

Programsko inženjerstvo

Ak. god. 2023./2024.

DentAll

Dokumentacija, Rev. 2

Grupa: *Tintilinici*

Voditelj: *Neven Lukić*

Datum predaje: 19. 01. 2024.

Nastavnik: *Goran Rajić*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	26
3.2 Ostali zahtjevi	33
4 Arhitektura i dizajn sustava	34
4.1 Baza podataka	35
4.1.1 Opis tablica	35
4.1.2 Dijagram baze podataka	41
4.2 Dijagram razreda	42
4.3 Dijagram stanja	47
4.4 Dijagram aktivnosti	48
4.5 Dijagram komponenti	50
5 Implementacija i korisničko sučelje	51
5.1 Korištene tehnologije i alati	51
5.2 Ispitivanje programskog rješenja	52
5.2.1 Ispitivanje komponenti	52
5.2.2 Ispitivanje sustava	56
5.3 Dijagram razmještaja	62
5.4 Upute za puštanje u pogon	63
6 Zaključak i budući rad	65
Popis literature	66
Indeks slika i dijagonama	68

Dodatak: Prikaz aktivnosti grupe

69

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak.	Neven Lukić	26.10.2023.
0.2	Započela opis projektnog zadatka.	Karla Šmuk	01.11.2023.
0.3	Dodani članovi tima i prva dva sastanaka.	Neven Lukić	1.11.2023.
0.3.1	Proširen opis projektnog zadatka. Dodani funkcionalni zahtjevi.	Karla Šmuk	04.11.2023.
0.4	Dodani <i>Use Case</i> dijagrami	Karla Šmuk	16.11.2023.
0.5	Dodan opis baze podataka, svih tablica i ER dijagram baze podataka.	Neven Lukić	16.11.2023.
0.6	Dodani opisi obrazaca upotrebe.	Karla Pišonić	16.11.2023.
0.7	Dodani ostali zahtjevi.	Filip Buljan	16.11.2023.
0.8	Dodani dijagrami razreda.	Roko Gligora	16.11.2023.
0.9	Dodani sekvencijski dijagrami. Promijenjeni neki dijelovi opisa projektnog zadatka.	Karla Šmuk	17.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Neven Lukić	17.11.2013.
1.1	Započela Zaključak i budući rad.	Karla Šmuk	16.01.2024.
1.1.1	Završen Zaključak i budući rad.	Karla Šmuk	17.01.2024.
1.2	Započela Korištene tehnologije i alati	Karla Pišonić	17.01.2024.
1.2.1	Dovršeno uređivanje Korištene tehnologije i alati	Karla Pišonić	17.01.2024.
1.3	Stvarno stanje implementacije - dijagrami razreda.	Roko Gligora	17.01.2024.
1.4	Dodan dijagram aktivnosti.	Karla Šmuk	18.01.2024.
1.5	Dodan dijagram komponenti.	Karla Pišonić	18.01.2024.
1.6	Ažurirana baza podataka.	Roko Gligora	18.01.2024.
1.7	Dijagram razmještaja.	Filip Buljan, Roko Gligora	18.01.2024.
1.8	Dodan testovi za ispitivanje sustava.	Neven Lukić	18.01.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.9	Upute za puštanje u pogon	Filip Buljan, Roko Gligora.	18.01.2024.
1.10	Dodani dijagrami promjena s GitHub-a.	Neven Lukić	18.01.2024.
1.10	Dodane upute za puštanje frontend-a u pogon.	Neven Lukić	18.01.2024.
1.11	Ažuriran ER dijagram.	Roko Gligora	18.01.2024.
1.12	Dodano ispitivanje komponenti	Maksim Madžar, Roko Gligora	19.01.2024.
1.13	Dodan dijagram stanja	Ante Vujčić, Roko Gligora	19.01.2024.
2.0	Verzija konačne predaje	Neven Lukić	19.01.2024.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku potporu za stvaranje web aplikacije "DentAll" koja će omogućiti učinkovito upravljanje smještajem i prijevozom korisnika zdravstvenog turizma.

Porastom zdravstvenog turizma, zdravstvene ustanove trude se privući korisnike nudeći im cjelovite usluge, uključujući smještaj i prijevoz. U mnogim slučajevima skuplje zdravstvene usluge u državama od kuda strani korisnici dolaze, potiču potrebu za pretragom usluga u drugim državama. Međutim, unatoč pristupačnjim troškovima medicinske usluge, korisnici se suočavaju s mnogim drugim izazovima koji obeshrabruju njihovu odluku za potragom medicinske usluge izvan svoje države. Spomenuti problemi su troškovi putovanja, udaljenost, osjećaj nesigurnosti i nedostatak poznавања destinacije u kojoj se zdravstvena usluga nudi.

Naglasak se sve više stavlja na potrebu razvoja rješenja koje će omogućiti učinkovitu, brzu i jednostavnu koordinaciju smještaja i prijevoza. S obzirom na sve izazove koje bi korisnik trebao proći da se odluči za zdravstvenu uslugu u inozemstvu, zdravstvenim ustanovama nije dovoljno imati samo financijsku prednost već i mnoge druge. Ideja o izradi aplikacije za pomoć potencijalnim korisnicima usluga zdravstvenog turizma u pronašlasku smještaja i prijevoza je ključna. Nije dovoljno privući korisnike samo povoljnim cijenama, nego je bitno pružiti im sigurnosti i udobnost tijekom boravka na novoj destinaciji. Organizacija smještaja i prijevoza uvelike bi povećala atraktivnost zdravstvenog turizma. Ovaj pristup omogućio bi korisnicima da se bolje informiraju i pripreme za njihovu medicinsku uslugu, bez potrebe za brigom o putovanju i smještaju. Organizacija smještaja i prijevoza do zdravstvenih ustanova korisnicima bi uzrokovala minimalan stres i smanjivala njihovu izgubljenost. To je ključni korak u motiviranju potencijalnih korisnika da se odluče za ovu uslugu.

Od ovakve aplikacije koristi bi imali korisnici, zdravstvene ustanove, prijevoznici te iznajmljivači smještaja. Aplikacija bi zdravstvenim ustanovama omogućila cjelovite usluge pacijentima povećavajući privlačnost njihove ponude. Prijevoznicima i iznajmljivačima smještaja pomogla bi u upravljanju svojim kapacitetima i vožnjama. Najbitniji korisnici imali bi najbolje moguće iskustvo jer bi im bilo

olakšano rezerviranje smještaja i prijevoza.

U aplikaciji postoje tri uloge korisnika:

- smještajni administrator
- administrator prijevoznih usluga
- korisnički administrator

Ulaskom u aplikaciju neprijavljeni korisnik dolazi na početnu stranicu. Može se odlučiti za prijavu u sustav te odabratи opciju "Prijava". Korisnika se preusmjerava na stranicu za prijavu te tamo upisuje svoje podatke e-mail i lozinku. Dalje ga se preusmjerava ovisno o njegovoj ulozi ili ulogama.

Smještajni administrator

Preusmjerava ga se na stranicu liste svih unesenih smještaja. Odabirom "Dodaj novi smještaj" može stvoriti novi smještaj. Također pritiskom na određeni smještaj preusmjerava ga se na stranicu detalja o tom smještaju. Tu se može vidjeti i prikaz smještaja na geografskoj karti. Odabirom "Uredi podatke" administrator može uređivati osnovne podatke te obrisati smještaj odabirom "Ukloni ovaj smještaj" nakon što dobije poruku potvrde kojom potvrđuje svoju odluku. Za kreiranje smještaja potrebni su podaci:

- adresa smještaja
- vrsta smještaja
- kategorija smještaja
- vremenski period dostupnosti (dostupno od, dostupno do)
- lokacija (u koordinatama) za grafički prikaz geografskog položaja na karti

Ima najveće ovlasti te može definirati druge korisnike te im dodjeljivati različite uloge (jedan korisnik može imati i više uloga). Pritiskom na "Upravljanje administratorima" preusmjerava ga se na stranicu s listom svih administratora. Za kreiranje administratora potrebni su podaci:

- ime
- prezime
- e-mail
- uloga/e

Opcijom "Dodaj novog administratora" može unositi nove korisnike. Odabirom opcije "Promijeni ulogu" korisniku se može dodati nova uloga ili promijeniti postojeća. Također, odabirom "Izbriši administratora", administrator ima mogućnost obrisati odabranog korisnika nakon što dobije poruku potvrde kojom potvrđuje svoju odluku.

Administrator prijevoznih usluga

Preusmjerava ga se na stranicu liste svih unesenih prijevoznika. Odabirom "Dodaj novog prijevoznika" može stvoriti novog prijevoznika. Za kreiranje prijevoznika potrebni su podaci:

- naziv prijevoznika
- e-mail
- broj telefona

Također pritiskom na određenog prijevoznika preusmjerava ga se na stranicu koja prikazuje listu unesenih transportnih vozila za tog prijevoznika. Za kreiranje vozila potrebni su podaci:

- tip vozila
- kapacitet

Odabirom "Uredi podatke" administrator može uređivati osnovne podatke. Osim toga, odabirom opcije "Ukloni vozilo", administrator ima mogućnost izbrisati odabранo vozilo nakon što dobije poruku potvrde kojom potvrđuje svoju odluku.

Korisnički administrator

Preusmjerava ga se na stranicu liste unesenih pacijenata. Odabirom opcije "Dodaj novog pacijenta" omogućuje se unos novih pacijenata. Za kreiranje korisnika potrebni su podaci:

- ime
- prezime
- PIN
- broj telefona
- e-mail

Također, on može napraviti zahtjev za smještaj za određenog pacijenta. Za kreiranje zahtjeva za smještaj potrebni su podaci:

- datum i vrijeme dolaska i odlaska
- tip smještaja
- kategoriju smještaja

Na kraju završenog korisnik se može odjaviti pritiskom na gumb “Odjava” te se vraća na početnu stranicu.

