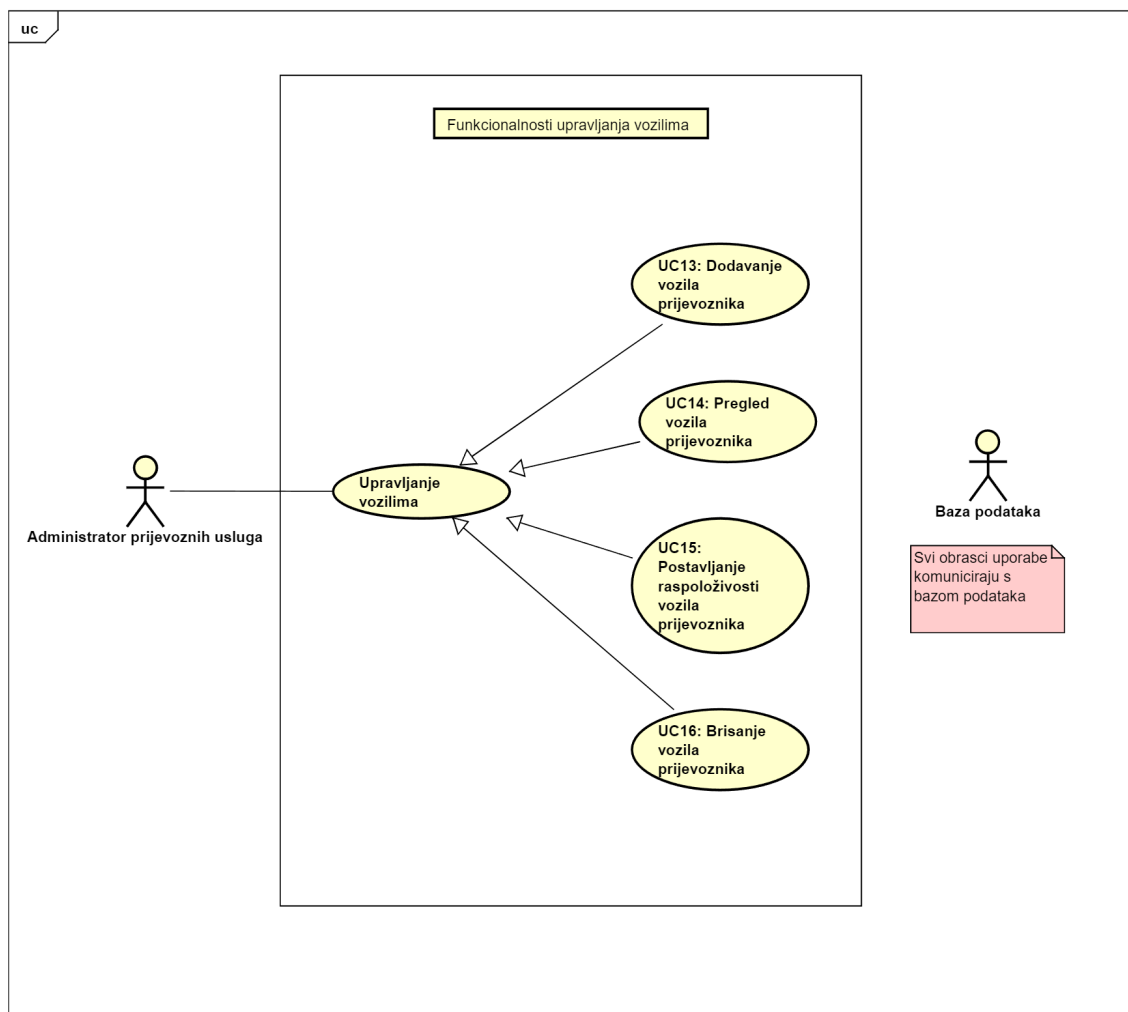
Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti upravljanja korisnicima



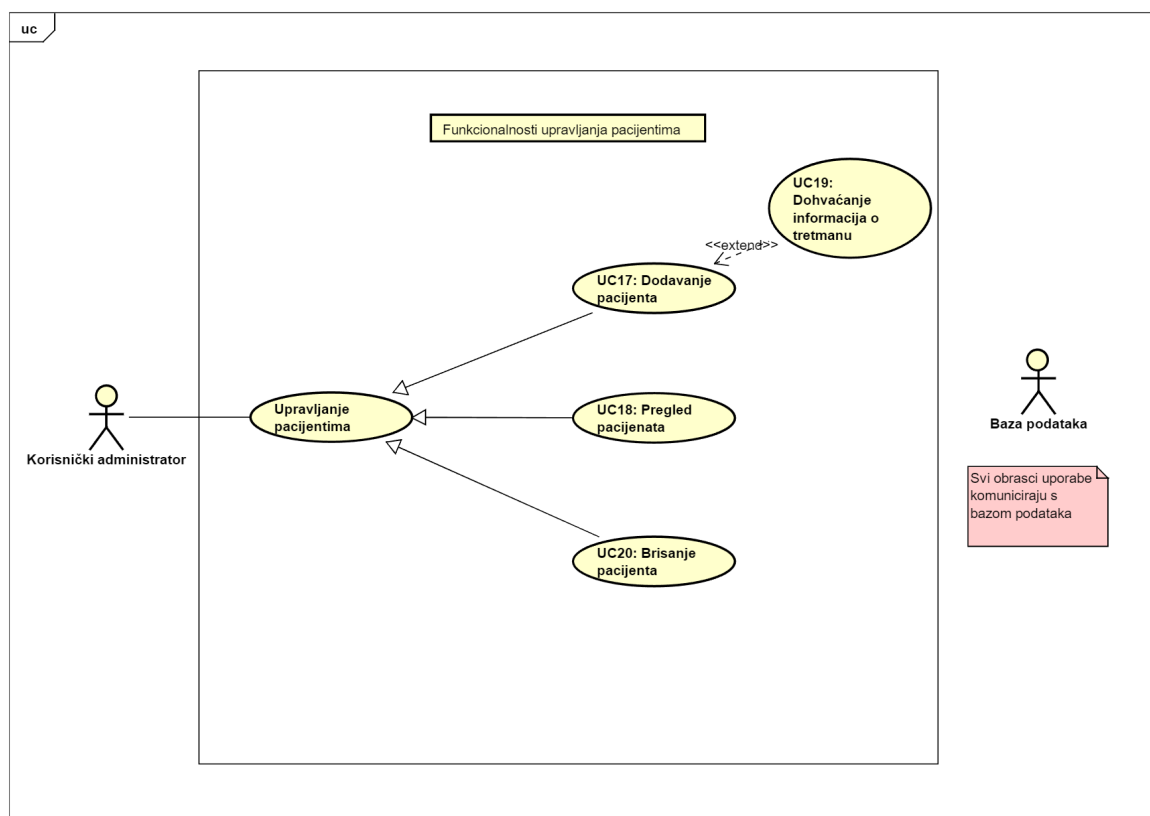
Slika 3.3: Dijagram obrasca uporabe, funkcionalnosti upravljanja smještajima



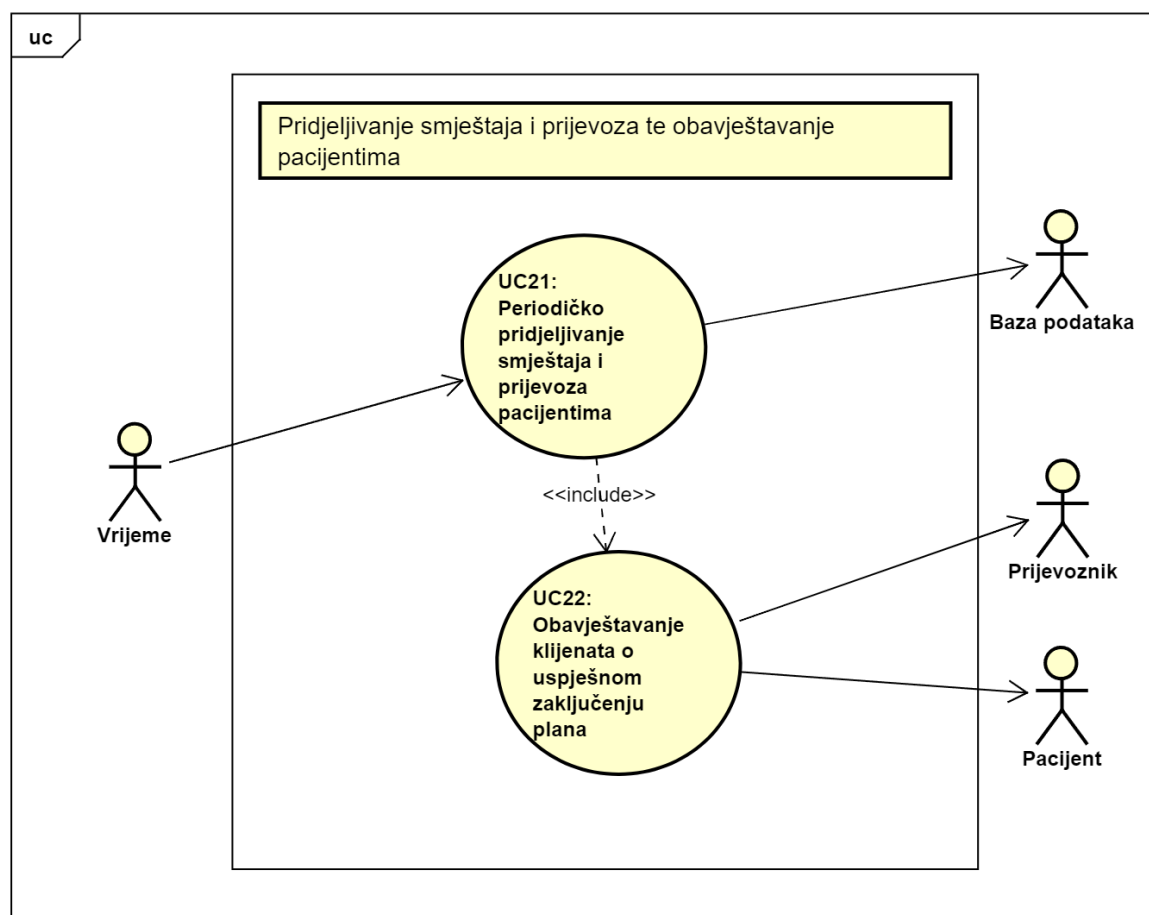
Slika 3.4: Dijagram obrasca uporabe, funkcionalnosti upravljanja prijevoznicima



Slika 3.5: Dijagram obrasca uporabe, funkcionalnosti upravljanja vozilima



Slika 3.6: Dijagram obrasca uporabe, funkcionalnosti upravljanja pacijentima

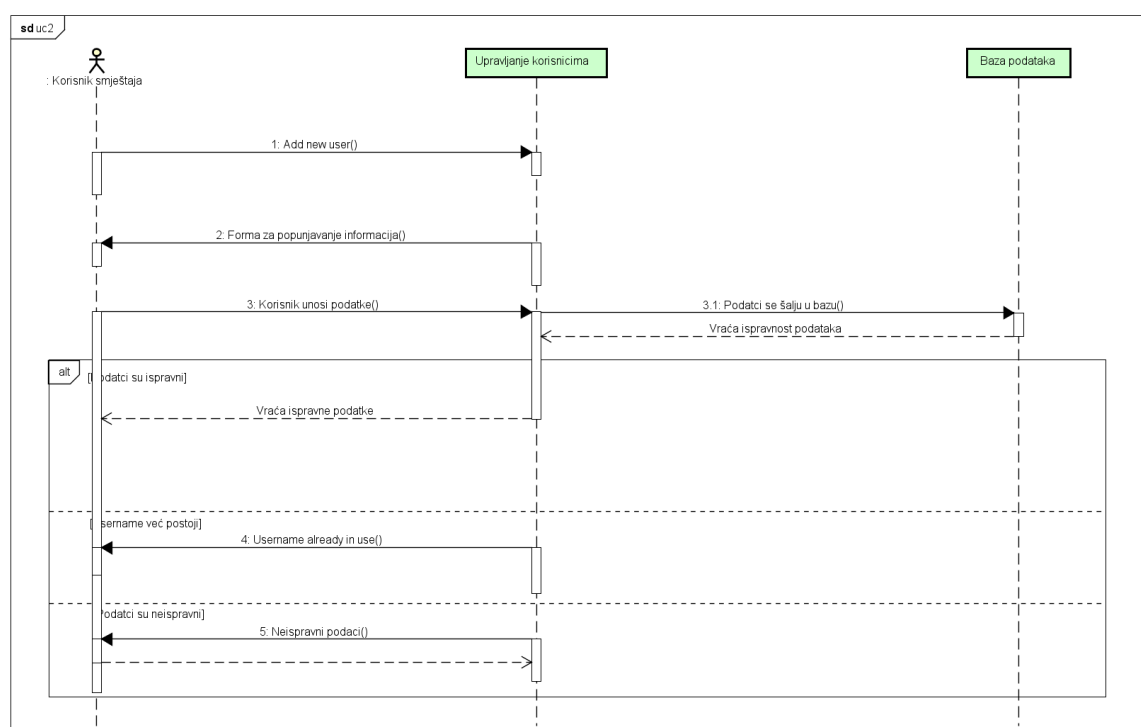


Slika 3.7: Dijagram obrasca uporabe, pridjeljivanje i obavještavanje

3.1.2 Sekvencijski dijagrami

Obrazac uporabe 2 - Dodavanje novog korisnika

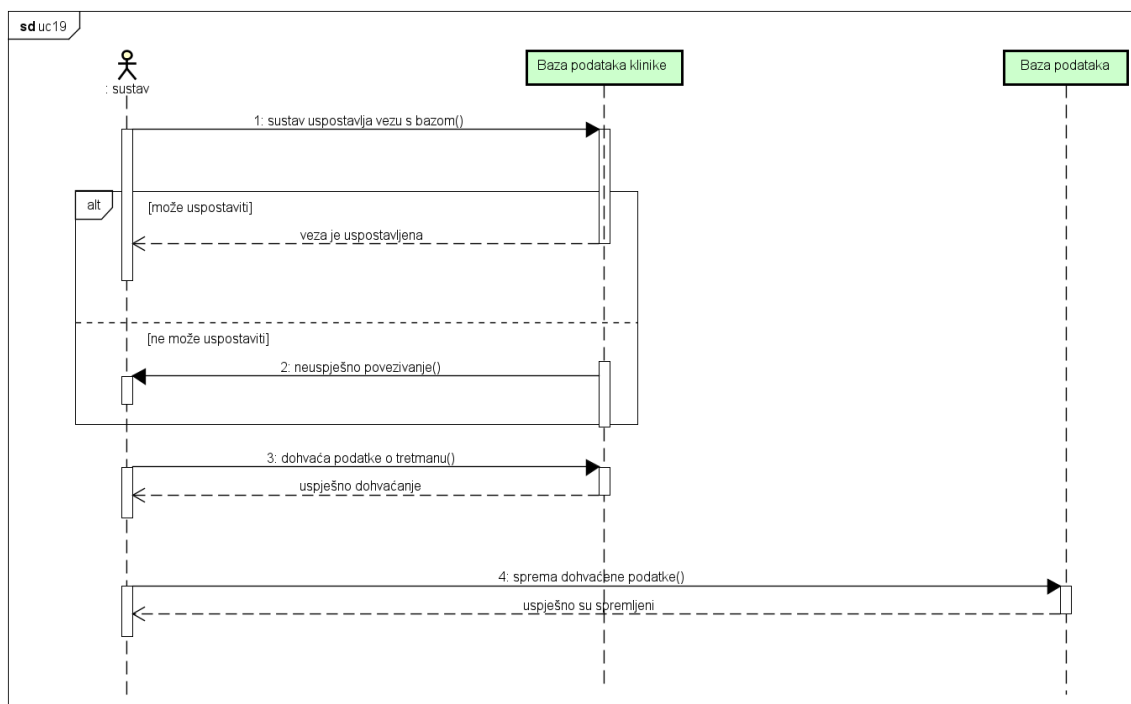
Kada korisnik koji ima ovlasti smještajnog administratora klikne na gumb "Add new user", pojavi mu se forma za ispunjavanje informacija o novom korisniku kojeg želi napraviti. Nakon što su podaci uneseni, oni se pošalju u bazu podataka koja potom javlja rezultat dodavanja novog korisnika u sustav, tj. ako nema nikakvih problema, novi korisnik je uspješno dodan u bazu podataka, no može doći do greške poput unošenja imena već postojećeg korisnika ili do unesenih neispravnih podataka kao što je krivi format e-mail-a, broja mobitela ili PIN-a.