Pacijent nema direktnu vezu s aplikacijom. Detalji o tretmanima ne unose se ručno u aplikaciju, već postoji umjetno ispitno sučelje koje komunicira s aplikacijom za evidenciju medicinskih usluga za koju se pretpostavlja da je već razvijena.

Nakon unosa novog pacijenta i potrebnih informacija aplikacija dodjeljuje raspoloživi smještaj pacijentu te je označuje kao zauzetu u periodu njegove medicinske usluge. Aplikacija periodički provjerava status medicinske usluge komunikacijom s aplikacijom medicinskih usluga. Ako aplikacija primi odgovor od aplikacije medicinskih usluga da je plan medicinskih usluga zaključan, to znači da su svi medicinski termini pacijenata potvrđeni te se tada može početi organizirati prijevoz za te termine. Potrebno je i označiti dodijeljene prijevoznike zauzetima u tim terminima.

Nakon završetka ukupnog plana puta, pacijentu će biti poslan e-mail o detaljima plana puta. Također, poruke će biti poslane i svakom prijevozniku s kontaktnim podacima korisnika te podacima o vremenima i adresama smještaja.

Jedna od mogućih nadogradnji ovoga zadatka bila bi dodavanje pristupa korisničkim iskustvima, točnije, omogućiti ocjenjivanja smještaja i prijevoza te pisanje recenzija kako bi korisnici mogli što bolje procijeniti kvalitetu usluga.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik (naručitelj)
2. Korisnici
 - (a) Smještajni administrator
 - (b) Administrator prijevoznih usluga
 - (c) Korisnički administrator
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neprijavljeni korisnik (inicijator) može:
 - (a) se prijaviti u sustav za što su mu potrebni e-mail i lozinka
2. Smještajni administrator (inicijator) može:
 - (a) unositi nove smještaje
 - (b) mijenjati osnovne podatke smještaja
 - (c) brisati smještaje
 - (d) unositi nove korisnike
 - (e) dodjeljivati uloge korisnicima
3. Administrator prijevoznih usluga (inicijator) može:
 - (a) unositi nove prijevoznike
 - (b) mijenjati podatke prijevoznika
 - (c) brisati prijevoznike
 - (d) unositi vozila za prijevoznike
 - (e) brisati vozila
4. Korisnički administrator (inicijator) može:

(a) unositi pacijente

5. Baza podataka (sudionik) može:

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o pacijentu
- (c) pohranjuje sve podatke o medicinskoj usluzi
- (d) pohranjuje sve podatke o smještaju
- (e) pohranjuje sve podatke o prijevozniku i vozilima
- (f) pohranjuje sve podatke o rezerviranju smještaja
- (g) pohranjuje sve podatke o rezerviranju prijevoznika

6. Aplikacija medicinske usluge (sudionik):

- (a) odgovorna za prikupljanje podatak o medicinskim tretmanima te naša aplikacija putem sučelja dohvata podatke o tretmanima korisnika

3.1.1 Obrasci uporabe

UC1 - prijava

- **Glavni sudionik:** neprijavljeni korisnik
- **Cilj:** dobiti pristup korisničkom sučelju
- **Sudionici:** baza podataka
- **Opis osnovnog tijeka:**
 1. Na zahtjev se otvara stranica administratoru
 2. Administrator prenosi podatke korisnika u sustav
 3. Pohranjuju se promjene u bazi podataka

UC2 - odjava

- **Glavni sudionik:** administrator
- **Cilj:** odjaviti korisnika iz sustava
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik mora postojati u sustavu, odnosno biti već u bazi podataka; administrator mora biti prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju odjave korisnika iz sustava
 2. Korisnik je odjavljen iz sustava i pohranjene su promjene

UC3 - pregled svih administratora

- **Glavni sudionik:** smještajni administrator
- **Cilj:** dobiti uvid u podatke svih administratora
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Smještajni administrator odabire opciju pregleda svih administratora kojima je on nadležan - korisničkih administratora i administratora prijevoznih usluga
 2. Otvara se popis svih administratora

UC4 - unos novog administratora

- **Glavni sudionik:** smještajni administrator
- **Cilj:** dodati administratora

- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Smještajni administrator odabere opciju dodavanja novog administratora
 2. Otvara se prozor za unos podataka
 3. Smještajni administrator upiše podatke o novom administratoru
 4. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Upisano je neispravno korisničko ime administratora
 1. Sustav obaveštava smještajnog administratora o neuspjelom upisu

UC5 - brisanje administratora

- **Glavni sudionik:** smještajni administrator
- **Cilj:** obrisati administratora
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Smještajni administrator odabere opciju brisanja administratora
 2. Otvara se prozor za unos podataka
 3. Smještajni administrator upiše podatke o administratoru
 4. Smještajni administrator odabere opciju 'Izbriši'
 5. Administrator se uklanja iz baze podataka

UC6 - dodjeljivanje uloga

- **Glavni sudionik:** smještajni administrator
- **Cilj:** dodijeliti ulogu administratoru
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Smještajni administrator otvara popis svih administratora
 2. Smještajni administrator odabere administratora kojem želi promijeniti ulogu

3. Otvori se prozor s podatcima administratora
 4. Smještajni administrator odabere jednu od ponuđenih uloga (korisnički administrator ili administrator prijevoznih usluga)
 5. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Upisano je neispravno korisničko ime administratora
 1. Sustav obaveštava smještajnog administratora o neuspjelom dodjeljivanju uloge administratoru

UC7 - pregled svih smještaja

- **Glavni sudionik:** smještajni administrator
- **Cilj:** pregledati sve smještaje
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora, postoji barem jedan smještaj unesen u sustav
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregleda svih smještaja
 2. Prikaže se popis svih smještaja koji postoje u sustavu

UC8 - pregled smještaja

- **Glavni sudionik:** smještajni administrator
- **Cilj:** pregledati traženi smještaj
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregleda svih smještaja
 2. Prikaže se popis svih smještaja koji postoje u sustavu
 3. Administrator odabere iz popisa traženi smještaj

UC9 - unos novog smještaja

- **Glavni sudionik:** smještajni administrator
- **Cilj:** dodati novi smještaj
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora

- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju dodavanja smještaja
 2. Otvara se prozor za unos podataka o novom smještaju
 3. Administrator unese podatke o smještaju
 4. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Upisano je neispravno ime smještaja
 1. Sustav obaveštava smještajnog administratora o neuspjelom upisu

UC10 - izmjena podataka o smještaju

- **Glavni sudionik:** smještajni administrator
- **Cilj:** izmijeniti podatke o smještaju
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregleda svih smještaja
 2. Prikaže se popis svih smještaja koji postoje u sustavu
 3. Administrator odabere iz popisa traženi smještaj
 4. Otvori se stranica s podatcima o smještaju
 5. Administrator izmjeni podatke o smještaju
 6. Administrator odabere opciju 'Spremi promjene'
 7. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Traženi smještaj ne postoji u sustavu
 1. Sustav obaveštava smještajnog administratora da smještaj ne postoji u sustavu

UC11 - brisanje smještaja

- **Glavni sudionik:** smještajni administrator
- **Cilj:** obrisati smještaj
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju brisanja smještaja

2. Otvara se prozor za unos podataka
3. Administrator unese ime smještaja i obriše ga
4. Smještaj se uklanja iz baze podataka

UC12 - prikaz smještaja na karti

- **Glavni sudionik:** smještajni administrator
- **Cilj:** prikazati smještaj na karti
- **Sudionici:** OpenMaps
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga smještajnog administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregleda svih smještaja
 2. Prikaže se popis svih smještaja koji postoje u sustavu
 3. Administrator odabere iz popisa traženi smještaj
 4. Administrator odabere opciju prikaza smještaja na karti
 5. Otvori se prozor gdje se vidi lokacija smještaja na karti
- **Opis mogućih odstupanja:**
 - 3.a Traženi smještaj ne postoji u sustavu
 1. Sustav obaveštava smještajnog administratora da smještaj ne postoji u sustavu
 - 4.a Smještaj se ne može locirati na karti
 1. Sustav obaveštava smještajnog administratora da smještaj nije moguće locirati

UC13 - pregled prijevoznika

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** pregledati sve prijevoznike
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga; postoji barem jedan prijevoznik u sustavu
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregleda svih prijevoznika
 2. Prikaže se popis svih prijevoznika
 3. Administrator odabere traženog prijevoznika

UC14 - izmjena podataka prijevoznika

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** izmijeniti podatke traženog administratora
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga; postoji barem jedan prijevoznik u sustavu
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregleda svih prijevoznika
 2. Prikaže se popis svih prijevoznika koji postoje u sustavu
 3. Administrator odabere iz popisa traženog prijevoznika
 4. Otvori se stranica s podatcima prijevoznika
 5. Administrator izmjeni podatke o prijevozniku
 6. Administrator odabere opciju 'Spremi promjene'
 7. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Traženi prijevoznik ne postoji u sustavu
 1. Sustav obavještava administratora da odabrani prijevoznik ne postoji u sustavu

UC15 - unos novog vozila

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** unijeti podatke o novom vozilu
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju dodavanja novog vozila
 2. Otvara se prozor za unos podataka o novom vozilu
 3. Administrator unese podatke o vozilu
 4. U bazu podataka se pohrani promjena

UC16 - brisanje vozila

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** obrisati vozilo
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga

- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju brisanja vozila
 2. Otvara se prozor za unos podataka
 3. Administrator unese ime vozila i obriše ga
 4. Vozilo se uklanja iz baze podataka

UC17 - pregled svih prijevoznika

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** dobiti uvid u sve prijevoznike
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregleda svih prijevoznika
 2. Prikaže se popis svih prijevoznika

UC18 - brisanje prijevoznika

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** obrisati prijevoznika
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju brisanja prijevoznika
 2. Otvara se prozor za unos podataka
 3. Administrator unese ime prijevoznika i obriše ga
 4. Prijevoznik se uklanja iz baze podataka

UC19 - unos novog prijevoznika

- **Glavni sudionik:** administrator prijevoznih usluga
- **Cilj:** unijeti novog prijevoznika
- **Sudionici:** baza podataka
- **Preduvjet:** administrator je prijavljen i dodijeljena mu je uloga administratora prijevoznih usluga
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju dodavanja prijevoznika

2. Otvara se prozor za unos podataka o novom prijevozniku
 3. Administrator unese podatke o prijevozniku
 4. Administrator odabere opciju 'Dodaj prijevoznika'
 5. U bazu podataka se pohrani promjena
- **Opis mogućih odstupanja:**
 - 3.a Upisano je neispravno ime prijevoznika
 1. Sustav obaveštava administratora o neuspjelom unosu

UC20 - unos novog pacijenta

- **Glavni sudionik:** korisnički administrator
- **Cilj:** unijeti podatke o novom pacijentu
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju unosa novog pacijenta
 2. Otvara se prozor za upis podataka novog pacijenta
 3. Administrator upiše podatke o novom pacijentu
 4. Pohranjuju se promjene u bazu podataka

UC21 - dohvaćanje termina pacijenta

- **Glavni sudionik:** korisnički administrator
- **Cilj:** dohvatiti podatke o terminu medicinskih usluga
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Administrator otvara aplikaciju medicinskih usluga
 2. Administrator iz aplikacije medicinskih usluga preuzima podatke preferiranog dolaska i odlaska pacijenta
 3. Podatci o terminu se upisuju u sustav
 4. Pohranjuju se promjene u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Za navedeno razdoblje dolaska i odlaska nema dostupnih termina
 1. Sustav javlja administratoru da u navedenom razdoblju nema dostupnih termina

UC22 - dodjeljivanje smještaja pacijentu

- **Glavni sudionik:** korisnički administrator
- **Cilj:** +dodijeliti smještaj pacijentu
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Smještajni administrator unosi u aplikaciju podatke o raspoloživom smještaju
 2. Administrator unosi korisnika u sustav
 3. Administrator pridjeljuje raspoloživu smještajnu jedinicu
 4. Smještajna jedinica se označava kao zauzeta u danom periodu

UC23 - dodjeljivanje prijevoznika terminima

- **Glavni sudionik:** korisnički administrator
- **Cilj:** dodijeliti prijevoznika
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Administrator prijevoznih usluga u aplikaciju unosi podatke o prijevoznicima
 2. Administrator unosi korisnika u sustav
 3. Zaključavanje plana medicinskih usluga s listom termina
 4. Administrator pridjeljuje raspoložive prijevoznike za svaki od termina
 5. Administrator označava prijevoznike zauzete u tim terminima

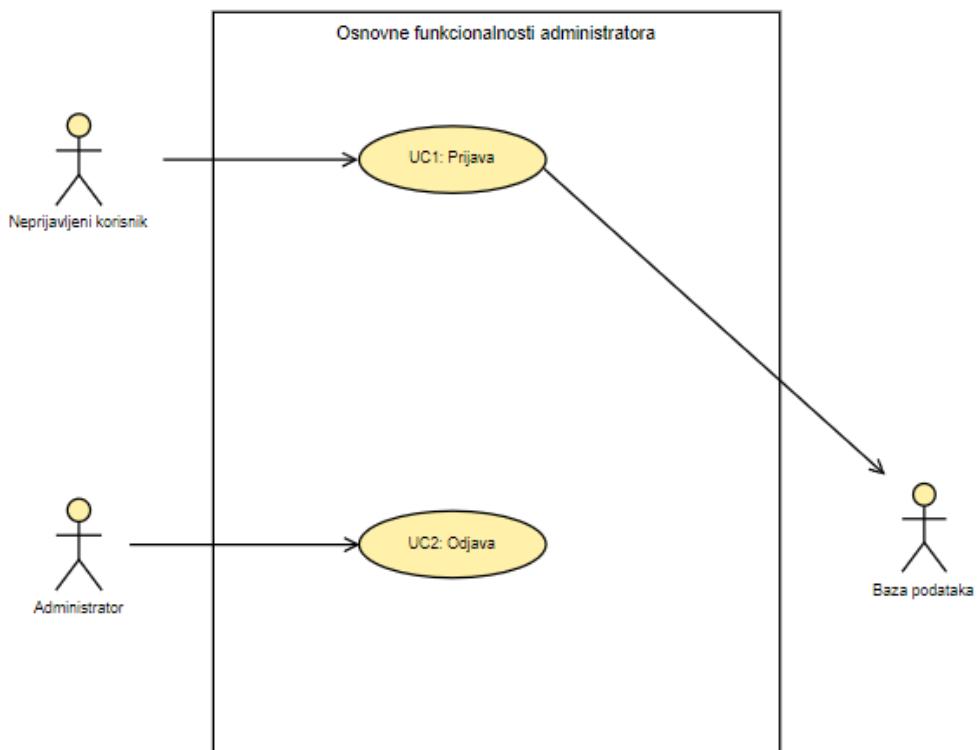
UC24 - slanje e-maila prijevozniku

- **Glavni sudionik:** korisnički administrator
- **Cilj:** poslati e-mail s podatcima korisnika prijevozniku
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Plan o medicinskim uslugama je zaključan
 2. Aplikacija dohvata e-mail adresu prijevoznika

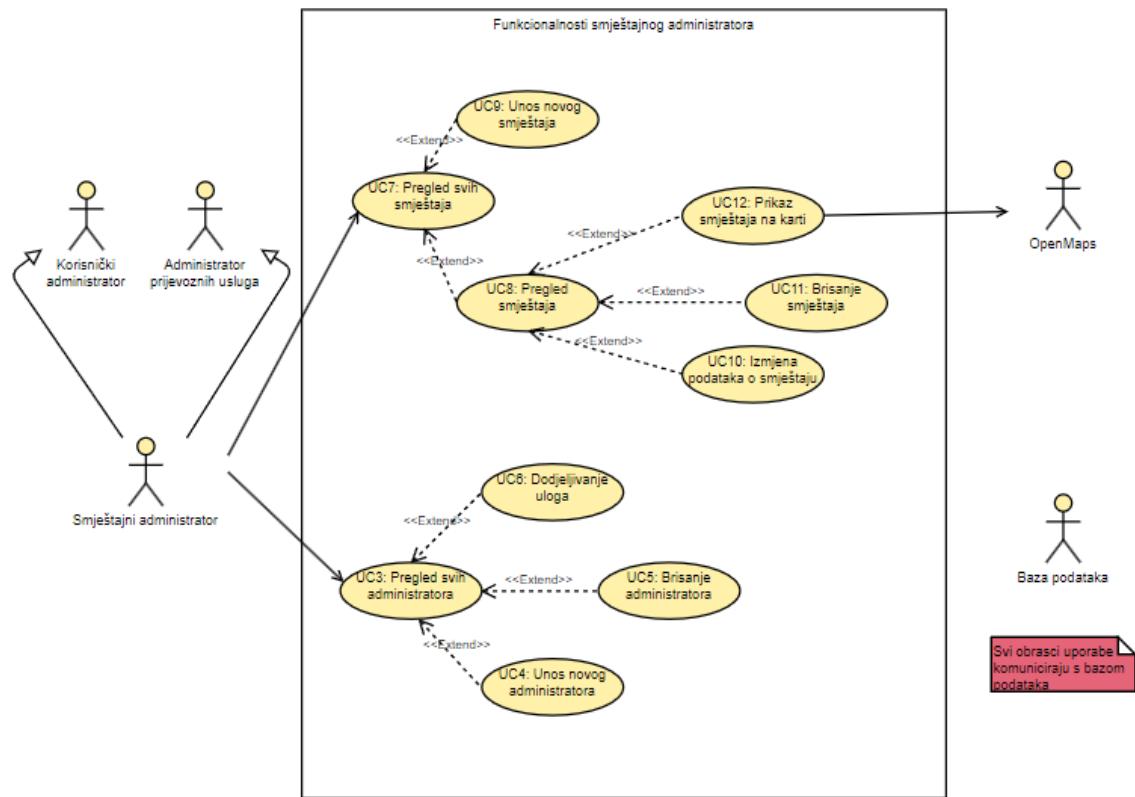
3. Aplikacija unutar predloška za e-mail postavlja podatke o medicinskoj usludi i pojedinostima
 4. Na e-mail adresu prijevoznika se šalje plan
- **Opis mogućih odstupanja:**
 - 3.a Podatci u aplikaciji ne kooperiraju sa podatcima prijevoznika
 1. Sustav upozorava da je došlo do pogrešne korelacije podataka