Slika 3.8: Sekvencijski dijagram za UC2

Obrazac uporabe 19 - Dohvaćanje informacija o tretmanu

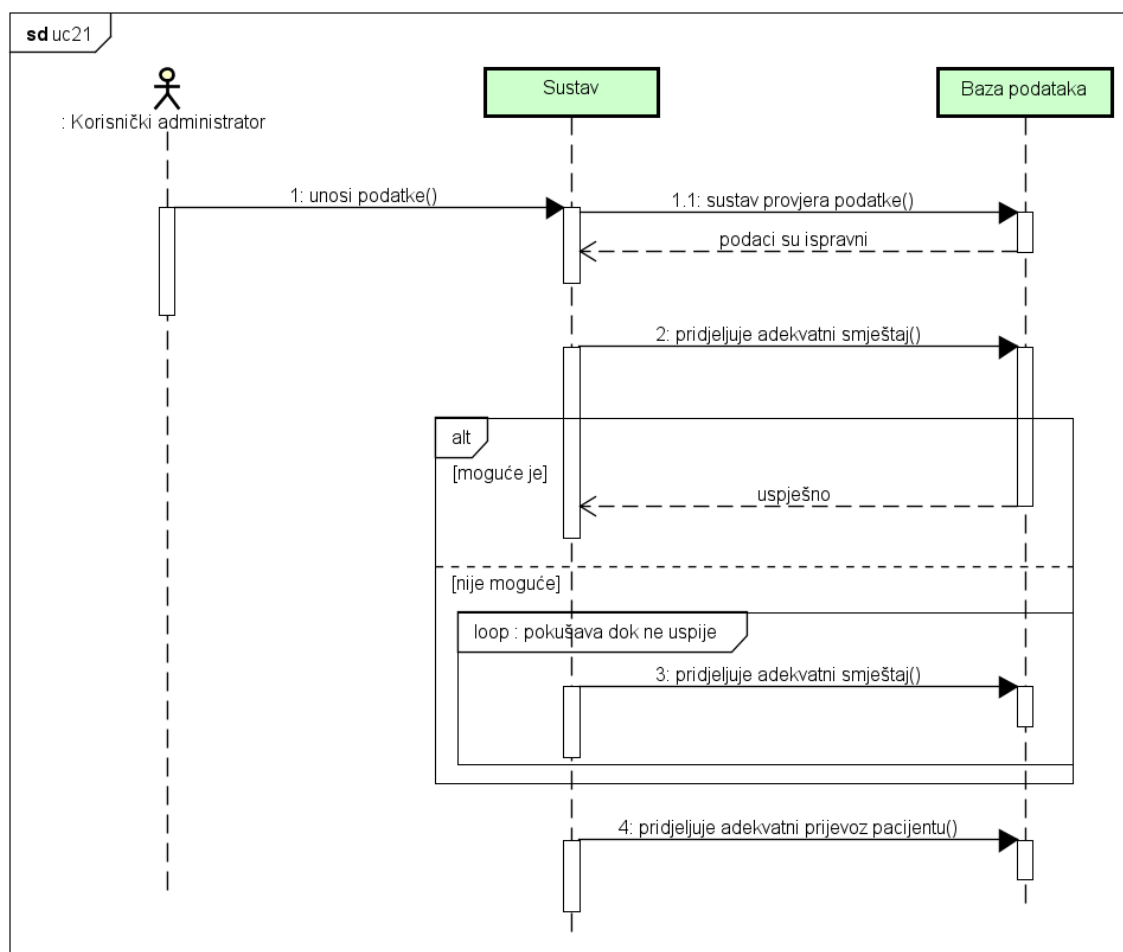
Nakon što je dodan novi pacijent, možemo iz vanjske baze podataka povući podatke o njegovom tretmanu. Prvotno sustav pokušava uspostaviti kontakt s vanjskom bazom podataka nakon čega može doći do dva slučaja: da se veza uspješno uspostavi, ili da se iz nekog razloga (npr. nedostatak interneta) ne uspije uspostaviti. Ako sustav uspije uspostaviti vezu, onda povuče tražene podatke te ih potom spremi u vlastitu bazu podataka.



Slika 3.9: Sekvencijski dijagram za UC19

Obrazac uporabe 21 - Periodičko pridjeljivanje smještaja i prijevoza pacijentima

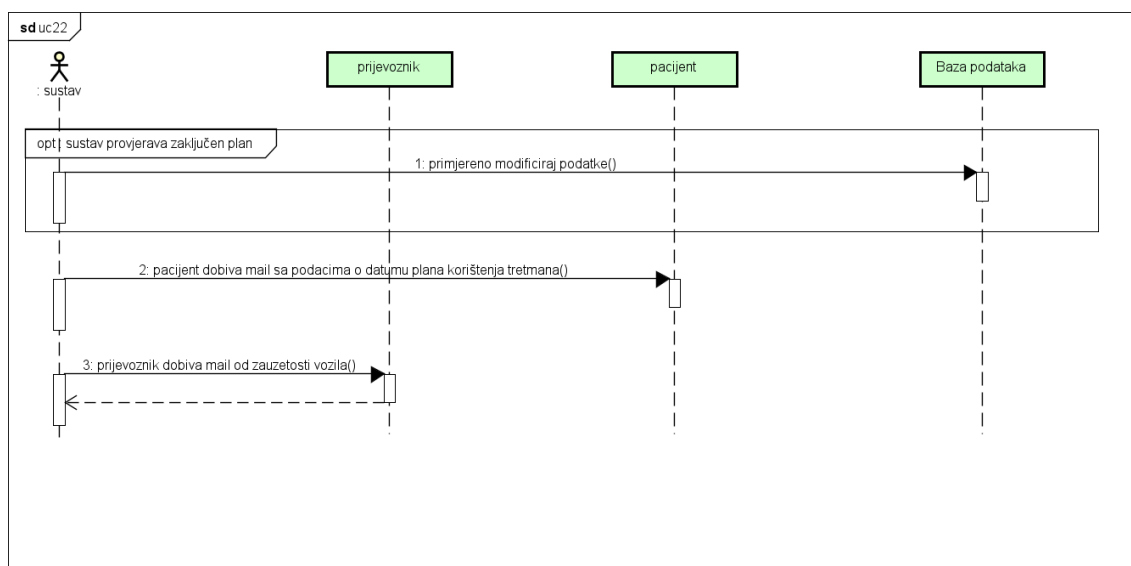
Svako malo sustav će pokušati dodijeliti postojećim pacijentima u bazi podataka koji još nemaju dodijeljen smještaj adekvatni smještaj. Ako sustav uspije pronaći adekvatan smještaj za pacijenta, onda se pacijentu dodijeli taj smještaj, a ako ne onda će se to ponovno pokušati nakon nekog vremena sve dok se ne uspije.



Slika 3.10: Sekvencijski dijagram za UC21

Obrazac uporabe 22 - Obavještavanje klijenata o uspješnom zaključenju plana

Nakon što se plan za nekog pacijenta zaključi, on se provjeri te se mijenjaju podaci u bazi podataka u vezi raspoloživosti smještaja i vozila prijevoznika. Potom se pacijentu šalje mail u kojemu ga se obavještava o datumu tretmana te smještaju i prijevozu tijekom plana tretmana te se prijevozniku šalje mail o zauzetosti vozila koje će se koristiti za pridjeljeni medicinski plan.



Slika 3.11: Sekvencijski dijagram za UC22

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Aplikacija treba biti jednostavna za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Aplikacija kao valutu koristi EUR
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS.

4. Arhitektura i dizajn sustava

Arhitektura sustava može se podijeliti na tri glavna podsustava: web preglednik, web poslužitelj i baza podataka.

- **Web preglednik** – program koji služi za pristupanje web stranicama. Puteš web preglednika korisnik komunicira sa poslužiteljem koristeći princip zahtjev-odgovor ili šalje podatke (najčešće u obliku obrazaca). Daljnja zadaća web preglednika je osigurati da se traženi podaci ispravno prikazuju ili da se ispravno prosljeđuju i spremaju na poslužitelj.
- **Web poslužitelj** – glavni je dio web aplikacije. To je most koji povezuje korisnika i bazu podataka koji se temelji na protokolu HTTP. Na zahtjeve korisnika dohvaća tražene podatke (resurse) ili obrađuje i sprema poslane podatke od korisnika.
- **Baza podataka** – srce je sustava. U njoj su pospremljeni svi podaci. Skoro ne postoji sustav u kojem nema komunikacije između aplikacije i baze.

Aplikacije je izgrađena na modelu MVC (Model – View - Controller) softverske arhitekture uz male modifikacije. Controller dio strukture je ostvaren tako što je integriran unutar same baze, tj. funkcije koje manipuliraju podacima se nalaze unutar baze. Shodno tome aplikaciju onda dijelimo na tri komponente:

- **Model** – glavna komponenta sustava. Reprezentira strukturu podataka.
- **View** – komponenta zaslužna za reprezentaciju podataka.
- **Controller** – komponenta koja odrađuje svu logiku i komunikaciju između sučelja i baze.

Backend naše aplikacije je ostvaren direktno u bazi PostgreSQL(razvojno sučelje pgadmin) za što koristimo API napisan u Node.js frameworku Express, koji služi kao middleware. Za izradu frontend-a korišten je React uz pomoć Material UI. Razvojno okruženje koje smo koristili je bilo Visual Studio Code.

4.1 Baza podataka

U ovom projektu koristit ćemo relacijsku bazu podataka, čije su osnovne jedinice entiteti, definirani imenom i skupom atributa. Osnovna zadaća baze podataka je pohrana podataka te brza i efikasna obrada tih podataka u ovisnosti i korisničkim zahtjevima. U bazi podataka su pohranjeni podaci o korisnicima, njihovim ulogama, preferencijama, kao i o smještaju te dostupnosti smještaja. Dodatno uz navedeno, zbog zahtjeva organizacije prijevoza, baza također sprema informacije o vrstama vozila i dostupnosti tih vozila kao i o vremenu i mjestu boravka korisnika zdravstvenog turizma. Tako su i definirani sljedeći entiteti:

- Clinic
- Town
- AdminUser
- Credentials
- UserRole
- Patient
- Treatment
- PatientPreferences
- PatientArrival
- Accomodation
- AccomodationType
- Equipped
- AccomodationOccupied
- Transporter
- Vehicle
- VehicleType
- VehicleOccupied
- VehicleSchedule

4.1.1 Opis tablica

Clinic Ovaj entitet sadrži podatke o klinikama u odabranoj zemlji. Atributi su: ClinicID (primary key), clinicName, latitude, longitude, clinicAddress, TownID (foreign key). Ovaj je entitet u vezi Many-to-One sa entitetom Town preko atributa TownID, u Many-to-Many vazi sa entitetom Accomodation preko atributa AccomodationID i ClinicID, u Many-to-Many vezi sa entitetom Transporter preko atributa TransporterID i ClinicID, u One-to-Many vezi sa entitetom Patient preko atributa PatientID i ClinicID te u Many-to-Many vezi sa entitetom Treatment preko atributa TreatmentID.

| Clinic | | |
|---------------|-----------|---|
| ClinicID | BIGSERIAL | Jedinstveni brožčani identifikator klinike,automatski generiran |
| clinicName | VARCHAR | Ime klinike |
| Latitude | DECIMAL | Geografska širina |
| Longitude | DECIMAL | Geografska dužina |
| clinicAddress | VARCHAR | Adresa klinike |
| TownID | INT | Grad u kojem se klinika nalazi |

Town Ovaj entitet sadrži podatke o gradovima u kojima se nalaze klinike u koje dolaze pacijenti na liječenje. Atributi su: TownID (primary key) i townName. Ovaj entitet je u vezi One-to-Many sa entitetom Clinic preko atributa TownID i u vezi One-to-Many sa entitetom Accomodation preko atributa TownID i u Many-to-Many vezi sa entitetom Transporter preko atributa TownID.

| Town | | |
|----------|-----------|--|
| TownID | BIGSERIAL | Jedinstveni brožčani identifikator grada, automatski generiran |
| townName | VARCHAR | Ime grada |

AdminUser Ovaj entitet sadrži podatke o korisnicima aplikacije; svi administratori i oni koji mogu dodavati ili ažurirati ili brisati podatke iz baze. Atributi su: UserID (primary key), PIN(personal identification number), firstname, lastname, phone, email. Ovaj je entitet u vezi Many-to-Many sa entitetom UserRole preko atributa RoleID te u vezi One-to-One sa entitetom Credentials preko atributa UserID.

| AdminUser | | |
|-----------|-----------|--|
| UserID | BIGSERIAL | Jedinstveni brojčani identifikator korisnika, automatski generiran |
| PIN | INT | Identifikacijski broj korisnika |
| firstname | VARCHAR | Ime korisnika |
| lastname | VARCHAR | Prezime korisnika |
| phone | VARCHAR | Broj mobitela korisnika |
| email | VARCHAR | Elektronička pošta korisnika |

Credentials Ovaj entitet sadrži podatke o korisničkim računima administratora. Atributi su: UserID (foreign key), username i pass. Ovaj je entitet u One-to-One vezi sa entitetom AdminUser preko atributa UserID.

| Credentials | | |
|-------------|-----------|---|
| UserID | BIGSERIAL | ID korisnika kojem pripadaju korisničko ime i lozinka |
| username | VARCHAR | Jedinstveno korisničko ime |
| pass | BINARY | Lozinka korisnika za prijavu u aplikaciju |

UserRole Ovaj entitet sadrži podatke o ulogama koje postoje u sustavu. Atributi su: RoleID i roleName. Ovaj je entitet u Many-to-One vezi sa entitetom UserAdmin preko atributa UserID.

| UserRole | | |
|----------|----------|--|
| RoleID | SMALLINT | Jedinstveni indentifikator uloge, automatski generiran |
| roleName | VARCHAR | ime uloge |

Patient Ovaj entitet sadrži podatke o korisnicima zdravstvenog turizma. Atributi su: PatientID (primary key), PIN (personal identification number), firstname, lastname, phone, email, residenceAddress. Ovaj je entitet u vezi One-to-One sa entitetom Accomodation preko atributa AccomodationID, u vezi One-to-Many vezi sa entitetom Treatment preko atributa TreatmentID te u One-to-Many vezi sa entitetom Clinic preko atributa ClinicID i PatientID.

| Patient | | |
|------------------|-----------|--|
| PatientID | BIGSERIAL | Jedinstveni brojčani identifikator pacijenta, automatski generiran |
| PIN | INT | Identifikacijski broj pacijenta |
| firstname | VARCHAR | Ime pacijenta |
| lastname | VARCHAR | Prezime pacijenta |
| phone | VARCHAR | Broj mobitela pacijenta |
| email | VARCHAR | Elektronička pošta pacijenta |
| residenceAddress | VARCHAR | Mjesto prebivališta pacijenta |

Treatment Ovaj entitet sadrži podatke o tretmanima. Atributi su: TreatmentID (primary key) i description. Ovaj je entite u Many-to-One vezi sa entitetom Patient preko atributa TreatmentID i PatientID te u One-to-Many vezi sa entitetom Clinic preko atributa TreatmentID i ClinicID.

| Treatment | | |
|-------------|-----------|---|
| TreatmentID | BIGSERIAL | Jedinstveni indentifikator tretmana, automatski generiran |
| description | text | opsi tretmana |

PatientPreferences Ovaj entitet sadrži podatke o preferencijama pacijenta vezanih za smještaj. Atributi su: PatientID (foreign key), TypeID (foreign key), EquippedID (foreign key). Ovaj je entite u Many-to-One vezi sa entitetom Patient preko atributa PatientID, u One-to-One vezi sa entitetom AccomodationType preko atributa AccomodationID te u One-to-One vezi sa entitetom Equipped preko atributa EquippedID.

| PatientPreferences | | |
|--------------------|-----------|--|
| PatientID | BIGSERIAL | Jedinstveni indentifikator pacijenta, automatski generiran |
| TypeID | SMALLINT | Jedinstveni indentifikator vrste smještaja, automatski generiran |
| EquippedID | SMALLINT | Jedinstveni indentifikator opremljenosti smještaja, automatski generiran |

PatientArrival Ovaj entitet sadrži podatke o vremenu dolaska/odlaska pacijenta te također i gradu u kojem se liječi. Atributi su: ArrivalID(primary key), PatientID (foreign key), TownID (foreign key), dateOfArrival, dateOfDeparture. Ovaj je entite u One-to-One vezi sa entitetom Patient preko atributa PatientID, u One-to-One vezi sa entitetom Town preko atributa TownID.

| PatientArrival | | |
|-----------------|-----------|---|
| ArrivalID | BIGSERIAL | Jedinstveni identifikator rekorda dolaska/odlaska pacijenta |
| PatientID | BIGSERIAL | Jedinstveni indentifikator pacijenta |
| TownID | SMALLINT | Grad u koji pacijent dolazi |
| dateOfArrival | DATETIME | Vrijeme i datum dolaska pacijenta |
| dateOfDeparture | DATETIME | Vrijeme i datum odlaska pacijenta |

Accommodation Ovaj entitet sadrži podatke o smještaju, vrsti smještaja, njegovoj opremljenosti te adresi na kojoj se nalazi kao i koordinatama. Atributi su: AccomodationID (primary key),TypeID (foreign key), EquippedID (foreign key), latitude, longitude, address, TownID (foreign key), activem dateAdded. Ovaj je entitet u vezi Many-to-One sa entitetom Town preko atributa TownID, nadalje u vezi Many-to-One sa entitetom AccomodationType preko atributa TypeID, u vezi Many-to-One sa entitetom Equipped preko atributa EquippedID, u vezi One-to-One sa entitetom Patient preko atributa AccomodationID i PatientID.

| Accommodation | | |
|----------------|-----------|--|
| AccomodationID | BIGSERIAL | Jedinstveni brojčani identifikator smještaja, automatski generiran |
| TypeID | INT | ID vrste smještaja |
| EquippedID | INT | ID opremljenosti smještaja |
| latitude | DECIMAL | Geografska širina |
| longitude | DECIMAL | Geografska dužina |
| address | VARCHAR | Adresa smještaja |
| TownID | INT | ID grada u kojem se smještaj nalazi |
| active | BIT | Je li smještaj upotrebljiv |
| dateAdded | DATE | Verijeme dodavanja smještaja u sustav |

AccommodationType Ovaj entitet sadrži podatke o tipu smještaja (stan u zgradi, stan u kući, iznajmljeno, u vlasništvu klinike). Atributi su: TypeID (primary key), description. Ovaj je entitet u One-to-Many vezi s entitetom Accomodation preko

atributaTypeID.

| AccommodationType | | |
|-------------------|----------|---|
| TypeID | SMALLINT | Jedinstveni identifikator vrste smještaja, automatski generiran |
| description | TEXT | Opis vrste smještaja (stan u kući, stan u zgradi, soba u hotelu ili motelu) |

Equipped Ovaj entitet sadrži podatke o vrsti opremljenosti smještaja (potpuno opremljen, djelomično opremljen). Atributi su: EquippedID (primary key), equipment. Ovaj je entitet u One-to-Many vezi sa entitetom Accommodation preko atributa EquippedID.

| Equipped | | |
|------------|----------|---|
| EquippedID | SMALLINT | Jedinstveni identifikator opremljenosti smještaja, automatski generiran |
| equipment | TEXT | Opis opremljenosti smještaja (potpuno opremljen, djelomično opremljen) |

AccommodationOccupied Ovaj entitet sadrži podatke o zauzetosti smještaja kojima raspolaže pojedina klinika. Atributi su: PatientID (foreign key), AccommodationID (foreign key), dateTo, dateFrom. Ovaj je entitet u One-to-One vezi sa entitetom Accommodation preko atributa AccommodationID te u One-to-One vezi sa entiteom Patient preko atributa PatientID.

| AccommodationOccupied | | |
|-----------------------|-----------|---|
| ID | BIGSERIAL | Jedinstveni identifikator rekorda zauzeća smještaja |
| PatientID | BIGSERIAL | Jedinstveni identifikator pacijenta |
| AccommodationID | BIGSERIAL | Jedinstveni identifikator smještaja |
| dateFrom | DATE | Datum od kojeg je smještaj dostupan |
| dateTo | DATE | Datum do kojeg je smještaj dostupan |

Transporter Ovaj entitet sadrži podatke o prijevoznicima s kojima klinika ima ugovore za prijevoz pacijenata. Atributi su: TransporterID (primary key), organisationName, phone, address, TownID (foreign key), active. Ovaj je entitet u Many-to-Many vezi sa entitetom Town preko atributa TownID, u Many-to-Many vezi sa entitetom Clinic preko atributa ClinicID te u One-to-Many vezi sa entitetom Vehicle preko atributa TransporterID.

| Transporter | | |
|------------------|-----------|--|
| TransporterID | BIGSERIAL | Jedinstveni identifikator prijevoznika, automatski generiran |
| organisationName | VARCHAR | Ime prijevoznika |
| phone | VARCHAR | Broj mobitela prijevoznika |
| address | VARCHAR | Adresa baze prijevoznika |
| TownID | INT | ID grada u kojem se smještaj nalazi |
| active | BIT | Je li prijevoznik radi |

Vehicle Ovaj entitet sadrži podatke o vozilima kojima transporter raspolaže. Atributi su: VehicleID (primary key), capacity TypeID (foreign key), TransporterID (foreign key) te active. Ovaj je entitet u Many-to-One vezi sa entitetom Transporter preko atributa TransporterID, u Many-to-Many vezi sa entitetom VehicleOccupied preko atributa VehicleID, u One-to-One vezi sa entitetom VehicleType preko atributa VehicleID te u One-to-Many vezi sa entitetom VehicleAvailability preko atributa VehicleID.

| Vehicle | | |
|---------------|-----------|---|
| VehicleID | BIGSERIAL | Jedinstveni identifikator vozila, automatski generiran |
| capacity | SMALLINT | Kapacitet vozila (2 osobe, 4 osobe, 5 osoba. . .) |
| TypeID | SMALLINT | Vresta vozila |
| TransporterID | INT | Jedinstveni identifikator prijevoznika kojem vozilo pripada |

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

| Vehicle | | |
|---------|-----|-------------------------|
| active | BIT | Je li vozilo u funkciji |

VehicleType Ovaj entitet sadrži podatke o vrsti vozila. Atributi su:TypeID(primary key), description. Ovaj je entitet u One-to-One vezi sa entitetom Vehicle preko atributa VehicleID.

| VehicleType | | |
|-------------|----------|--|
| TypeID | SMALLINT | Jedinstveni identifikator vrste vozila, automatski generiran |
| description | TEXT | Opis vrste vozila (auto, kombi, min-bus...) |

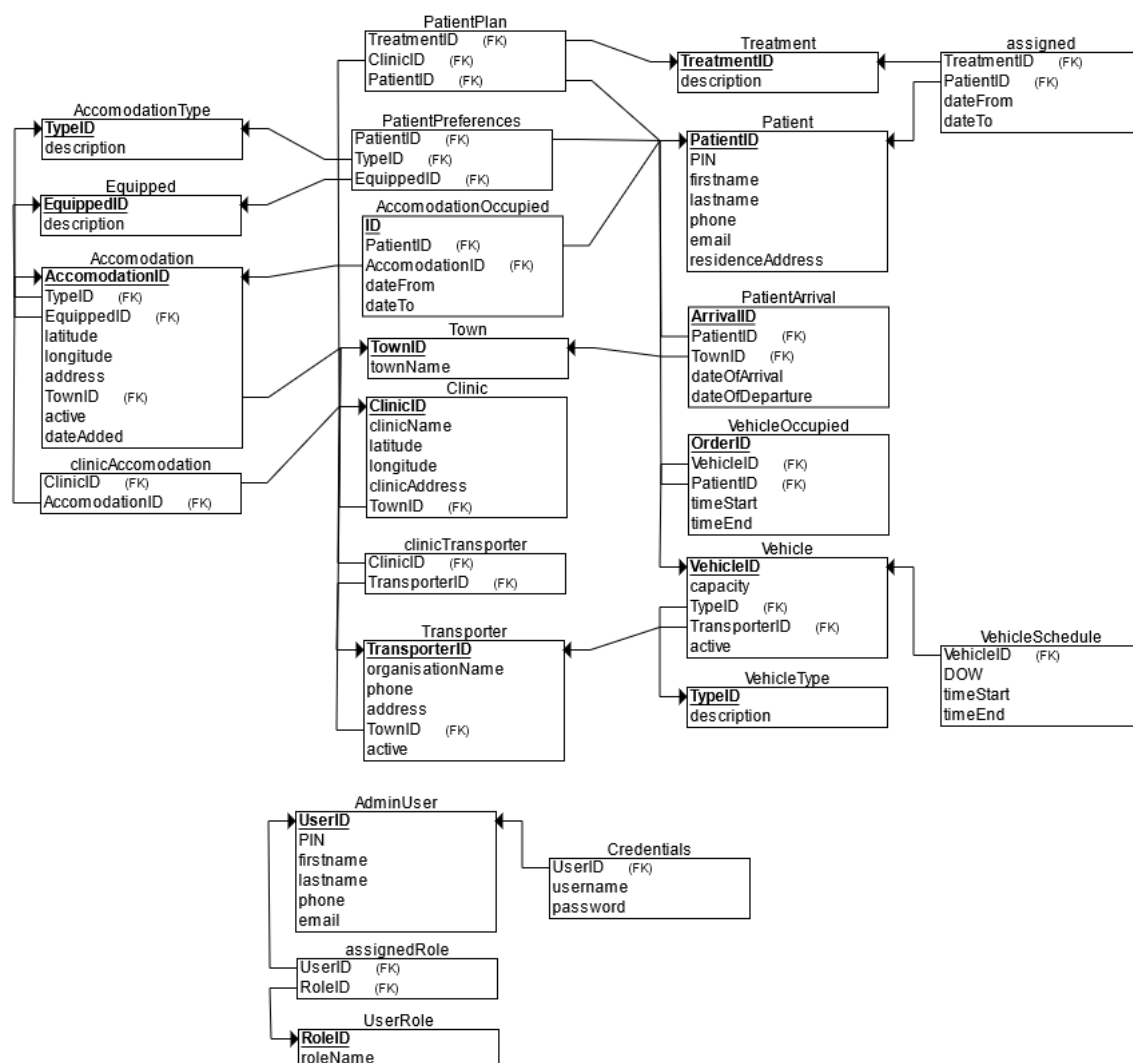
VehicleOccupied Ovaj entitet sadrži podatke o vremenima kada je koje vozilo zauzeto, tj. kada prevozi pacijente od smještaja do klinike te natrag. Atributi su: VehicleID (foreign key), PatientID (foreign key), timeStart, timeEnd. Ovaj je entitet u One-to-Many vezi sa entitetom Vehicle preko atributa VehicleID te u One-to-One vezi sa entitetom Patient preko atributa PatientID.

| VehicleOccupied | | |
|-----------------|-----------|---|
| OrderID | BIGSERIAL | Jedinstveni identifikator rekorda zauzeća vozila |
| VehicleID | BIGSERIAL | Jedinstveni identifikator vozila koje je zauzeto |
| PatientID | BIGSERIAL | Jedinstveni identifikator pacijenta kojem je vozilo dodijeljeno |
| timeStart | TIME | Vrijem od kada je vozilo zauzeto |
| timeEnd | TIME | Vrijeme do kada je vozilo zauzeto |

VehicleSchedule Ovaj entitet sadrži podatke o vremenima kada je koje vozilo dostupno, tj. radno vrijeme radnih dana. Atributi su: VehicleID (foreign key), DOW (day of the week), timeStart, timeEnd. Ovaj je entitet u Many-to-Many vezi sa entitetom Vehicle preko atributa VehicleID.

| VehicleSchedule | | |
|-----------------|----------|---|
| VehicleID | INT | Jedinstveni identifikator vozila |
| DOW | SMALLINT | Dan u tjednu u kojem je vozilo slobodno |
| timeStart | TIME | Vrijem od kada je vozilo slobodno |
| timeEnd | TIME | Vrijeme do kada je vozilo slobodno |

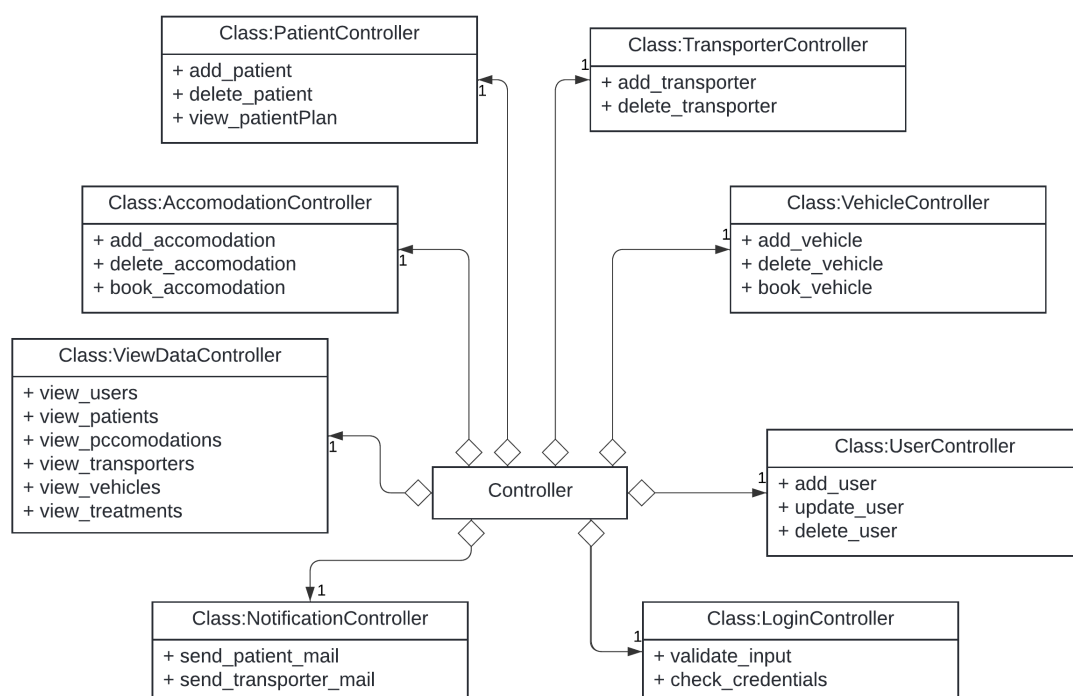
4.1.2 Dijagram baze podataka



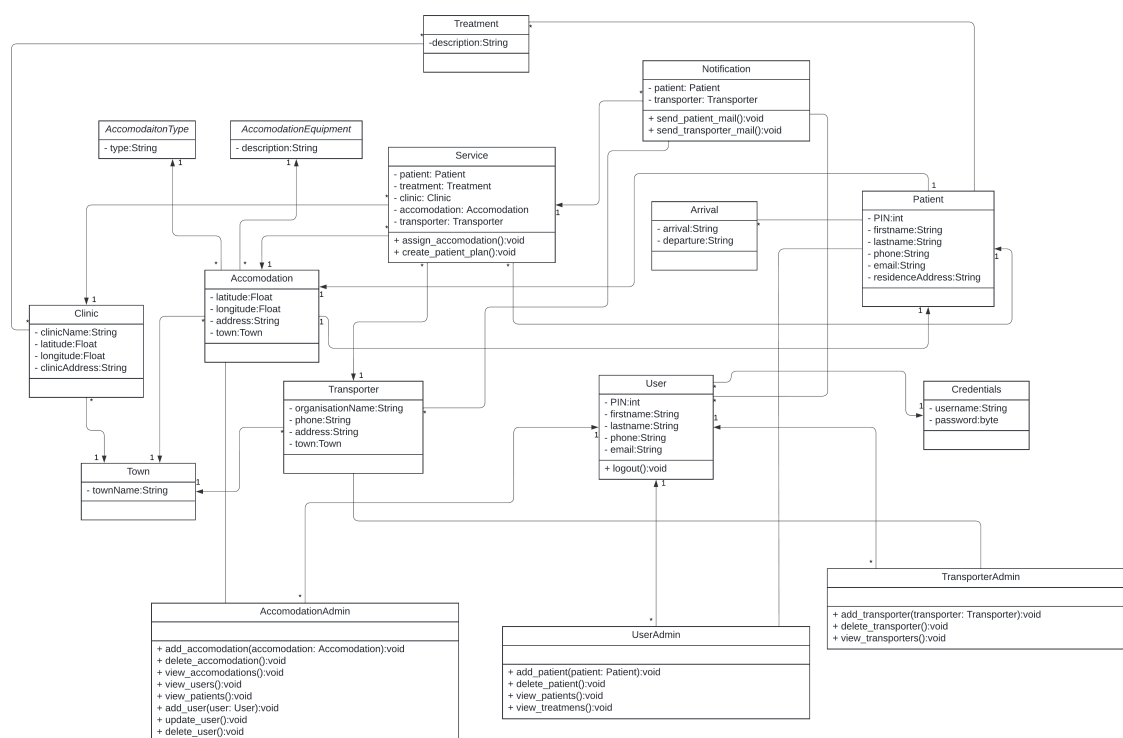
Slika 4.1: Sheme baze podataka

4.2 Dijagram razreda

Na slikama 4.2 i 4.3 su prikazani razredi *Model* i *Controller* iz MVC arhitekture. Razredi prikazani slikom 4.2 nasljeđuju razred *Controller*. Metode koje smo definirali unutar tih razreda pripremaju podatke i šalju ih bazi koja ih onda obrađuje. Baza manipulira modelima te na kraju vraća podatke kako bi ih *View* mogao prikazati. Model razredi, prikazani slikom 4.3, prikazuju strukturu baze podataka te integrirane funkcije koje služe za obradu, slanje ili primanje podataka.