UC25 - slanje e-maila pacijentu

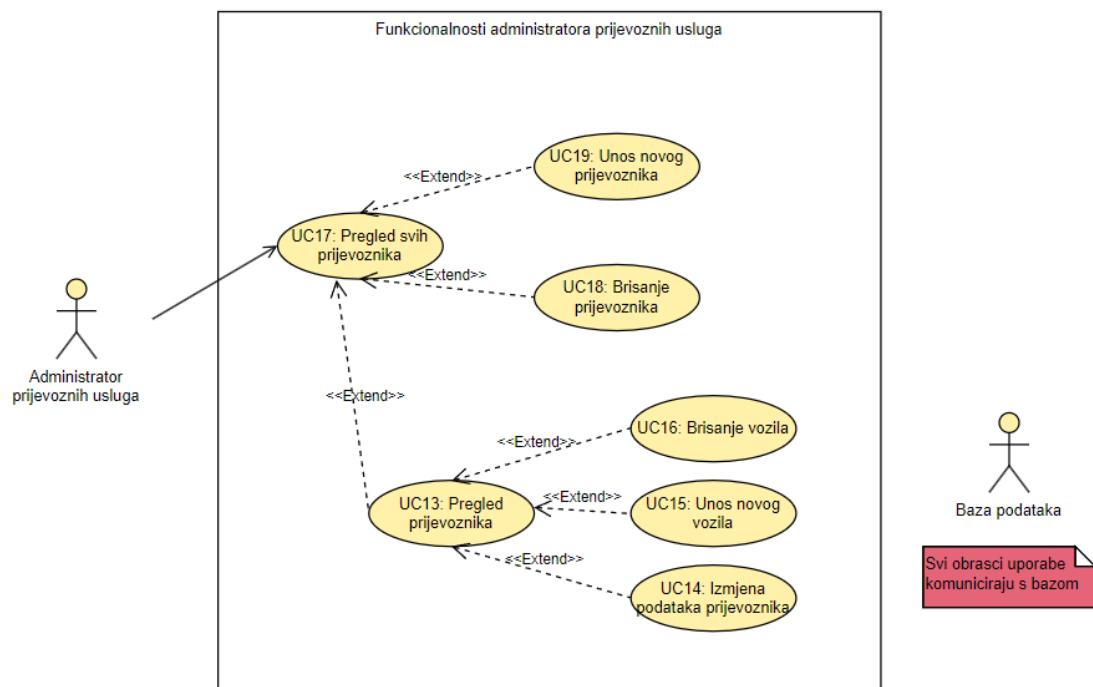
- **Glavni sudionik:** korisnički administrator
- **Cilj:** poslati e-mail s pojedinostima termina pacijentu
- **Sudionici:** baza podataka, aplikacija medicinskih usluga
- **Preduvjet:** administrator je prijavljen u sustav i ima ulogu korisničkog administratora
- **Opis osnovnog tijeka:**
 1. Plan o medicinskim uslugama je zaključan
 2. Dohvaća se e-mail adresa pacijenta
 3. Aplikacija unutar e-maila postavlja pojedinosti o terminu medicinske usluge - termin, prijevoznika, smještaj
 4. Na e-mail adresu pacijenta se šalje plan
- **Opis mogućih odstupanja:**
 - 3.a Podatci u aplikaciji ne kooperiraju sa pacijentovim podatcima
 1. Sustav upozorava da je došlo do pogrešne korelacije podataka

Dijagrami obrazaca uporabe

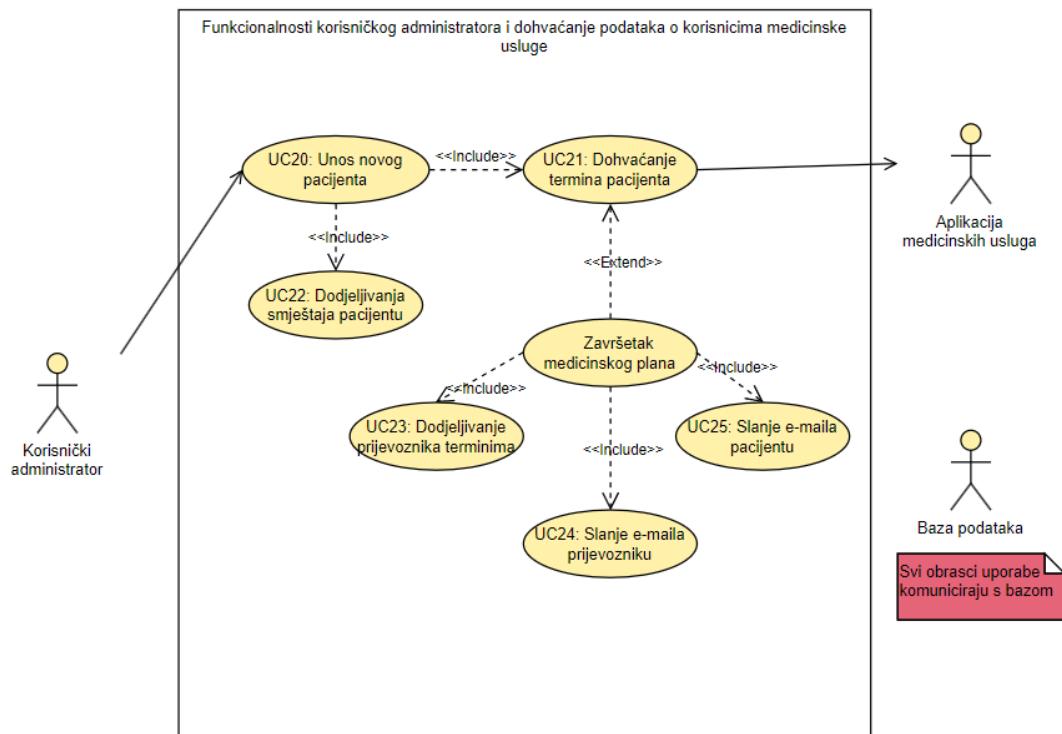
Slika 3.1: Dijagram obrasca uporabe, osnovne funkcionalnosti administratora



Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti smještajnog administratora



Slika 3.3: Dijagram obrasca uporabe, funkcionalnosti administratora prijevoznih usluga

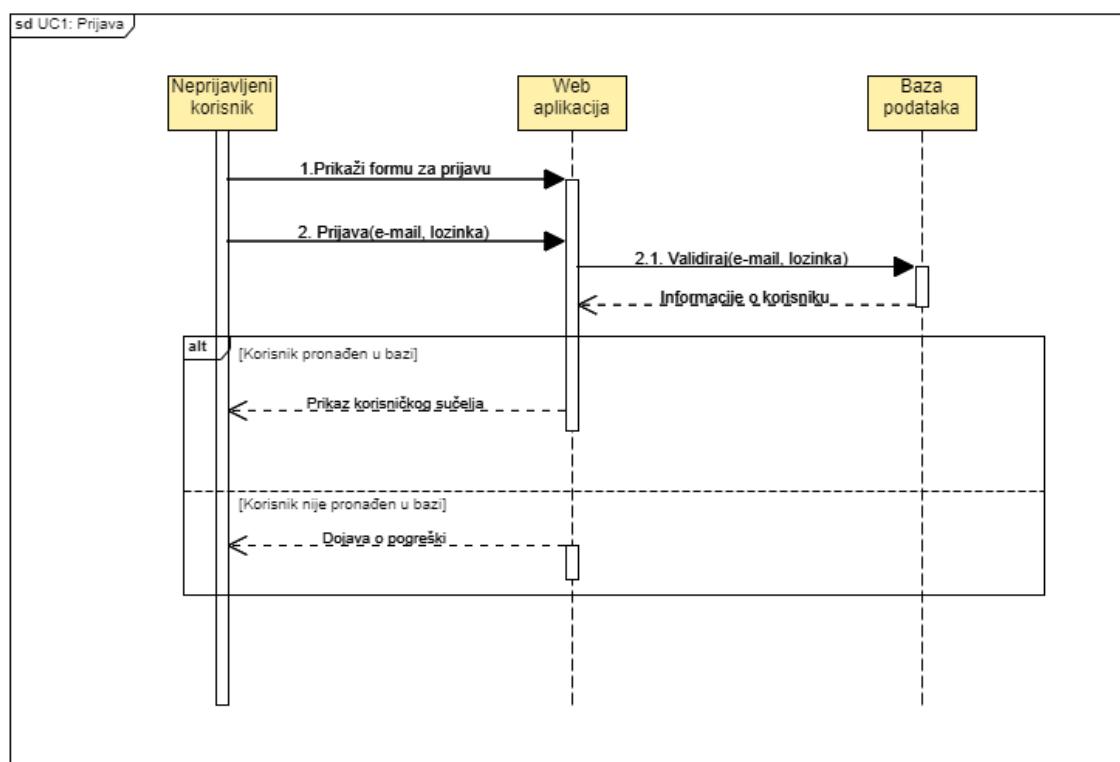


Slika 3.4: Dijagram obrasca uporabe, funkcionalnosti korisničkog administratora

3.1.2 Sekvencijski dijagrami

Prijava – obrazac uporabe UC1

Neregistrirani korisnik odabire gumb “Prijavi se” te ga se preusmjeruje na stranicu s formom za prijavu. Ovdje upisuje svoj e-mail i lozinku te klikom na gumb ”Nastavi” uneseni podaci se šalju na validaciju. Ukoliko se podaci podudaraju s onima pohranjenima u bazi, dostavljaju se informacije o administratoru te token za određene uloge koje administrator ima. Ovisno o tokenu, administratoru postaje dostupno korisničko sučelje prilagođeno njegovoj ulozi ili ulogama. U suprotnom, ako uneseni podaci ne odgovaraju niti jednom administratoru u bazi, korisnik prima obavijest o pogreški kako bi znao da nije registriran u sustavu.