Slika 4.2: Dijagram razreda Controller



Slika 4.3: Dijagram razreda Model

dio 2. revizije

Prilikom druge predaje projekta dijagram razreda i opisi moraju odgovarati stvarnom stanju implementacije

4.3 Dijagram stanja

dio 2. revizije

*Potrebno je priložiti dijagram stanja i opisati ga. Dovoljan je jedan dijagram stanja koji prikazuje **značajan dio funkcionalnosti** sustava. Na primjer, stanja korisničkog sučelja i tijekom korištenja neke ključne funkcionalnosti jesu značajan dio sustava, a registracija i prijava nisu.*

4.4 Dijagram aktivnosti

dio 2. revizije

Potrebno je priložiti dijagram aktivnosti s pripadajućim opisom. Dijagram aktivnosti treba prikazivati značajan dio sustava.

4.5 Dijagram komponenti

dio 2. revizije

Potrebno je priložiti dijagram komponenti s pripadajućim opisom. Dijagram komponenti treba prikazivati strukturu cijele aplikacije.

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

U timu smo za vrijeme projekta komunicirali ponajviše preko aplikacija WhatsApp¹ te Discorda², a sa asistentom i demonstratorom preko Microsoft Teams³ i Microsoft Outlook⁴. Tekst u dokumentaciji smo uređivali pomoću latex-a⁵ u editoru texStudio⁶ te generirali PDF dokument iz latex dokumenta pomoću TexLive⁷. Dijagrame uporabe te sekvencijske dijagrame smo izradili pomoću alata Astah UML⁸.

Nedostaje za ostale dijagrame!!!!

Model baze podataka je napravljen pomoću alata ERDplus⁹. Kako bi smo zajedno mogli raditi na projektu u isto vrijeme te ujedno i pratiti razvojne verzije našeg projekta smo koristili Git¹⁰ zajedno za udaljenim repozitorijom GitHub¹¹. Razvojna okruženja koja su korištena su Visual Studio Code¹² te pgAdmin¹³. Frontend je napisan u programskom jeziku javascript¹⁴ pomoću biblioteke React¹⁵, dok je za backend korišten PostgreSQL¹⁶. Unit testovi su odrađeni pomoću alata pgTAP¹⁷, dok su integracijski testovi napravljeni pomoću alata Selenium¹⁸ i programskog jezika Java¹⁹????

¹<https://www.whatsapp.com/>

²<https://discord.com/>

³<https://teams.microsoft.com/v2/>

⁴<https://outlook.office.com/mail/>

⁵<https://www.latex-project.org/>

⁶<https://www.texstudio.org/>

⁷<https://www.tug.org/texlive/>

⁸<https://astah.net/products/astah-uml/>

⁹<https://erdplus.com/>

¹⁰<https://git-scm.com/>

¹¹<https://github.com/>

¹²<https://code.visualstudio.com/>

¹³<https://www.pgadmin.org/>

¹⁴<https://www.javascript.com/>

¹⁵<https://react.dev/>

¹⁶<https://www.postgresql.org/>

¹⁷<https://pgtap.org/>

¹⁸<https://www.postgresql.org/>

¹⁹<https://www.java.com/en/>

Aplikacija je puštena u pogon na servisu render²⁰.

Fali za latex i texStudio

²⁰<https://render.com/>

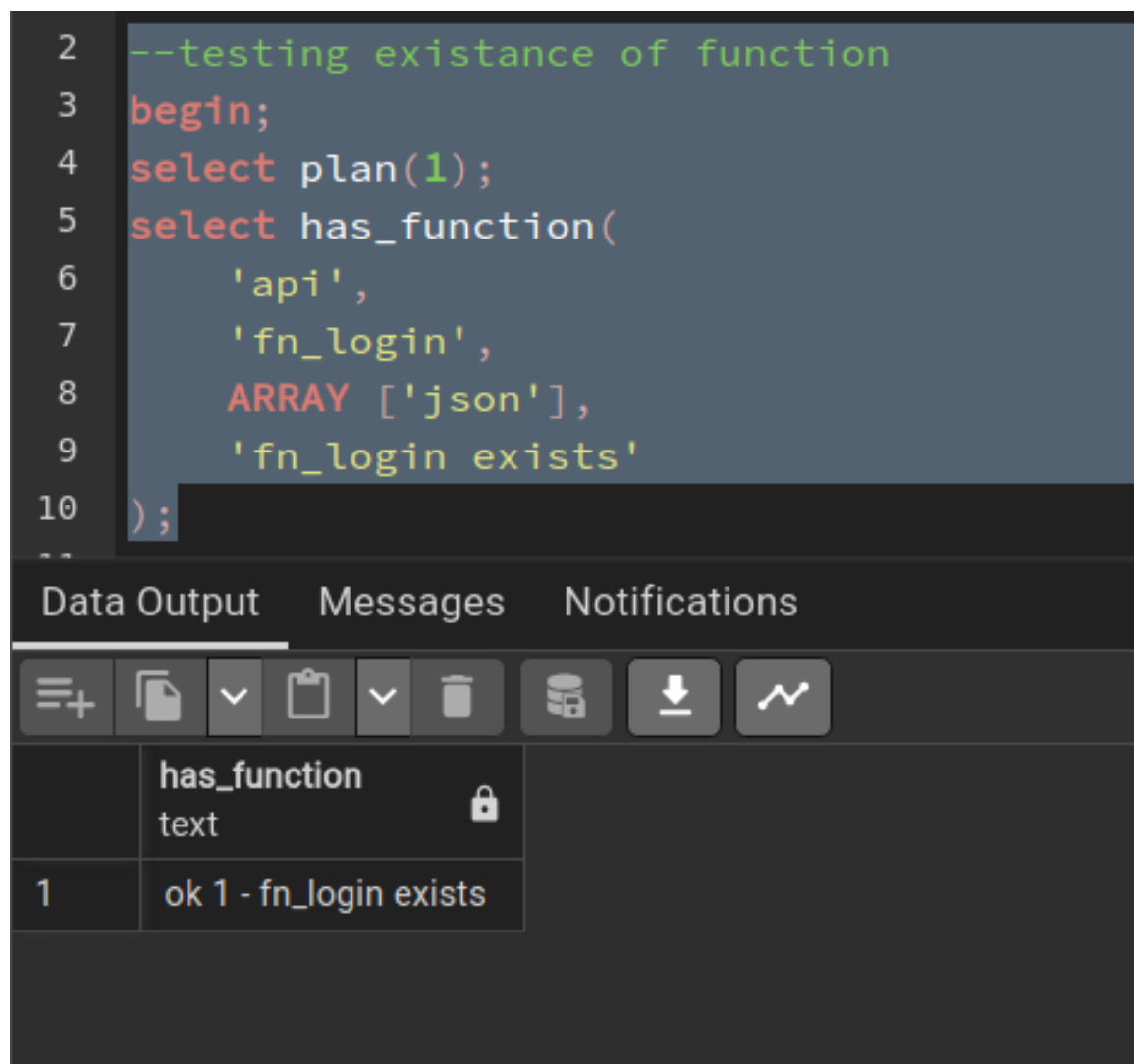
5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje jedinica je provedeno pomoću alata pgTAP. pgTAP je skup funkcija baze podataka koje olakšavaju pisanje testova. Pomoću ovog smo alata testirali izvođenje i ponašanje backenda-a ostvarenog kroz funkcije u bazi napisane plpgsql programskim jezikom. Alat nudi brojne definirane metode koje omogućuju definiranje očekivanog ispisa za određeni ulaz, uspoređivanje rezultata odrađene funkcije sa podacima iz tablica, itd.

5.2.2 Ispitni slučaj 1 - funkcionalnost prijave

Ovaj ispitni slučaj ispituje funkcionalnost prijave u sustav. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* ze provjeru dvaju argumenata te na osnovu njihovog podudaranje ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok <broj testa> - <opis testa>* ili *not ok <broj testa> - <opis testa>*.



```
2  --testing existance of function
3  begin;
4  select plan(1);
5  select has_function(
6      'api',
7      'fn_login',
8      ARRAY ['json'],
9      'fn_login exists'
10 );
```

| | has_function | |
|---|------------------------|--|
| | text | |
| 1 | ok 1 - fn_login exists | |

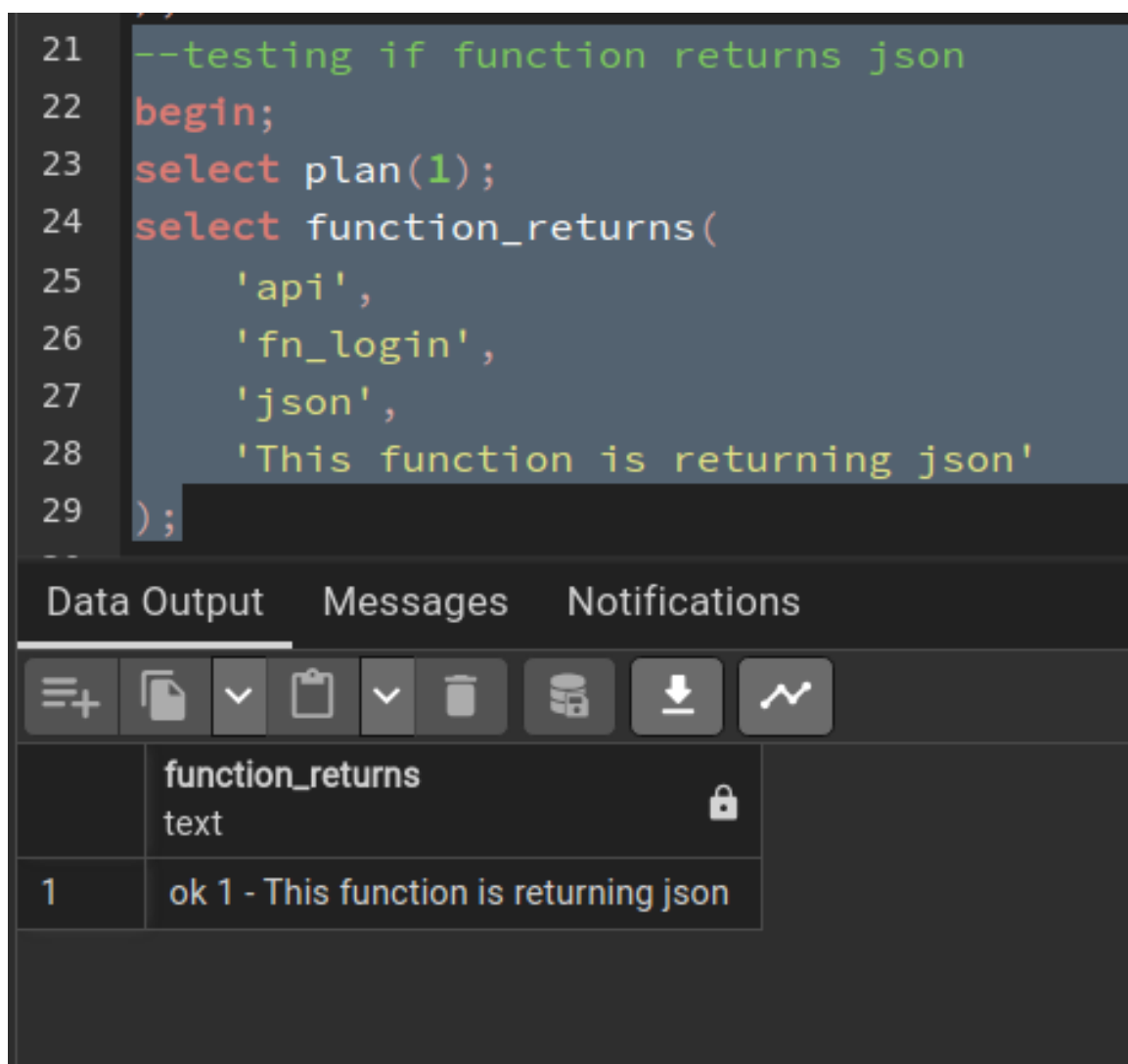
Slika 5.1: Pokretanje testa za provjeru postojanja funkcije

```
11  --testing is function written in plpgsql
12  begin;
13  select plan(1);
14  select function_lang_is(
15      'api',
16      'fn_login',
17      ARRAY ['json'],
18      'plpgsql',
19      'fn_login is written using plpgsql'
20  );
```

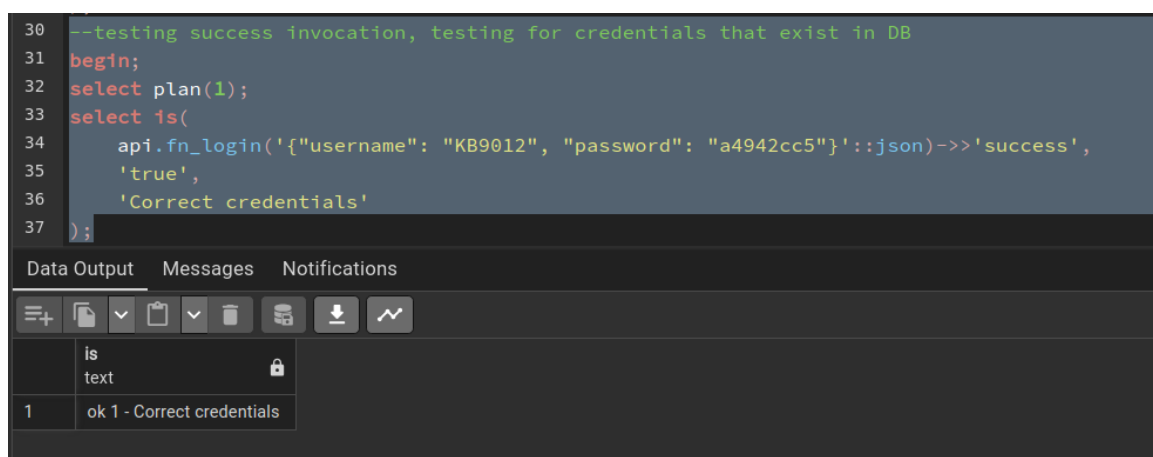
Data Output Messages Notifications

| | | |
|---|--|---|
| | function_lang_is text | 🔒 |
| 1 | ok 1 - fn_login is written using plpgsql | |

Slika 5.2: Pokretanje testa za provjeru jezika kojim je napisana funkcija



Slika 5.3: Pokretanje testa za provjeru povratnog tipa podatka funkcije



Slika 5.4: Pokretanje testa za provjeru rada same funkcije, uspijeh


```
38 --testing failure invocation, testing for credentials that don't exist
39 begin;
40 select plan(1);
41 select is(
42     api.fn_login('{ "username": "KB9012", "password": "a4942"}'::json)->>'success',
43     'false',
44     'Wrong credentials'
45 );
```

Data Output Messages Notifications

| | is | |
|---|--------------------------|---|
| | text | 🔒 |
| 1 | ok 1 - Wrong credentials | |

Slika 5.5: Pokretanje testa za provjeru rada same funkcije, neuspjeh

```
46 --testing if no username and no password are providede, should fail
47 begin;
48 select plan(1);
49 select is(
50     api.fn_login('{ "username": "", "password": ""}'::json)->>'success',
51     'false',
52     'Wrong credentials'
53 );
```

Data Output Messages Notifications

| | is | |
|---|--------------------------|---|
| | text | 🔒 |
| 1 | ok 1 - Wrong credentials | |

Slika 5.6: Pokretanje testa za provjeru rada same funkcije, neuspjeh

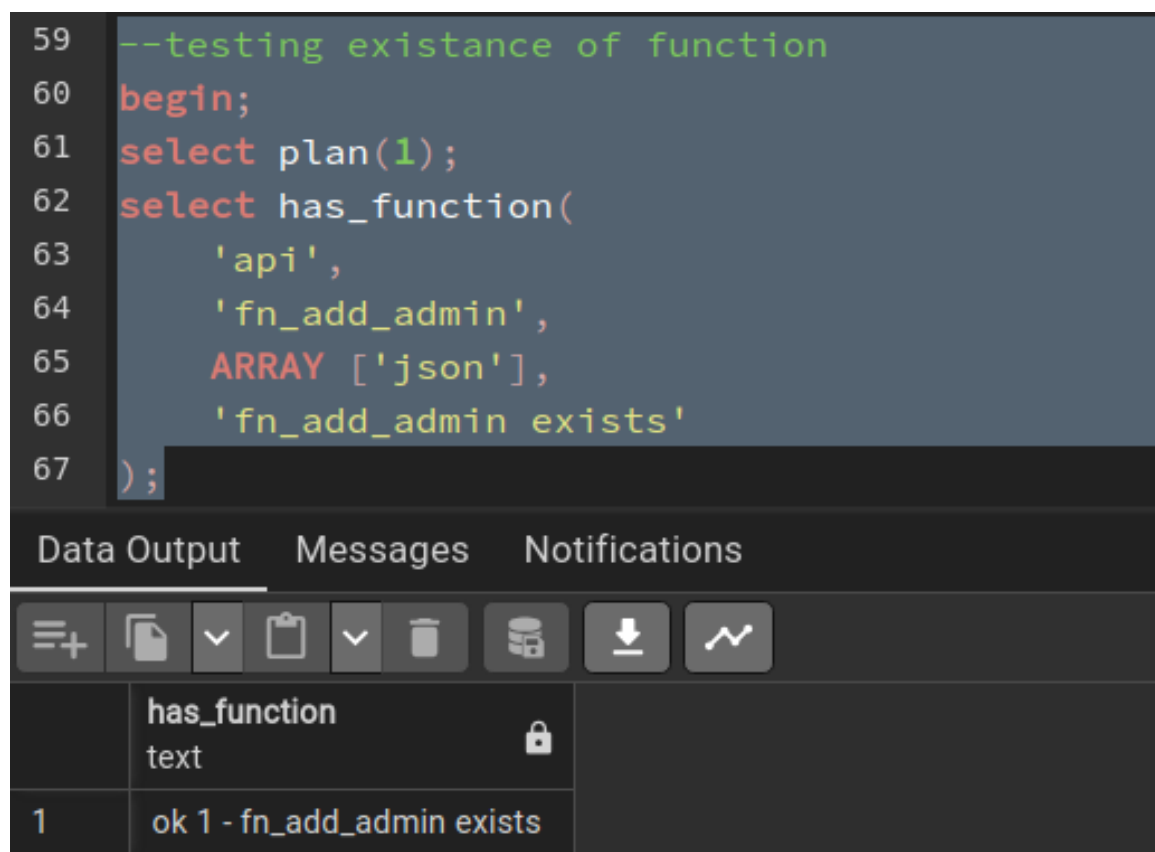
```
1  --##### UNIT TEST 1; TESTING LOGIN FUNCTION #####
2  --testing existence of function
3  begin;
4  select plan(1);
5  select has_function(
6    'api',
7    'fn_login',
8    ARRAY ['json'],
9    'fn_login exists'
10 );
11 --testing is function written in plpgsql
12 begin;
13 select plan(1);
14 select function_lang_is(
15   'api',
16   'fn_login',
17   ARRAY ['json'],
18   'plpgsql',
19   'fn_login is written using plpgsql'
20 );
21 --testing if function returns json
22 begin;
23 select plan(1);
24 select function_returns(
25   'api',
26   'fn_login',
27   'json',
28   'This function is returning json'
29 );
30 --testing success invocation, testing for credentials that exist in DB
31 begin;
32 select plan(1);
33 select is(
34   api.fn_login('{"username": "KB9012", "password": "a4942cc5"}'::json)-->'success',
35   'true',
36   'Correct credentials'
37 );
38 --testing failure invocation, testing for credentials that don't exist
39 begin;
40 select plan(1);
41 select is(
42   api.fn_login('{"username": "KB9012", "password": "a4942"}'::json)-->'success',
43   'false',
44   'Wrong credentials'
45 );
46 --testing if no username and no password are provided, should fail
47 begin;
48 select plan(1);
49 select is(
50   api.fn_login('{"username": "", "password": ""}'::json)-->'success',
51   'false',
52   'Wrong credentials'
53 -- );
54 select * from finish(true);
55 rollback;
56 --##### UNIT TEST 1; TESTING LOGIN FUNCTION #####
```

Slika 5.7: Kod ispitnog slučaja 1

5.2.3 Ispitni slučaj 2 - funkcionalnost dodavanja novog administratora

Ovaj ispitni slučaj ispituje funkcionalnost dodavanja novog administratora. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* za provjeru dvaju argumenata te na osnovu njihovog podudaranja ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok <broj testa> - <opis testa>* ili *not ok <broj testa> - <opis testa>*.

```
59  --testing existance of function
60  begin;
61  select plan(1);
62  select has_function(
63      'api',
64      'fn_add_admin',
65      ARRAY ['json'],
66      'fn_add_admin exists'
67  );
```



| | has_function | |
|---|----------------------------|--|
| | text | |
| 1 | ok 1 - fn_add_admin exists | |

Slika 5.8: Pokretanje testa za provjeru postojanja funkcije

```
68 --testing is function written in plpgsql
69 begin;
70 select plan(1);
71 select function_lang_is(
72     'api',
73     'fn_add_admin',
74     ARRAY ['json'],
75     'plpgsql',
76     'fn_add_admin is written using plpgsql'
77 );
```

Data Output Messages Notifications

| | | |
|---|--|---|
| | function_lang_is text | 🔒 |
| 1 | ok 1 - fn_add_admin is written using plpgsql | |

Slika 5.9: Pokretanje testa za provjeru jezika kojim je napisana funkcija

```
78 --testing if function returns json
79 begin;
80 select plan(1);
81 select function_returns(
82     'api',
83     'fn_add_admin',
84     'json',
85     'This function is returning json'
86 );
```

Data Output Messages Notifications

function_returns
text

| | |
|---|--|
| 1 | ok 1 - This function is returning json |
|---|--|

Slika 5.10: Pokretanje testa za provjeru povratnog tipa podatka funkcije

```
87 --testing success invocation, when adding new admin
88 begin;
89 select plan(1);
90 select is (api.fn_add_admin(
91     '{
92         "PIN": "50125736",
93         "firstname": "Pero",
94         "lastname": "Perić",
95         "phone": "+3859173035",
96         "email": "pero.peric@gmail.com",
97         "roleList": [2]
98     }::json)->>'success',
99     'true',
100     'New admin has been created'
101 );
```

Data Output Messages Notifications

| | | |
|---|-----------------------------------|---|
| | is text | 🔒 |
| 1 | ok 1 - New admin has been created | |

Slika 5.11: Pokretanje testa za provjeru rada same funkcije, uspijeh

```
88 begin;
89 select plan(2);
90 select is (api.fn_add_admin(
91     '{
92         "PIN": "50125736",
93         "firstname": "Pero",
94         "lastname": "Perić",
95         "phone": "+3859173035",
96         "email": "pero.peric@gmail.com",
97         "roleList": [2]
98     }'::json)->>'success',
99     'true',
100     'New admin has been created'
101 );
102 --testing failure invocation, when adding existing admin
103 select is (api.fn_add_admin(
104     '{
105         "PIN": "50125736",
106         "firstname": "Pero",
107         "lastname": "Perić",
108         "phone": "+3859173035",
109         "email": "pero.peric@gmail.com",
110         "roleList": [2]
111     }'::json)->>'success',
112     'false',
113     'Admin already exists'
114 );
```

Data OutputMessagesNotifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

| | | |
|---|-----------------------------|---|
| | is text | 🔒 |
| 1 | ok 2 - Admin already exists | |

Slika 5.12: Pokretanje testa za provjeru rada same funkcije, neuspjeh

Potplaćeni

stranica 62/98

17. siječnja 2024.

```
88 begin;
89 select plan(2);
90 select is (api.fn_add_admin(
91     '{
92         "PIN": "50125736",
93         "firstname": "Pero",
94         "lastname": "Perić",
95         "phone": "+3859173035",
96         "email": "pero.peric@gmail.com",
97         "roleList": [2]
98     }'::json)->>'success',
99     'true',
100     'New admin has been created'
101 );
102
103 prepare new_credentials_entry as
104     select
105         *
106     from
107         auth.credentials
108     order by
109         userid desc
110     limit 1;
111
112 prepare expected_credentials_entry as select 'PP5736'::varchar, left(MD5('50125736'), 8)::varchar, 30::bigint;
113
114 select results_eq(
115     'new_credentials_entry',
116     'expected_credentials_entry',
117     'Added new row to table credentials'
118 );
119 --testing failure invocation, when adding existing admin
```

Data Output Messages Notifications

| | results_eq | |
|---|---|---|
| | text | 🔒 |
| 1 | ok 2 - Added new row to table credentials | |

Slika 5.13: Pokretanje testa za provjeru automatskog kreiranja vjerodajnica za novog administratora


```

88 begin;
89 select plan(2);
90 select is (api.fn_add_admin(
91     '{
92         "PIN": "50125736",
93         "firstname": "Pero",
94         "lastname": "Perić",
95         "phone": "+3859173035",
96         "email": "pero.peric@gmail.com",
97         "roleList": [2]
98     }'::json)->>'success',
99     'true',
100     'New admin has been created'
101 );
102 --testing failure invocation, when adding existing admin
103 -- select is (api.fn_add_admin(
104 --     '{
105 --         "PIN": "50125736",
106 --         "firstname": "Pero",
107 --         "lastname": "Perić",
108 --         "phone": "+3859173035",
109 --         "email": "pero.peric@gmail.com",
110 --         "roleList": [2]
111 --     }'::json)->>'success',
112 --     'false',
113 --     'Admin already exists'
114 -- );
115 --testing if roles are added to newly created admin
116 prepare new_assignedrole_entry as
117     select
118         *
119     from
120         assignedrole
121     order by
122         userid desc
123     limit 1;
124 prepare expected_assignedrole_entry as select 29::bigint, 2::bigint;
125
126 select results_eq(
127     'new_assignedrole_entry',
128     'expected_assignedrole_entry',
129     'Added new row to table assignedrole'
130 );

```

Data Output Messages Notifications

+

📄

▼

📋

▼

🗑️

🔄

⬇️

📈

| | | |
|---|--|---|
| | results_eq text | 🔒 |
| 1 | ok 2 - Added new row to table assignedrole | |

Slika 5.14: Pokretanje testa za provjeru automatskog dodjeljivanja uloge/uloga za novog administratora

Potplaćeni

stranica 64/98

17. siječnja 2024.

```
58  --##### UNIT TEST 2; TESTING ADDING ADMIN #####
59  --testing existance of function
60  begin;
61  select plan(1);
62  select has_function(
63      'api',
64      'fn_add_admin',
65      ARRAY ['json'],
66      'fn_add_admin exists'
67  );
68  --testing is function written in plpgsql
69  begin;
70  select plan(1);
71  select function_lang_is(
72      'api',
73      'fn_add_admin',
74      ARRAY ['json'],
75      'plpgsql',
76      'fn_add_admin is written using plpgsql'
77  );
78  --testing if function returns json
79  begin;
80  select plan(1);
81  select function_returns(
82      'api',
83      'fn_add_admin',
84      'json',
85      'This function is returning json'
86  );
87  --testing success invocation, when adding new admin
88  begin;
89  select plan(2);
90  select is (api.fn_add_admin(
91      '{
92          "PIN": "50125736",
93          "firstname": "Pero",
94          "lastname": "Perić",
95          "phone": "+3859173035",
96          "email": "pero.peric@gmail.com",
97          "roleList": [2]
98      }::json)->'success',
99      'true',
100      'New admin has been created'
101  );
```

```
102 --testing failure invocation, when adding existing admin
103 select is (api.fn_add_admin(
104     '{
105         "PIN": "50125736",
106         "firstname": "Pero",
107         "lastname": "Perić",
108         "phone": "+3859173035",
109         "email": "pero.peric@gmail.com",
110         "roleList": [2]
111     }::json)-->'success',
112     'false',
113     'Admin already exists'
114 );
115 --testing if roles are added to newly created admin
116 prepare new_assignedrole_entry as
117     select
118         *
119     from
120         assignedrole
121     order by
122         userid desc
123     limit 1;
124 prepare expected_assignedrole_entry as select 29::bigint, 2::bigint;
125
126 select results_eq(
127     'new_assignedrole_entry',
128     'expected_assignedrole_entry',
129     'Added new row to table assignedrole'
130 );
131
132 --testin if credentials are created for newly added admin
133 prepare new_credentials_entry as
134     select
135         *
136     from
137         auth.credentials
138     order by
139         userid desc
140     limit 1;
141
142 prepare expected_credentials_entry as select 'PP5736'::varchar, left(MD5('50125736'), 8)::varchar, 30::bigint;
143
144 select results_eq(
145     'new_credentials_entry',
146     'expected_credentials_entry',
147     'Added new row to table credentials'
148 );
149
150 select * from finish(true);
151 deallocate new_assignedrole_entry;
152 deallocate expected_assignedrole_entry;
153 deallocate new_credentials_entry;
154 deallocate expected_credentials_entry;
155 rollback;
156 --##### UNIT TEST 2; TESTING ADDING ADMIN #####
```

Slika 5.15: Kod ispitnog slučaja 2

5.2.4 Ispitni slučaj 3 - funkcionalnost brisanja administratora

Ovaj ispitni slučaj ispituje funkcionalnost brisanja administratora. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* za provjeru dvaju argumenata te na osnovu njihovog podudaranja ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok* <broj testa> - <opis testa> ili *not ok* <broj testa> - <opis testa>.

```
159 --testing existance of function
160 begin;
161 select plan(1);
162 select has_function(
163     'api',
164     'fn_delete_admin',
165     ARRAY ['json'],
166     'fn_delete_admin exists'
167 );
```

Data Output Messages Notifications

| | | |
|---|-------------------------------|--|
| | has_function | |
| | text | |
| 1 | ok 1 - fn_delete_admin exists | |