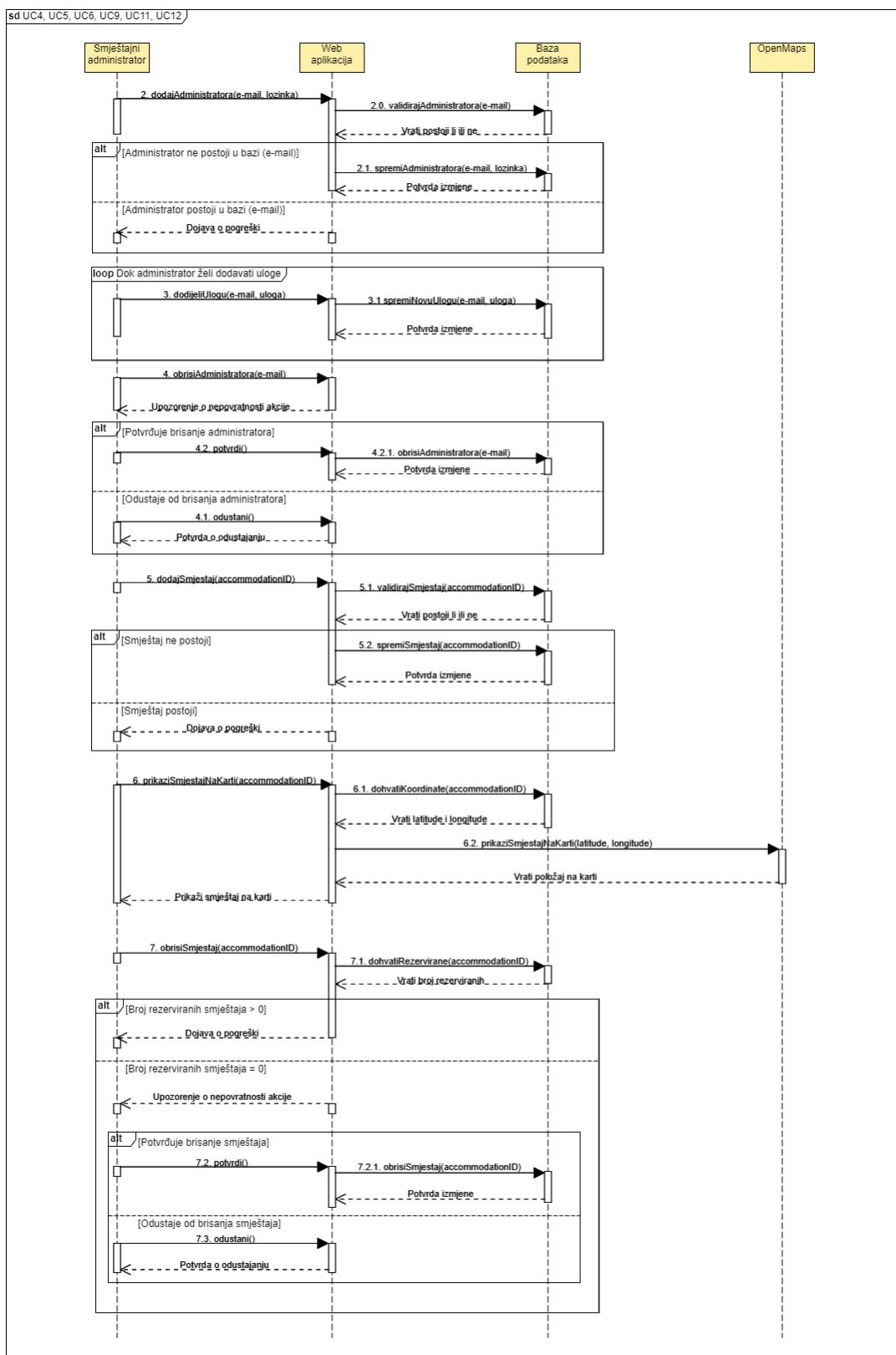
Slika 3.5: Sekvencijski dijagram za UC1

Stvaranje, brisanje, dodjeljivanje uloga administratorima – obrazac uporabe UC4, UC5, UC6

Smještajni administrator ima pristup listi svih administratora u bazi. On je ovlašten unositi nove administratore ukoliko oni već ne postoje u bazi, identificirajući ih prema e-mail adresi. U slučaju da administrator već postoji, sustav generira poruku o pogreški. Nadalje, smještajni administrator ima ovlasti dodjeljivanja uloga drugim administratorima. Također, ima mogućnost uklanjanja drugih administratora, pri čemu mu se šalje upozorenje o nepovratnosti akcije. U tom trenutku, smještajni administrator ima opciju potvrde ili odustajanja. Ukoliko potvrdi brisanje, izmjene se pohranjuju u bazu.

Stvaranje, brisanje i prikaz smještaja na karti – obrasci uporabe UC9, UC11, UC12

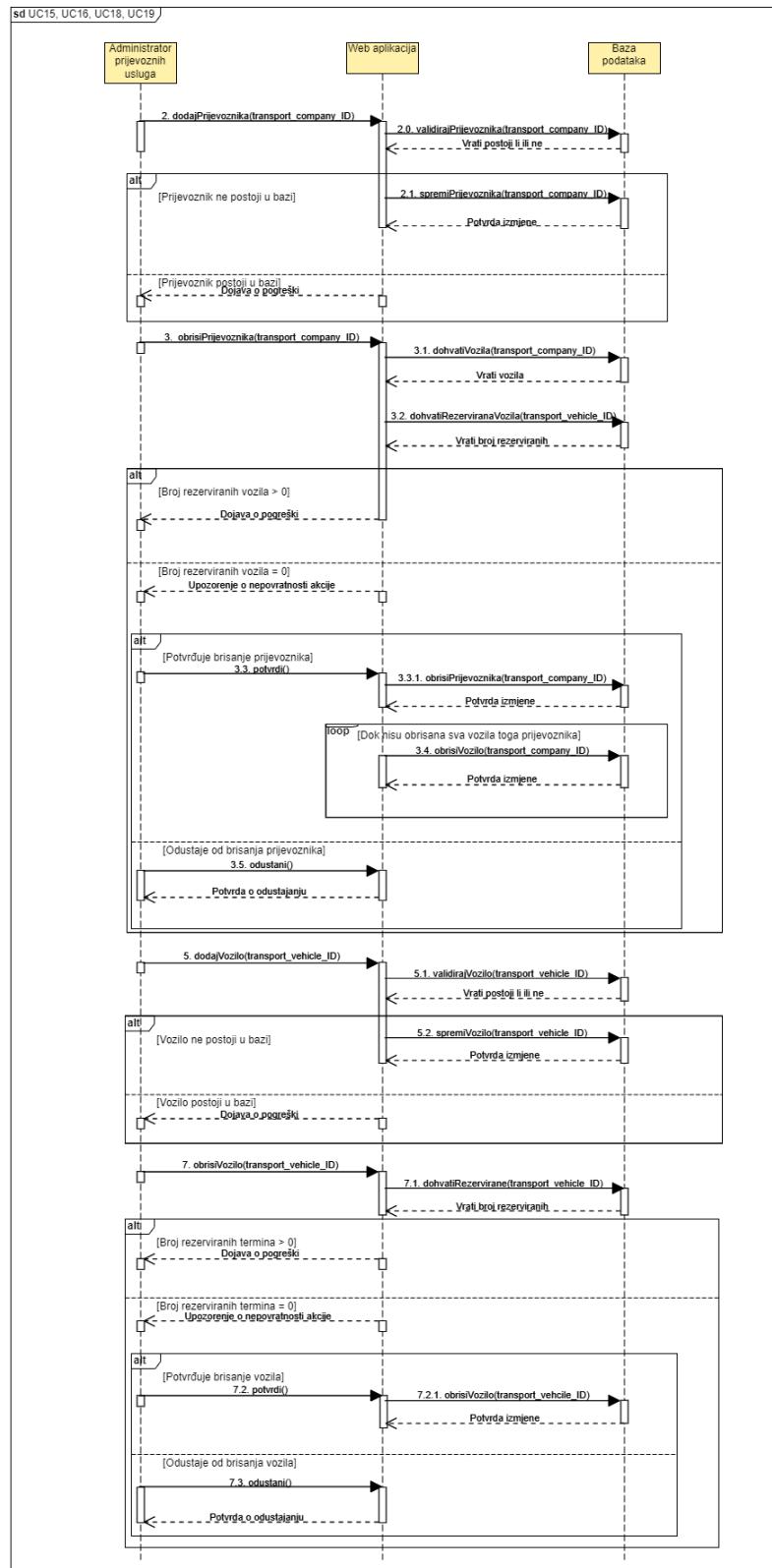
Smještajni administrator ima ovlasti i nad smještajem. Može dodavati nove smještaje, a ukoliko takav smještaj već ne postoji, podaci se pohranjuju u bazu. Nadalje, smještajni administrator ima mogućnost pregledavanja lokacije smještaja na karti putem koordinata (latitude i longitude). U postupku brisanja smještaja, sustav provjerava da li je određeni smještaj rezerviran. Ako nije, administratoru se šalje upozorenje o nepovratnosti akcije te ima opciju potvrde ili odustajanja. Ukoliko potvrди, izmjene se pohranjuju u bazu.