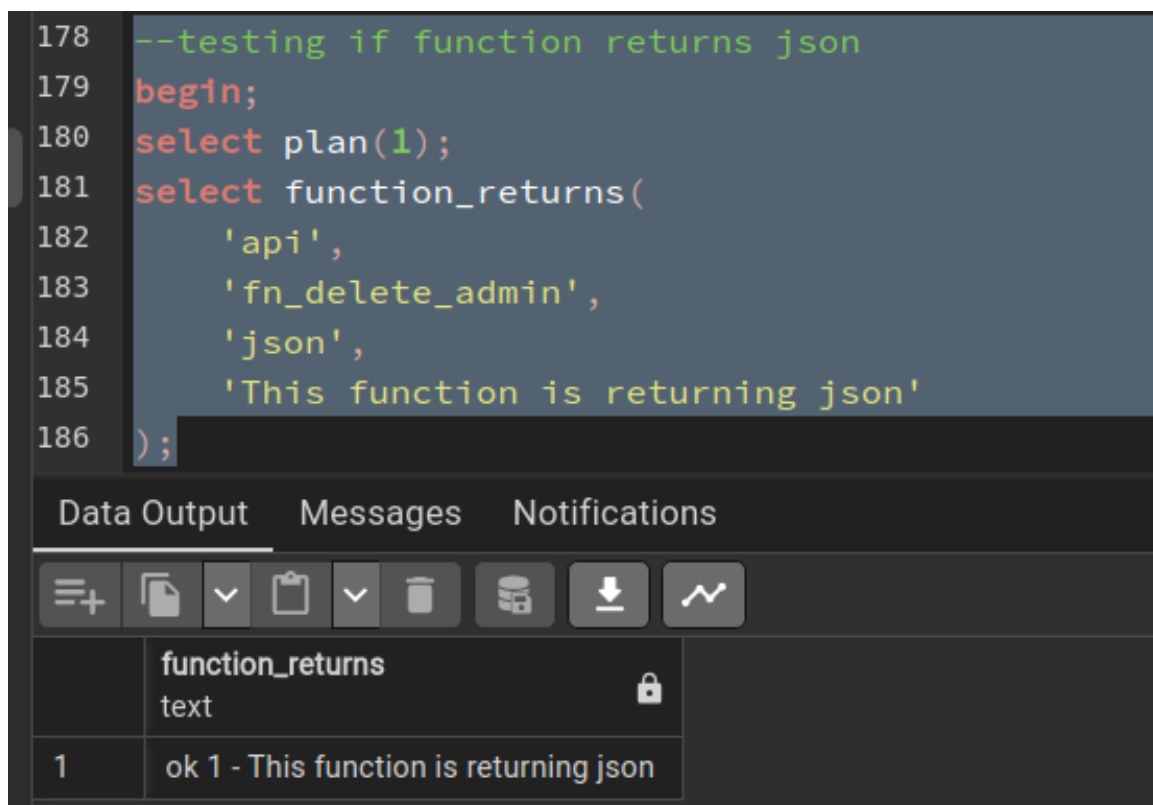
Slika 5.16: Pokretanje testa za provjeru postojanja funkcije

```
168 --testing is function written in plpgsql
169 begin;
170 select plan(1);
171 select function_lang_is(
172     'api',
173     'fn_delete_admin',
174     ARRAY ['json'],
175     'plpgsql',
176     'fn_delete_admin is written using plpgsql'
177 );
```

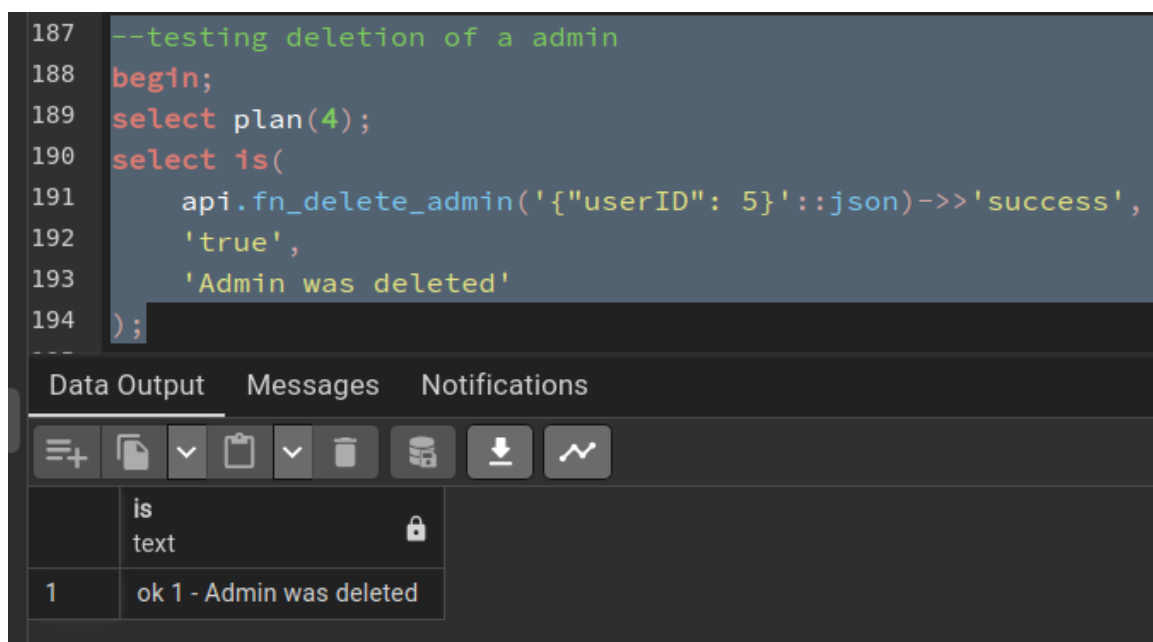
Data Output Messages Notifications

| | | |
|---|---|---|
| | function_lang_is text | 🔒 |
| 1 | ok 1 - fn_delete_admin is written using plpgsql | |

Slika 5.17: Pokretanje testa za provjeru jezika kojim je napisana funkcija



Slika 5.18: Pokretanje testa za provjeru povratnog tipa podatka funkcije



Slika 5.19: Pokretanje testa za provjeru rada same funkcije, uspijeh

```
188 begin;
189 select plan(4);
190 select is(
191     api.fn_delete_admin('{"userID": 5}'::json)->>'success',
192     'true',
193     'Admin was deleted'
194 );
195 --testing if deleting nonexisting admin returns false
196 select is(
197     api.fn_delete_admin('{"userID": 5}'::json)->>'success',
198     'false',
199     'Admin does not exist, can not delete'
200 );
```

| | Data Output | Messages | Notifications |
|---|---|----------|---------------|
| | is text | | |
| 1 | ok 2 - Admin does not exist, can not delete | | |

Slika 5.20: Pokretanje testa za provjeru rada same funkcije, neuspjeh

```
188 begin;
189 select plan(4);
190 select is(
191     api.fn_delete_admin('{"userID": 5}'::json)->>'success',
192     'true',
193     'Admin was deleted'
194 );
195 --testing to see if data connected to deleted admin user is also deleted, such as credentials and assignedrole(s)
196 prepare deleted_admin_credentials as select count(*) from auth.credentials where userid = 5;
197 prepare zero_rows as select 0::bigint;
198
199 select results_eq(
200     'deleted_admin_credentials',
201     'zero_rows',
202     'Admin credentials have been deleted'
203 );
```

| | Data Output | Messages | Notifications |
|---|--|----------|---------------|
| | results_eq text | | |
| 1 | ok 2 - Admin credentials have been deleted | | |

Slika 5.21: Pokretanje testa za provjeru automatskog brisanja vjerodajnica za izbrisanog administratora

```
188 begin;
189 select plan(4);
190 select is(
191     api.fn_delete_admin('{"userID": 5}::json')->>'success',
192     'true',
193     'Admin was deleted'
194 );
195 prepare deleted_admin_roles as select count(*) from assignedrole where userid = 5;
196 prepare zero_role_rows as select 0::bigint;
197
198 select results_eq(
199     'deleted_admin_roles',
200     'zero_role_rows',
201     'Admin role(s) have been deleted'
202 );
203
```

Data Output Messages Notifications

results_eq
text

| | |
|---|--|
| 1 | ok 2 - Admin role(s) have been deleted |
|---|--|

Slika 5.22: Pokretanje testa za provjeru automatskog brisanja pridjeljenih uloga za obrisano administratora


```
158 ----- UNIT TEST 3; TESTING DELETING ADMIN -----
159 --testing existence of function
160 begin;
161 select plan(1);
162 select has_function(
163     'api',
164     'fn_delete_admin',
165     ARRAY ['json'],
166     'fn_delete_admin exists'
167 );
168 --testing is function written in plpgsql
169 begin;
170 select plan(1);
171 select function_lang_is(
172     'api',
173     'fn_delete_admin',
174     ARRAY ['json'],
175     'plpgsql',
176     'fn_delete_admin is written using plpgsql'
177 );
178 --testing if function returns json
179 begin;
180 select plan(1);
181 select function_returns(
182     'api',
183     'fn_delete_admin',
184     'json',
185     'This function is returning json'
186 );
187 --testing deletion of a admin
188 begin;
189 select plan(4);
190 select is(
191     api.fn_delete_admin('{"userID": 5}::json-->>'success',
192     'true',
193     'Admin was deleted'
194 );
195 --testing if deleting nonexisting admin returns false
196 select is(
197     api.fn_delete_admin('{"userID": 5}::json-->>'success',
198     'false',
199     'Admin does not exist, can not delete'
200 );
201 --testing to see if data connected to deleted admin user is also deleted, such as credentials and assignedrole(s)
202 prepare deleted_admin_credentials as select count(*) from auth.credentials where userid = 5;
203 prepare zero_rows as select 0::bigint;
204
205 select results_eq(
206     'deleted_admin_credentials',
207     'zero_rows',
208     'Admin credentials have been deleted'
209 );
210 prepare deleted_admin_roles as select count(*) from assignedrole where userid = 5;
211 prepare zero_role_rows as select 0::bigint;
212
213 select results_eq(
214     'deleted_admin_roles',
215     'zero_role_rows',
216     'Admin role(s) have been deleted'
217 );
218 select * from finish(true);
219 deallocate deleted_admin_credentials;
220 deallocate zero_rows;
221 deallocate deleted_admin_roles;
222 deallocate zero_role_rows;
223 rollback;
224 ----- UNIT TEST 3; TESTING DELETING ADMIN -----
```

Slika 5.23: Kod isptinog slučaja 3

5.2.5 Ispitni slučaj 4 - funkcionalnost dodavanja smještaja

Ovaj ispitni slučaj ispituje funkcionalnost dodavanja smještaja. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* za provjeru dvaju argumenata te na osnovu njihovog podudaranja ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok <broj testa> - <opis testa>* ili *not ok <broj testa> - <opis testa>*.

```
227 --testing existance of function
228 begin;
229 select plan(1);
230 select has_function(
231     'api',
232     'fn_add_accommodation',
233     ARRAY ['json'],
234     'fn_add_accommodation exists'
235 );
```

Data Output Messages Notifications

| | | |
|---|------------------------------------|---|
| | has_function text | 🔒 |
| 1 | ok 1 - fn_add_accommodation exists | |

Slika 5.24: Pokretanje testa za provjeru postojanja funkcije

```
236 --testing is function written in plpgsql
237 begin;
238 select plan(1);
239 select function_lang_is(
240     'api',
241     'fn_add_accommodation',
242     ARRAY ['json'],
243     'plpgsql',
244     'fn_add_accommodation is written using plpgsql'
245 );
```

Data Output Messages Notifications

function_lang_is
text

| | |
|---|--|
| 1 | ok 1 - fn_add_accommodation is written using plpgsql |
|---|--|

Slika 5.25: Pokretanje testa za provjeru jezika kojim je napisana funkcija

```
246 --testing if function returns json
247 begin;
248 select plan(1);
249 select function_returns(
250     'api',
251     'fn_add_accommodation',
252     'json',
253     'This function is returning json'
254 );
```

Data Output Messages Notifications

function_returns
text

| | |
|---|--|
| 1 | ok 1 - This function is returning json |
|---|--|

Slika 5.26: Pokretanje testa za provjeru povratnog tipa podatka funkcije

```
255 --testing success invocation, when adding new accommodation
256 begin;
257 select plan(2);
258 select is (api.fn_add_accommodation(
259     '{
260         "realEstateID": "IS-00141",
261         "typeID": 3,
262         "equippedID": 1,
263         "latitude": "45.083481",
264         "longitude": "13.648974",
265         "address": "Ljerke Šram 5",
266         "townID": 41,
267         "clinicID": 6,
268         "active": 1
269     }'::json)->>'success',
270     'true',
271     'New accommodation has been created'
272 );
```

Data Output Messages Notifications

Is text

| | |
|---|---|
| 1 | ok 1 - New accommodation has been created |
|---|---|

Slika 5.27: Pokretanje testa za provjeru rada same funkcije, uspijeh

```
256 begin;
257 select plan(2);
258 select is (api.fn_add_accommodation(
259     '{
260         "realEstateID": "IS-00141",
261         "typeID": 3,
262         "equippedID": 1,
263         "latitude": "45.083481",
264         "longitude": "13.648974",
265         "address": "Ljerke Šram 5",
266         "townID": 41,
267         "clinicID": 6,
268         "active": 1
269     }'::json)->>'success',
270     'true',
271     'New accommodation has been created'
272 );
273 --testing failure invocation, when adding existing accommodation
274 select is (api.fn_add_accommodation(
275     '{
276         "realEstateID": "IS-00141",
277         "typeID": 3,
278         "equippedID": 1,
279         "latitude": "45.083481",
280         "longitude": "13.648974",
281         "address": "Ljerke Šram 5",
282         "townID": 41,
283         "clinicID": 6,
284         "active": 1
285     }'::json)->>'success',
286     'false',
287     'Accommodation already exists, can not create'
288 );
```

Data Output Messages Notifications

| | | |
|---|---|---|
| | is text | 🔒 |
| 1 | ok 2 - Accommodation already exists, can not create | |

Slika 5.28: Pokretanje testa za provjeru rada same funkcije, neuspjeh

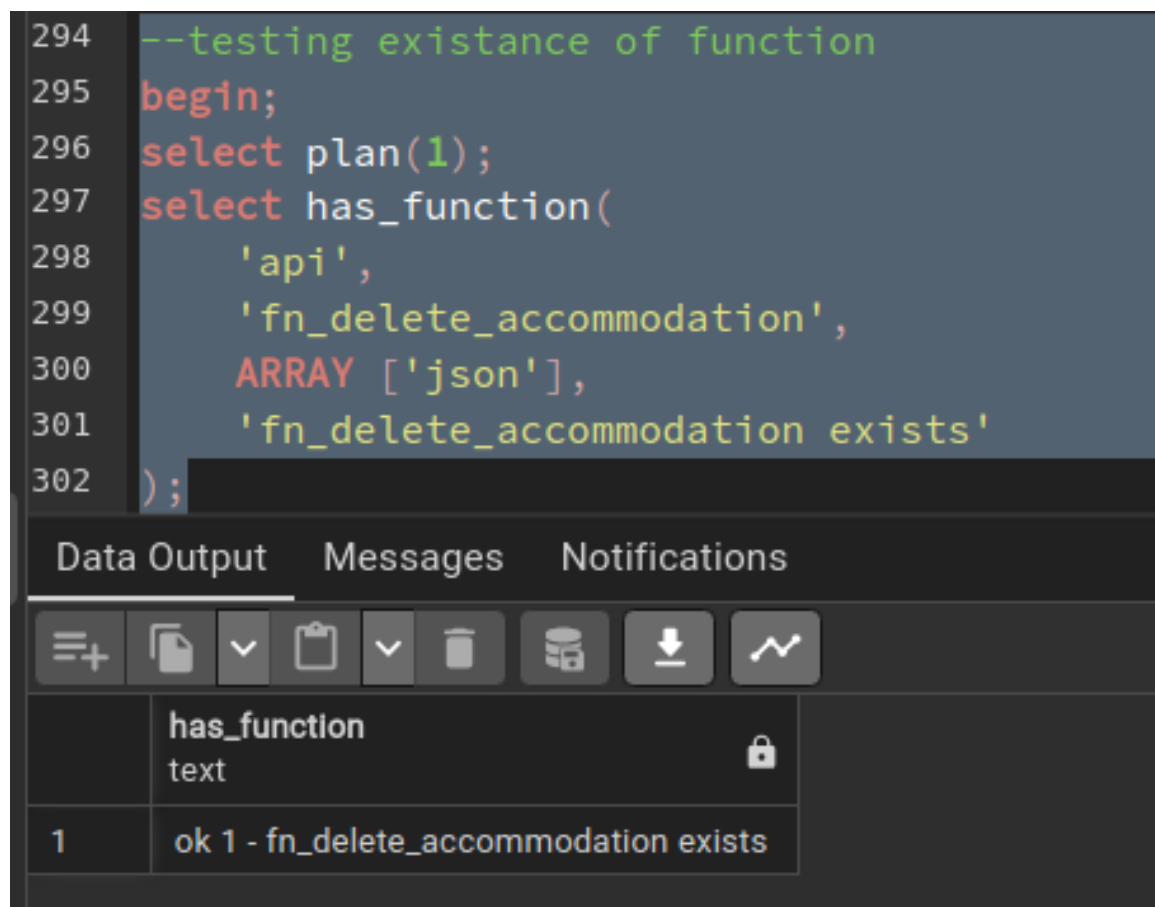
```
226 ----- UNIT TEST 4; TESTING ADDING ACCOMMODATION -----
227 --testing existence of function
228 begin;
229 select plan(1);
230 select has_function(
231     'api',
232     'fn_add_accommodation',
233     ARRAY ['json'],
234     'fn_add_accommodation exists'
235 );
236 --testing is function written in plpgsql
237 begin;
238 select plan(1);
239 select function_lang_is(
240     'api',
241     'fn_add_accommodation',
242     ARRAY ['json'],
243     'plpgsql',
244     'fn_add_accommodation is written using plpgsql'
245 );
246 --testing if function returns json
247 begin;
248 select plan(1);
249 select function_returns(
250     'api',
251     'fn_add_accommodation',
252     'json',
253     'This function is returning json'
254 );
255 --testing success invocation, when adding new accommodation
256 begin;
257 select plan(2);
258 select is (api.fn_add_accommodation(
259     '{
260         "realEstateID": "IS-00141",
261         "typeID": 3,
262         "equippedID": 1,
263         "latitude": "45.083481",
264         "longitude": "13.648974",
265         "address": "Ljerke Sram 5",
266         "townID": 41,
267         "clinicID": 6,
268         "active": 1
269     }::json)-->'success',
270     'true',
271     'New accommodation has been created'
272 );
273 --testing failure invocation, when adding existing accommodation
274 select is (api.fn_add_accommodation(
275     '{
276         "realEstateID": "IS-00141",
277         "typeID": 3,
278         "equippedID": 1,
279         "latitude": "45.083481",
280         "longitude": "13.648974",
281         "address": "Ljerke Sram 5",
282         "townID": 41,
283         "clinicID": 6,
284         "active": 1
285     }::json)-->'success',
286     'false',
287     'Accommodation already exists, can not create'
288 );
289 select * from finish(true);
290 rollback;
291 ----- UNIT TEST 4; TESTING ADDING ACCOMMODATION -----
```