Slika 3.6: Sekvencijski dijagram za UC4, UC5, UC6, UC9, UC11, UC12

**Stvaranje i brisanje prijevoznika, stvaranje i brisanje vozila – obrasci uporabe
UC15, UC16, UC18, UC19**

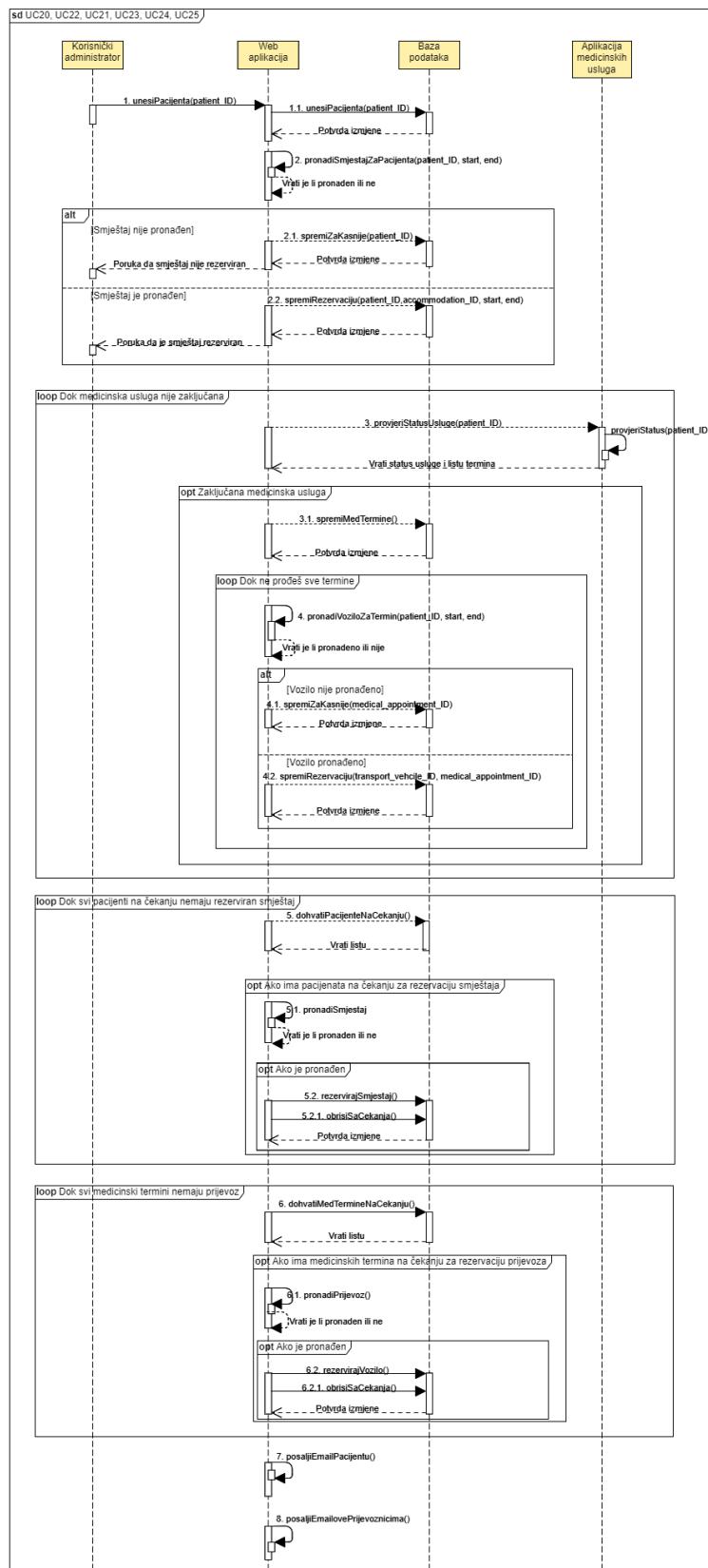
Administrator prijevoznih usluga ima ovlasti dodavanja novih prijevoznika, pod uvjetom da već nisu prisutni u sustavu. Kada želi obrisati prijevoznika, prvo se provjerava jesu li vozila toga prijevoznika rezervirana u nekom terminu. Ukoliko jesu, šalje se poruka pogreške, a ako nisu administratoru se šalje upozorenje o nepovratnosti akcije te ima mogućnost potvrditi svoj izbor. Ako potvrди, briše se prijevoznik i sva vozila toga prijevoznika. Dodavanjem novog vozila provjerava se postoji li već isto vozilo, ako ne postoji vozilo se dodaje u bazu. Brisanjem vozila provjerava se je li vozilo rezervirano u nekom terminu. Ukoliko jest, šalje se poruka o pogreški, a ako nije administratoru se šalje upozorenje o nepovratnosti te ima mogućnost potvrditi svoj izbor. Potvrđivanjem akcije briše se odabранo vozilo.



Slika 3.7: Sekvencijski dijagram za UC15, UC16, UC18, UC19

Unos pacijenta, dodjeljivanje smještaja, dohvatanje termina, dodjeljivanje prijevoza terminima, slanje e-maila – obrasci uporabe UC20, UC21, UC22, UC23, UC24, UC25

Korisnički administrator unosi podatke o pacijentu te se on unosi u bazu. Odmah se traži smještaj za termin pacijenta, ako se pronađe smještaj se rezervira u tome vremenu, a ako ne pacijenta se stavlja na čekanje. Nakon unosa pacijenta, moramo provjeravati je li aplikacija medicinskih usluga zaključala medicinsku uslugu. Kada je medicinska usluga zaključana spremamo medicinske termine te svima moramo pokušati dodijeliti prijevoznika, to jest, vozilo u vremenu termina usluge. Ako je vozilo pronađeno, rezervira se, a ako nije termin se stavlja na čekanje. Kada se prođu svi termini, moramo dodijeliti smještaj pacijentu ako mu još nije dodijeljen te nakon toga za termine koji nemaju rezerviran prijevoz pronaći i rezervirati vozilo. Nakon rezerviranih prijevoza i smještaja pacijentu se šalje mail o svim rezervacijama, a prijevoznicima kontaktni podaci korisnika te podaci o rezervaciji smještaja.



Slika 3.8: Sekvencijski dijagram za UC20, UC21, UC22, UC23, UC24, UC25

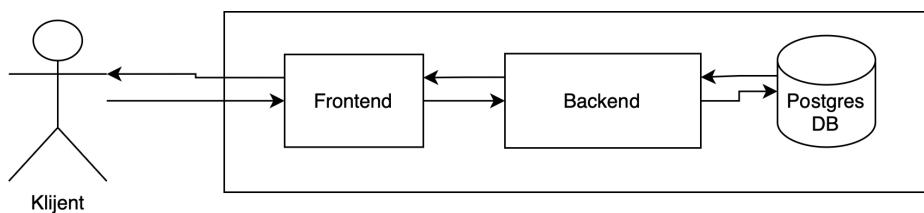
3.2 Ostali zahtjevi

- Aplikacija mora osigurati visoku razinu pouzdanosti u evidenciji i koordinaciji smještaja i prijevoza, uz minimalnu grešku u podacima.
- Sučelje treba biti intuitivno i prilagođeno korisnicima koji možda nisu upoznati s aplikacijama sličnog sadržaja.
- Vrijeme odziva aplikacije na korisničke zahtjeve treba biti brzo kako bi se osiguralo učinkovito upravljanje rezervacijama.
- Aplikacija treba podržavati veliki broj korisnika i smještajnih kapaciteta, uz mogućnost lakoće nadogradnje za dodatne kapacitete.
- Sigurnosni protokoli za zaštitu osobnih podataka korisnika i osjetljivih informacija o rezervacijama su neophodni.
- Aplikacija treba biti kompatibilna s postojećim sustavima za administraciju zdravstvenih usluga.
- Aplikacija treba osigurati efikasno upravljanje uslugama kako bi se zadovoljile potrebe i očekivanja klijenata.
- Aplikacija mora omogućiti efikasno praćenje i upravljanje rezervacijama smještaja i prijevoza, s detaljnim izvještajima o statusu i dostupnosti.
- Aplikacija treba biti dizajnirana s fokusom na efikasnost i jednostavnost navigacije, omogućujući brzu i laku upotrebu bez nepotrebnih vizualnih distrakcija.

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri glavna dijela:

- *REST API poslužitelj (Spring Boot)*
- *Klijentska aplikacija (React)*
- *Baza podataka*



Slika 4.1: Arhitektura sustava

Korisnik interagira s klijentskom aplikacijom putem web preglednika. Ova aplikacija je odvojena od backend sustava te koristi React uz typescript za stvaranje dinamičkog korisničkog sučelja.

REST API poslužitelj izgrađen je koristeći Spring Boot, koji omogućuje izradu visoko efikasnih i skalabilnih RESTful servisa. Komunikacija između klijenta i servera odvija se putem HTTP protokola, gdje frontend šalje zahtjeve backendu, a backend odgovara s potrebnim podacima u JSON formatu.

Klijentska aplikacija je odgovorna za prikazivanje podataka korisniku i obradu korisničkih interakcija. Ona komunicira s backendom putem REST API-ja za dohvaćanje i slanje podataka.

Postgres baza podataka čuva sve podatke potrebne za aplikaciju. Spring Boot aplikacija komunicira s bazom podataka koristeći Spring Data JPA za upravljanje podacima.

Odlučili smo se za korištenje Intellij-a kao razvojnog okruženja za backend, a za frontend koristimo Visual Studio Code. Za razliku od tradicionalnog MVC koncepta, naš sustav je podijeljen na odvojene slojeve gdje backend (REST API poslužitelj) i frontend (klijentska aplikacija) rade neovisno jedan o drugome. Uspoređujući

s tradicionalnim MVC konceptom, mogli bismo reći da je View (V u MVC) na frontendu, a Model I Controller (M I C u MVC) na backendu. Ovo omogućuje fleksibilnost i lakše skaliranje svakog dijela aplikacije zasebno.

4.1 Baza podataka

Za potrebe naše aplikacije korisiti ćemo relacijsku bazu podataka kako bismo lakše oblikovali stvarni svijet. Baza nam je potrbna za metodičku pohranu podataka te njihovo brzo dohvaćanje. Naša baza podataka se sastoji od sljedećih entiteta:

- Patient
- Accommodation
- AccommodationOrder
- AccommodationBooking
- TransportCompany
- TransportVehicle
- TransportBooking
- MedicalAppointment
- AdminRole
- AdminRoles
- Admin

4.1.1 Opis tablica

Patient Ovaj entitet sadrži sve informacije o korisniku usluga našeg sustava. Kako su svi korisnici pacijenti odlučili smo ih imenovati tako. Entitet sadrži atribute: patient_ID, first_name, last_name, PIN, email i phone_number. Ovaj entitet je u vezi *One-to-Many* s entitetom AccommodationOrder preko atributa patient_ID.

Patient		
patient_ID	VARCHAR	primarni ključ tablice
first_name	VARCHAR	ime pacijenta
last_name	VARCHAR	prezime pacijenta

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Patient		
PIN	VARCHAR	Personal Identification Number, kao OIB u hrvatskoj
email	VARCHAR	pacijentov email
phone_number	VARCHAR	pacijentov telefonski broj

Accommodation Ovaj entitet sadržava potrebne podatke o nekom smještaju koji je smještajni administrator unio. Entitet sadrži atribute: accommodation_ID, type, address, availability_start, availability_end i location. Ovaj entitet je u *One-to-Many* vezi s entitetom AccommodationBooking preko atributa accommodation_ID.

Accommodation		
accommodation_ID	VARCHAR	primarni ključ tablice
type	VARCHAR	vrsta smještaja
address	VARCHAR	adresa smještaja
availability_start	DATETIME	datum i vrijeme od kada je smještaj dostupan
availability_end	DATETIME	datum i vrijeme do kada je smještaj dosupan
location	POINT	kordinate smještaja

AccommodationOrder Ovaj entitet se koristi za pohravnjivanje pacijentovih zah-tjeva o traženom smještaju. U slučaju da traženi smještaj nije odmah dostupan zahtjev se sprema kako bih se kasnije mogao opet pogledati. Entitet sadrži atri-bute: accommodation_order_ID, arrival_datetime, departure_datetime, accommo-dation_type i patient_ID. Ovaj entitet je u *Many-to-One* vezi s entitetom Patient preko atributa patient_id i u vezi *One-to-One* s entitetom AccommodationBooking preko atributa accommodation_booking_id.

AccommodationOrder		
accommodation_order_ID	VARCHAR	primarni ključ tablice
arrival_datetime	DATETIME	vrijeme dolaska pacijenta u državu, od tada mu treba smještaj
departure_datetime	DATETIME	vrijeme odlaska pacijenta iz države, do tada treba smještaj
accommodation_type	VARCHAR	željeni tip smještaja koji pacijent traži
location	VARCHAR	kordinate područja na kojem je potrebno naći smještaj
patient_ID	VARCHAR	ID pacijenta koji je napravio ovaj zahtjev
accommodation_booking_id	VARCHAR	ID bookinga koji je napravljen na temelju zahtjeva

AccommodationBooking Ovaj vezni entitet se koristi za pohranjivanje informacije koji pacijent je kada u kojem smještaju. Entitet sadrži atribute: accommodation_booking_ID i accommodation_ID, accommodation_order_ID. Ovaj entitet je u *Many-to-One* vezi s entitetom Accommodation preko atributa accommodation_ID. U *One-to-One* vezi s entitetom AccommodationOrder preko atributa accommodation_order_ID.

AccommodationBooking		
accommodation_booking_ID	VARCHAR	primarni ključ tablice
accommodation_ID	VARCHAR	smještaj u koji je pacijent smješten
accommodation_order_ID	VARCHAR	ID zahtjeva na temelju kojeg je napravljen booking

TransportCompany Ovaj entitet sadržava informacije o transportnoj firmi. Entitet sadrži atribute: transport_company_ID, name, phone_number, email. Ovaj entitet je u *One-to-Many* vezi s entitetom TransportVehicle preko atributa trans-

port_company_ID.

TransportCompany		
transport_company_ID	VARCHAR	primarni ključ tablice
name	VARCHAR	ime firme
phone_number	VARCHAR	telefonski broj firme
email	VARCHAR	email firme

TransportVehicle Ovaj entitet sadržava informacije o nekom transportnom vozilu. Entitet sadrži atribute: transport_comapny_ID, name, phone_number, email. Ovaj entitet je u *Many-to-One* vezi s entitetom TransportCompany preko atributa transport_company_ID. I u *One-to-Many* vezi s entitetom TransportBooking preko atributa transport_vehicle.ID.

TransportVehicle		
transport_vehicle_ID	VARCHAR	primarni ključ tablice
type	VARCHAR	vrsta vozila
capacity	INT	kapacitet vozila
transport_company_ID	VARCHAR	transportna firma kojoj pripada ovo vozilo

TransportBooking Ovo je slabi vezni entitet koji označava prijevoz pacijenta. Ujedinojuje dva entiteta iz kojih saznajemo koji MedicalAppointment je povezan s vozilom. Entitet sadrži atribute: transport_vehicle_ID i medical_appointment_. Entitet je u *Many-to-One* vezi s entitetom TransportVehicle preko atributa transport_vehicle.ID. I u *One-to-One* vezi s entitetom MedicalAppointment preko atributa medical_appointment_ID.

TransportBooking		
transport_booking_ID	VARCHAR	primarni ključ tablice
transport_vehicle_ID	VARCHAR	vozilo koje obavlja ovaj projekoz

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

TransportBooking		
medical_appointment_ID	VARCHAR	poveznica na medicinski tretman koji nam daje informaciju gdje pacijenta treba voziti

MedicalAppointment Ovaj entitet korisimo za spremanje podataka o medicinskim tretmanima nekog pacijenta. Informacije o ovom ćemo dobiti iz medicinskog sustava, ali ako nije moguće odmah napraviti TransportBooking zbog npr. nedostatka vozila, želimo pospremiti informacije o tretmanu. Entitet sadrži atributе: medical_appointment_ID, patient_PIN, clinic_address, start_datetime, end_datetime. Entitet je u *One-to-One* vezi s entitetom TransportBooking preko atributa medical_appointment.ID. I u *Many-to-One* vezi s entitetom AccommodationOrder.

MedicalAppointment		
medical_appointment_ID	VARCHAR	primarni ključ tablice
patient_PIN	VARCHAR	Personal Identification Number pacijenta za kojeg je ovo medicinski tretman
clinic_address	VARCHAR	adresa klinike u kojoj je tretman
start_datetime	DATETIME	datum i vrijeme početka tretmana
end_datetime	DATETIME	datum i vrijeme kraja tretmana

AdminRole Ovaj entitet označava role admina. Sadrži atributе, admin_role_ID i name. Entitet je u *One-to-Many* vezi s entitetom AdminRoles preko atributa admin_role_ID.

AdminRole		
admin_role_ID	VARCHAR	primarni ključ tablice
name	VARCHAR	deskriptivni naziv role

Admin Ovaj entitet sadrži podatke o adminima. Entitet sadrži atributе: admin_ID,

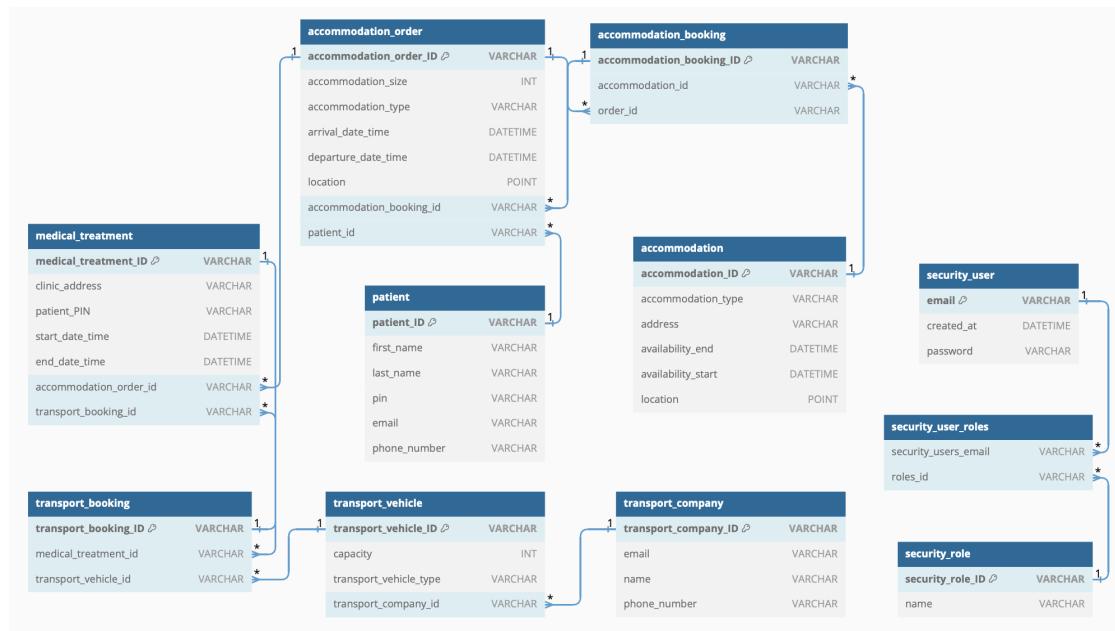
email, first_name, last_name. Entitet je u *One-to-Many* vezi s entitetom AdminRoles preko atributa admin_ID.