Slika 5.29: Kod isptinog slučaja 4

5.2.6 Ispitni slučaj 5 - funkcionalnost brisanja smještaja

Ovaj ispitni slučaj ispituje funkcionalnost brisanja smještaja. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* ze provjeru dvaju argumenata te na osnovu njihovog podudaranje ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok <broj testa> - <opis testa>* ili *not ok <broj testa> - <opis testa>*.

```
294 --testing existance of function
295 begin;
296 select plan(1);
297 select has_function(
298     'api',
299     'fn_delete_accommodation',
300     ARRAY ['json'],
301     'fn_delete_accommodation exists'
302 );
```



| | has_function text |
|---|---------------------------------------|
| 1 | ok 1 - fn_delete_accommodation exists |

Slika 5.30: Pokretanje testa za provjeru postojanja funkcije

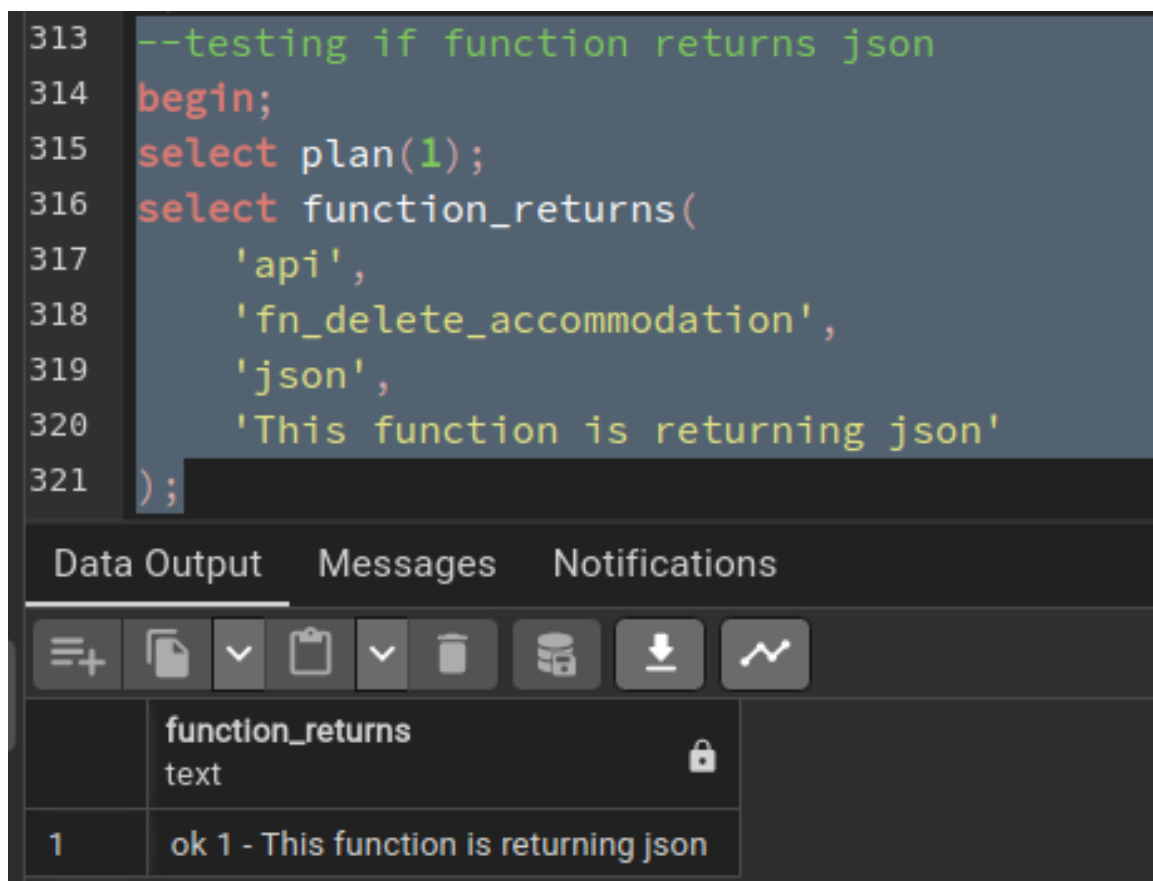

```
303 --testing is function written in plpgsql
304 begin;
305 select plan(1);
306 select function_lang_is(
307     'api',
308     'fn_delete_accommodation',
309     ARRAY ['json'],
310     'plpgsql',
311     'fn_delete_accommodation is written using plpgsql'
312 );
```

Data Output Messages Notifications

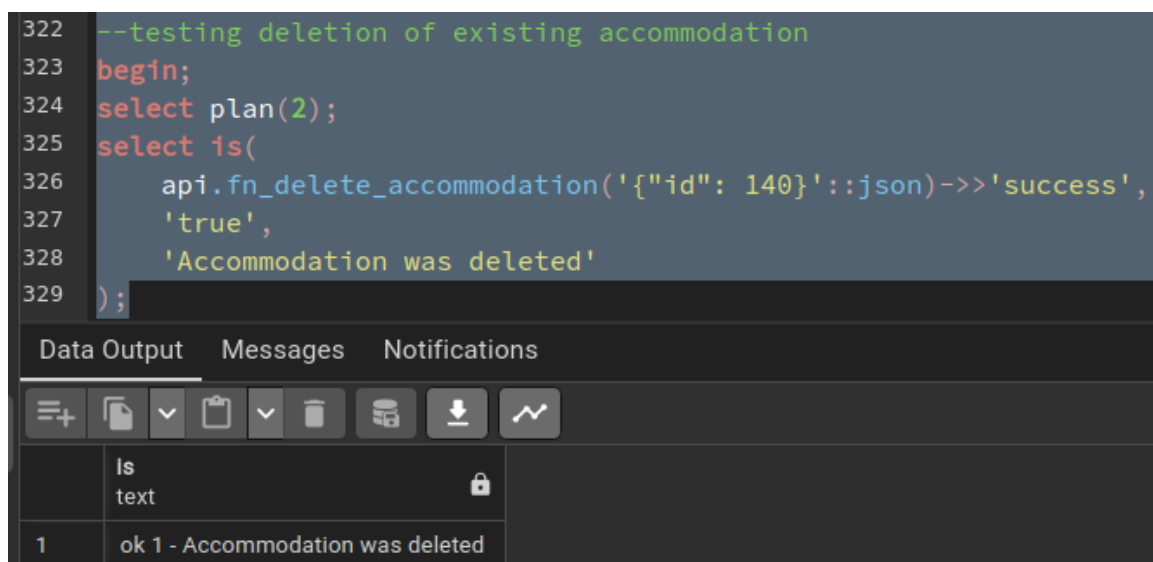
function_lang_is
text

1 ok 1 - fn_delete_accommodation is written using plpgsql

Slika 5.31: Pokretanje testa za provjeru jezika kojim je napisana funkcija



Slika 5.32: Pokretanje testa za provjeru povratnog tipa podatka funkcije



Slika 5.33: Pokretanje testa za provjeru rada same funkcije, uspijeh

```
323 begin;
324 select plan(2);
325 select is(
326     api.fn_delete_accommodation('{"id": 140}'::json)->>'success',
327     'true',
328     'Accommodation was deleted'
329 );
330 --testing deletion of nonexisting accommodation
331 select is(
332     api.fn_delete_accommodation('{"id": 140}'::json)->>'success',
333     'false',
334     'Can not delete something that does not exist'
335 );
```

Data Output Messages Notifications

| | Is | |
|---|---|--|
| | text | |
| 1 | ok 2 - Can not delete something that does not exist | |

Slika 5.34: Pokretanje testa za provjeru rada same funkcije, neuspjeh

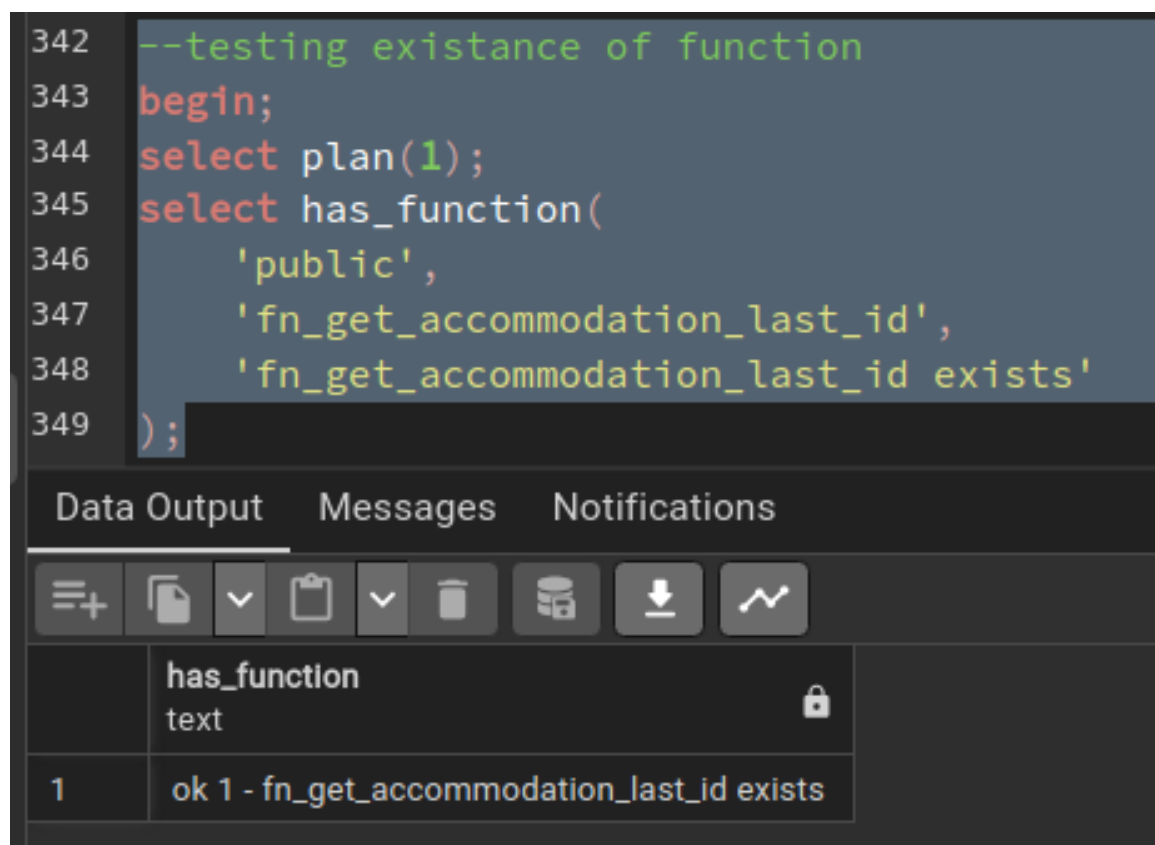
```
293 --##### UNIT TEST 5; TESTING DELETING ACCOMMODATION #####
294 --testing existence of function
295 begin;
296 select plan(1);
297 select has_function(
298     'api',
299     'fn_delete_accommodation',
300     ARRAY ['json'],
301     'fn_delete_accommodation exists'
302 );
303 --testing is function written in plpgsql
304 begin;
305 select plan(1);
306 select function_lang_is(
307     'api',
308     'fn_delete_accommodation',
309     ARRAY ['json'],
310     'plpgsql',
311     'fn_delete_accommodation is written using plpgsql'
312 );
313 --testing if function returns json
314 begin;
315 select plan(1);
316 select function_returns(
317     'api',
318     'fn_delete_accommodation',
319     'json',
320     'This function is returning json'
321 );
322 --testing deletion of existing accommodation
323 begin;
324 select plan(2);
325 select is(
326     api.fn_delete_accommodation('{"id": 140}'::json)->>'success',
327     'true',
328     'Accommodation was deleted'
329 );
330 --testing deletion of nonexisting accommodation
331 select is(
332     api.fn_delete_accommodation('{"id": 140}'::json)->>'success',
333     'false',
334     'Can not delete something that does not exist'
335 );
336
337 select * from finish(true);
338 rollback;
339 --##### UNIT TEST 5; TESTING DELETING ACCOMMODATION #####
```

Slika 5.35: Kod isptinog slučaja 5

5.2.7 Ispitni slučaj 6 - funkcionalnost dohvaćanja posljednjeg unesenog realestateid-a

Ovaj ispitni slučaj ispituje funkcionalnost dohvaćanja posljednjeg unesenog realestateid-a. Testovi ispitnog slučaja testiraju postoji li funkcija u bazi, je li funkcija napisana plpgsql jezikom te koji su ulazni i izlazni tipovi podataka funkcije. To se testira koristeći ugrađene funkcije pgTAP-a kao što su *has_function* za testiranje postoji li definirana funkcija u bazi, *function_lang_is* za testiranje je li funkcija napisana plpgsql jezikom, *function_returns* za testiranje vraća li funkcija neki tip podataka ili je tipa void, *is* za provjeru dvaju argumenata te na osnovu njihovog podudaranja ili odudaranja se izbacuje rezultat. Rezultat je uvijek u obliku jednog reda sa jednom kolonom koja može sadržavati tekst *ok <broj testa> - <opis testa>* ili *not ok <broj testa> - <opis testa>*.

```
342  --testing existence of function
343  begin;
344  select plan(1);
345  select has_function(
346      'public',
347      'fn_get_accommodation_last_id',
348      'fn_get_accommodation_last_id exists'
349  );
```



| | has_function | text |
|---|--|------|
| 1 | ok 1 - fn_get_accommodation_last_id exists | |

Slika 5.36: Pokretanje testa za provjeru postojanja funkcije

```
350 --testing is function written in plpgsql
351 begin;
352 select plan(1);
353 select function_lang_is(
354     'fn_get_accommodation_last_id',
355     'plpgsql',
356     'fn_get_accommodation_last_id is written using plpgsql'
357 );
```

Data Output Messages Notifications

function_lang_is
text

| | |
|---|--|
| 1 | ok 1 - fn_get_accommodation_last_id is written using plpgsql |
|---|--|

Slika 5.37: Pokretanje testa za provjeru jezika kojim je napisana funkcija

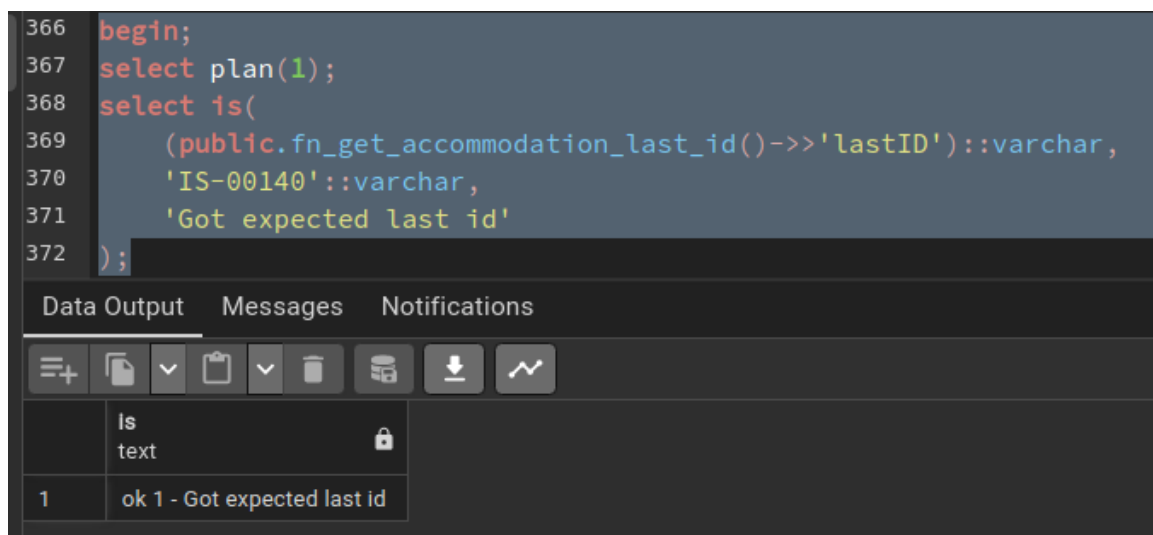
```
358 --testing if function returns json
359 begin;
360 select plan(1);
361 select function_returns(
362     'fn_get_accommodation_last_id',
363     'json',
364     'This function is returning json'
365 );
```

Data Output Messages Notifications

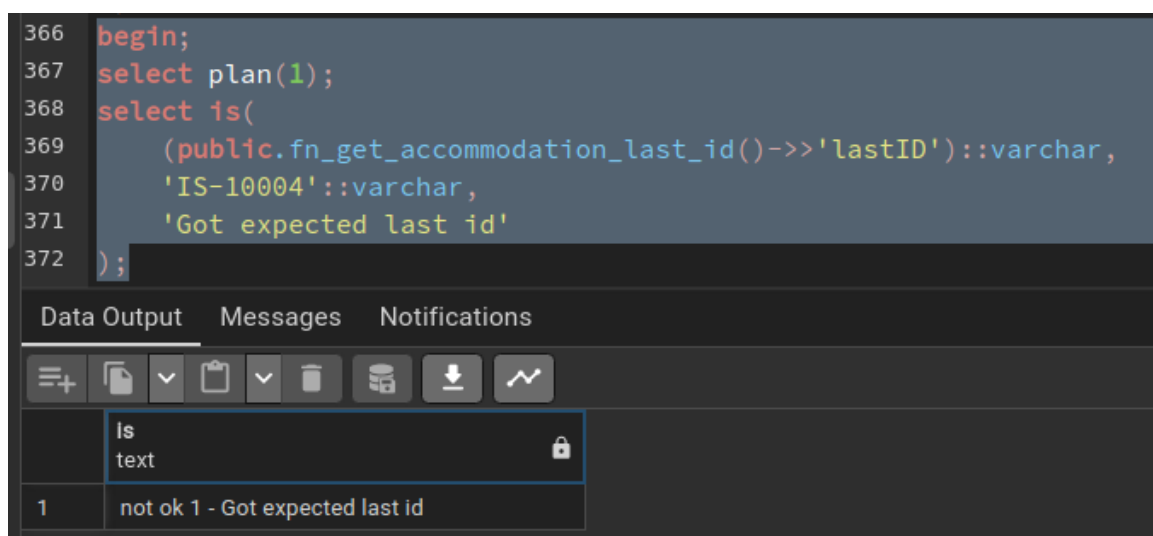
function_returns
text

| | |
|---|--|
| 1 | ok 1 - This function is returning json |
|---|--|

Slika 5.38: Pokretanje testa za provjeru povratnog tipa podatka funkcije



Slika 5.39: Pokretanje testa za provjeru rada same funkcije, uspijeh



Slika 5.40: Pokretanje testa za provjeru rada same funkcije, neuspjeh

```
341 --##### UNIT TEST 6; TESTING RETRIVAL OF LAST ACCOMMODATION ID#####
342 --testing existance of function
343 begin;
344 select plan(1);
345 select has_function(
346     'public',
347     'fn_get_accommodation_last_id',
348     'fn_get_accommodation_last_id exists'
349 );
350 --testing is function written in plpgsql
351 begin;
352 select plan(1);
353 select function_lang_is(
354     'fn_get_accommodation_last_id',
355     'plpgsql',
356     'fn_get_accommodation_last_id is written using plpgsql'
357 );
358 --testing if function returns json
359 begin;
360 select plan(1);
361 select function_returns(
362     'fn_get_accommodation_last_id',
363     'json',
364     'This function is returning json'
365 );
366 begin;
367 select plan(1);
368 select is(
369     (public.fn_get_accommodation_last_id()->>'lastID')::varchar,
370     'IS-00140'::varchar,
371     'Got expected last id'
372 );
373 begin;
374 select plan(1);
375 select is(
376     (public.fn_get_accommodation_last_id()->>'lastID')::varchar,
377     'IS-10004'::varchar,
378     'Got expected last id'
379 );
380 select * from finish()
381 rollback;
382 --##### UNIT TEST 6; TESTING RETRIVAL OF LAST ACCOMMODATION ID#####
```

Slika 5.41: Kod isptinog slučaja 6

5.3 Dijagram razmještaja

dio 2. revizije

Potrebno je umetnuti **specifikacijski** dijagram razmještaja i opisati ga. Moguće je umjesto specifikacijskog dijagrama razmještaja umetnuti dijagram razmještaja instanci, pod uvjetom da taj dijagram bolje opisuje neki važniji dio sustava.

5.4 Upute za puštanje u pogon

Aplikacija je puštena u pogon na cloud servisu Render čime smo omogućili javni pristup aplikaciji.

Konfiguracija frontenda

Konfiguracija baze/backenda

6. Zaključak i budući rad

Naš je zadatak bio napraviti web aplikaciju DentAll koja bi pomogla u svijetu zdravstvenog turizma tako što bi olakšala postupke traženja smještaja i prijevoza za liječnički tretman u pacijentu nepoznatom mjestu.

Dio kako je aplikacija gotova.

Ovaj projekt je zasigurno bio vrijedno iskustvo za sve nas iz više razloga: ovo je bio prvi put da smo koristili program git te neki od udaljenih repozitorija(u našem slučaju GitHub) kako bi smo zajedno radili na projektu, prvi put da smo radili teksni dokument u alatu Latex koji iako na prvu ruku je dosta kompliciraniji od popularnih uređivača teksta poput Microsoft Word-a, on je daleko bolji za veće i kompliciranije dokumente(poput naše dokumentacije). Ovo je također prvi ozbiljniji projekt za sve nas gdje smo se trebali sami snaći i organizirati, uz naravno pomoć asistenata i demonstratora ako nam nešto nije bilo jasno u vezi zadatka. Iz prve smo ruke vidjeli koji su sve koraci potrebni kako bi se od ideje došlo do gotove web stranice. Tijekom projekta nismo imali neke značajnije probleme vrijedne spomena.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

| | | |
|------|---|----|
| 2.1 | Mogućnosti povezivanja klijenata i pružatelja koje nudi web stranica "MedicalTourism.com" | 6 |
| 3.1 | Dijagram obrasca uporabe, osnovne funkcionalnosti | 21 |
| 3.2 | Dijagram obrasca uporabe, funkcionalnosti upravljanja korisnicima | 22 |
| 3.3 | Dijagram obrasca uporabe, funkcionalnosti upravljanja smještajima | 23 |
| 3.4 | Dijagram obrasca uporabe, funkcionalnosti upravljanja prijevoznici- cima | 24 |
| 3.5 | Dijagram obrasca uporabe, funkcionalnosti upravljanja vozilima | 25 |
| 3.6 | Dijagram obrasca uporabe, funkcionalnosti upravljanja pacijentima | 26 |
| 3.7 | Dijagram obrasca uporabe, pridjeljivanje i obavješćavanje | 27 |
| 3.8 | Sekvencijski dijagram za UC2 | 28 |
| 3.9 | Sekvencijski dijagram za UC19 | 29 |
| 3.10 | Sekvencijski dijagram za UC21 | 30 |
| 3.11 | Sekvencijski dijagram za UC22 | 31 |
| 4.1 | Sheme baze podataka | 44 |
| 4.2 | Dijagram razreda Controller | 45 |
| 4.3 | Dijagram razreda Model | 46 |
| 5.1 | Pokretanje testa za provjeru postojanja funkcije | 53 |
| 5.2 | Pokretanje testa za provjeru jezika kojim je napisana funkcija | 54 |
| 5.3 | Pokretanje testa za provjeru povratnog tipa podatka funkcije | 55 |
| 5.4 | Pokretanje testa za provjeru rada same funkcije, uspijeh | 55 |
| 5.5 | Pokretanje testa za provjeru rada same funkcije, neuspjeh | 56 |
| 5.6 | Pokretanje testa za provjeru rada same funkcije, neuspjeh | 56 |
| 5.7 | Kod isptinog slučaja 1 | 57 |
| 5.8 | Pokretanje testa za provjeru postojanja funkcije | 58 |
| 5.9 | Pokretanje testa za provjeru jezika kojim je napisana funkcija | 59 |
| 5.10 | Pokretanje testa za provjeru povratnog tipa podatka funkcije | 60 |
| 5.11 | Pokretanje testa za provjeru rada same funkcije, uspijeh | 61 |

| | |
|---|----|
| 5.12 Pokretanje testa za provjeru rada same funkcije, neuspjeh | 62 |
| 5.13 Pokretanje testa za provjeru automatskog kreiranja vjerodajnica za novog administratora | 63 |
| 5.14 Pokretanje testa za provjeru automatskog dodjeljivanja uloge/uloga za novog administratora | 64 |
| 5.15 Kod ispitnog slučaja 2 | 66 |
| 5.16 Pokretanje testa za provjeru postojanja funkcije | 67 |
| 5.17 Pokretanje testa za provjeru jezika kojim je napisana funkcija | 68 |
| 5.18 Pokretanje testa za provjeru povratnog tipa podatka funkcije | 69 |
| 5.19 Pokretanje testa za provjeru rada same funkcije, uspijeh | 69 |
| 5.20 Pokretanje testa za provjeru rada same funkcije, neuspjeh | 70 |
| 5.21 Pokretanje testa za provjeru automatskog brisanja vjerodajnica za izbrisanog administratora | 70 |
| 5.22 Pokretanje testa za provjeru automatskog brisanja pridjeljenih uloga za obrisano administratora | 71 |
| 5.23 Kod ispitnog slučaja 3 | 72 |
| 5.24 Pokretanje testa za provjeru postojanja funkcije | 73 |
| 5.25 Pokretanje testa za provjeru jezika kojim je napisana funkcija | 74 |
| 5.26 Pokretanje testa za provjeru povratnog tipa podatka funkcije | 75 |
| 5.27 Pokretanje testa za provjeru rada same funkcije, uspijeh | 76 |
| 5.28 Pokretanje testa za provjeru rada same funkcije, neuspjeh | 77 |
| 5.29 Kod ispitnog slučaja 4 | 78 |
| 5.30 Pokretanje testa za provjeru postojanja funkcije | 79 |
| 5.31 Pokretanje testa za provjeru jezika kojim je napisana funkcija | 80 |
| 5.32 Pokretanje testa za provjeru povratnog tipa podatka funkcije | 81 |
| 5.33 Pokretanje testa za provjeru rada same funkcije, uspijeh | 81 |
| 5.34 Pokretanje testa za provjeru rada same funkcije, neuspjeh | 82 |
| 5.35 Kod ispitnog slučaja 5 | 83 |
| 5.36 Pokretanje testa za provjeru postojanja funkcije | 84 |
| 5.37 Pokretanje testa za provjeru jezika kojim je napisana funkcija | 85 |
| 5.38 Pokretanje testa za provjeru povratnog tipa podatka funkcije | 85 |
| 5.39 Pokretanje testa za provjeru rada same funkcije, uspijeh | 86 |
| 5.40 Pokretanje testa za provjeru rada same funkcije, neuspjeh | 86 |
| 5.41 Kod ispitnog slučaja 6 | 87 |

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum:18.10.2023.
- Prisustvovali: Karlo Baljak, Luka Kokić, Ian Marković, Mateo Martić, Mislav Matić, Bruno Milaković, Teo Musa
- Teme sastanka:
 - Međusobno upoznavanje
 - Rasprava o projektnom zadatku

2. sastanak

- Datum:25.10.2023.
- Prisustvovali: Karlo Baljak, Luka Kokić, Ian Marković, Mateo Martić, Mislav Matić, Bruno Milaković, Teo Musa
- Teme sastanka:
 - Podjela zadataka na projektu
 - Rasprava o alatima koje ćemo koristiti

3. sastanak

- Datum:10.11.2023.
- Prisustvovali: Karlo Baljak, Luka Kokić, Ian Marković, Mateo Martić, Mislav Matić, Bruno Milaković, Teo Musa
- Teme sastanka:
 - Rasprava o finalizaciji finijih detalja vezano uz obrasce uporabe i dijagrame te njihova finalizacija
 - Raspodjela zadataka koji trebaju biti obavljeni do prve predaje

4. sastanak

- Datum:6.12.2023.
- Prisustvovali: Karlo Baljak, Luka Kokić, Ian Marković, Mateo Martić, Mislav Matić, Bruno Milaković, Teo Musa
- Teme sastanka:

- Podjela zadataka na projektu u drugom ciklusu

5. sastanak

- Datum: 11.1.2024.
- Prisustvovali: Karlo Baljak, Luka Kokić, Ian Marković, Mislav Matić, Bruno Milaković, Teo Musa
- Teme sastanka:
 - Usklađivanje frontend-a i backend-a
 - Rasprava o izgledu stranice i o tome kako su podstranice organizirane
 - Raspodjela preostalih zadataka

Tablica aktivnosti

| | Luka Kokić | Karlo Baljak | Ian Marković | Mateo Martić | Mislav Matić | Bruno Milaković | Teo Musa |
|----------------------------------|------------|--------------|--------------|--------------|--------------|-----------------|----------|
| Upravljanje projektom | 7 | | | | | | |
| Opis projektnog zadatka | | | 4 | | | | |
| Funkcionalni zahtjevi | | | 3 | | | | |
| Opis pojedinih obrazaca | | | 8 | | | | 8 |
| Dijagram obrazaca | | | | | 7 | | |
| Sekvencijski dijagrami | | | | | | 11 | |
| Opis ostalih zahtjeva | | | 1 | | | | |
| Arhitektura i dizajn sustava | | 3 | | | | | |
| Baza podataka | | 15 | | | | | |
| Dijagram razreda | | 5 | | 5 | | | |
| Dijagram stanja | | | | | | | |
| Dijagram aktivnosti | | | | | | | |
| Dijagram komponenti | | | | | | | |
| Korištene tehnologije i alati | | | | | | | |
| Ispitivanje programskog rješenja | | | | | | | |
| Dijagram razmještaja | | | | | | | |
| Upute za puštanje u pogon | | | | | | | |
| Dnevnik sastajanja | 1 | | | | | | |
| Zaključak i budući rad | | | | | | | |
| Popis literature | | | | | 1 | | |

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

| | Luka Kokić | Karlo Baljak | Ian Marković | Mateo Martić | Mislav Matić | Bruno Milaković | Teo Musa |
|--|------------|--------------|--------------|--------------|--------------|-----------------|----------|
| Ukupno dokumentacija | 8 | 23 | 16 | 5 | 8 | 11 | 8 |
| Sastanci | 14 | 14 | 14 | 12 | 12 | 12 | 12 |
| Istraživanje informacija i tehnologija | 10 | 13 | 5 | 4 | 4 | 7 | 4 |
| Deployment | | 10 | | | | | |
| Izrada baze podataka | | 7 | | | | | |
| Spajanje s bazom podataka | | 5 | | | | | |
| Backend | 2 | 10 | | | | | |
| Frontend | 3 | 5 | | | | | |
| Ukupno razvoj projekta | 37 | 87 | 35 | 21 | 24 | 30 | 24 |

Dijagrami pregleda promjena

dio 2. revizije

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