Admin		
admin_ID	VARCHAR	primarni ključ tablice
email	VARCHAR	email admina
first_name	VARCHAR	ime admina
last_name	VARCHAR	prezime admina

AdminRoles Ovaj vezni entitet služi kao spoj nekog admina s njegovom roloom. Atributi entiteta su: admin_role_ID i admin_ID. Entitet je u *Many-to-One* vezi s entitetom AdminRole preko atributa admin_role_ID. I entitet je u *Many-to-One* vezi s entitetom Admin preko atributa admin_ID.

AdminRoles		
admin_role_ID	VARCHAR	poveznica na rolu ovog admina
admin_ID	VARCHAR	poveznica na kojeg admina se odnosi ova rola

4.1.2 Dijagram baze podataka

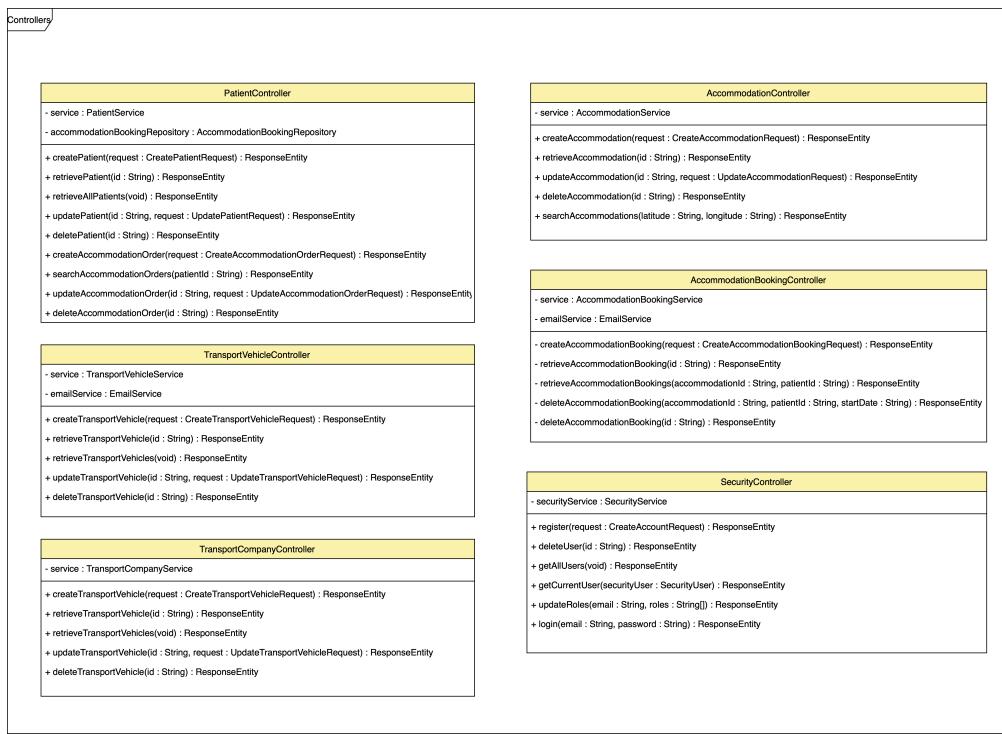


Slika 4.2: ER Dijagram baze podataka

4.2 Dijagram razreda

Na slikama 4.3 do 4.7 prikazani su ključni elementi backend dijela arhitekture DentAll aplikacije.

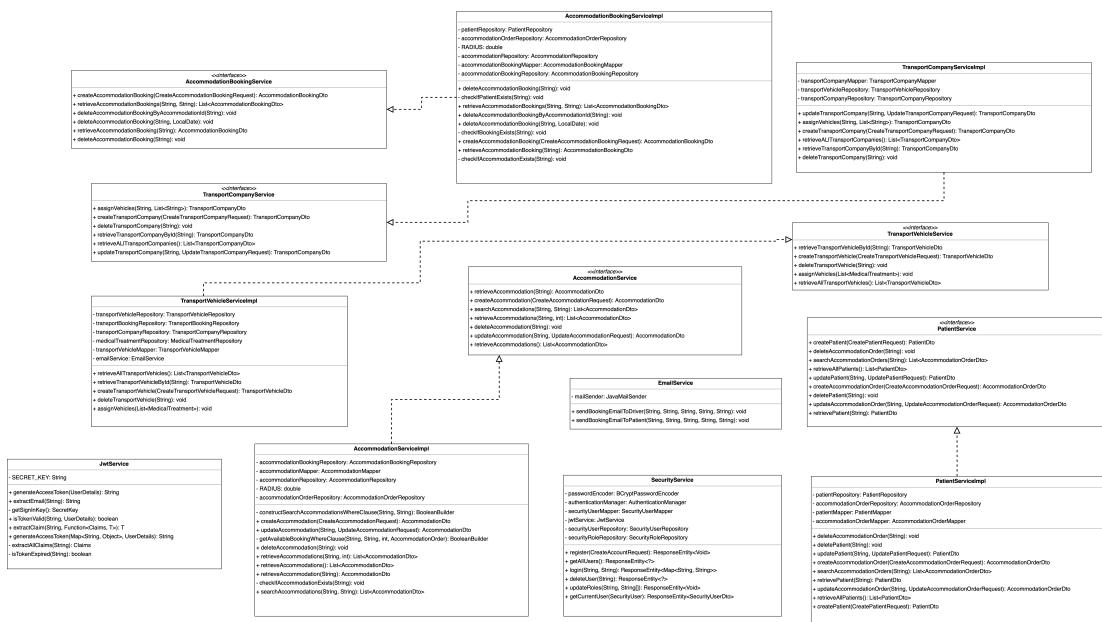
Prva slika 4.3 prikazuje dijagram razreda Controllera, prikazujući sve Controller razrede u aplikaciji. Sljedeći veliki element arhitekture backenda su Service sučelja i pripadajući ServiceImpl razredi prikazani na 4.4. Slika 4.5 fokusira se na dijagram razreda s DTO razredima koje služe za prijenos podataka između Controllera i Modela. Svaka DTO klasa odgovara određenom razredu iz Modela, osiguravajući učinkovit prijenos podataka unutar aplikacije. Na slici 4.6 nalaze se Requests razredi koje predstavljaju zahtjeve od klijenta prema Controller razredima. Ovi razredi sadrže sve potrebne informacije za izvođenje funkcionalnosti poput CREATE i UPDATE. Obuhvaćajući sve domain razrede iz modela DentAll aplikacije, dijagram razreda sa slike 4.7 prikazuje relacije i međusobnu povezanost između razreda unutar modela. Svaki segment dokumentacije dijagrama razreda osmišljen je da naglasi važnost svakog dijela arhitekture u dizajnu i implementaciji sustava.



Slika 4.3: Dijagram razreda - dio Controllers

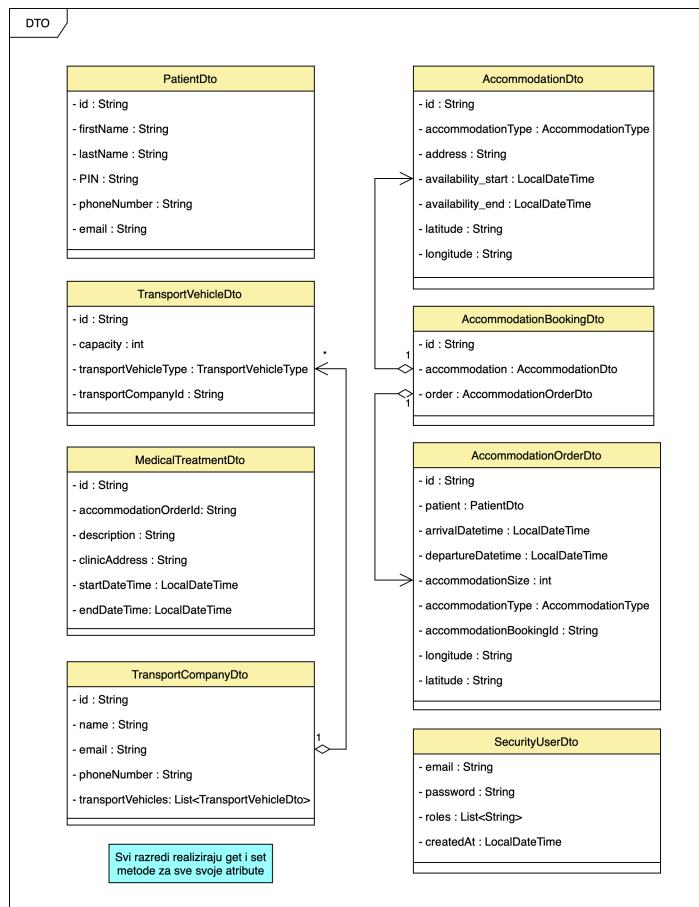
Kontroleri su poveznica između korisnika i aplikacije, odgovorni za upravljanje

okom HTTP zahtjeva i odgovora. Svaki Controller razred u DentAll aplikaciji povezan je s određenim Service sučeljem koji omogućuje obradu poslovne logike. PatientController upravlja pacijentima i sadrži metode poput createPatient i deletePatient, koje prate RESTful princip za CRUD operacije. TransportVehicleController i TransportCompanyController omogućavaju upravljanje vozilima i tvrtkama za transport, s metodama kao što su createTransportVehicle i updateTransportCompany. AccommodationController i AccommodationBookingController služe za upravljanje smještajem te njegovu rezervaciju kroz booking. SecurityController upravlja autentifikacijom i autorizacijom, s metodama poput register i login.



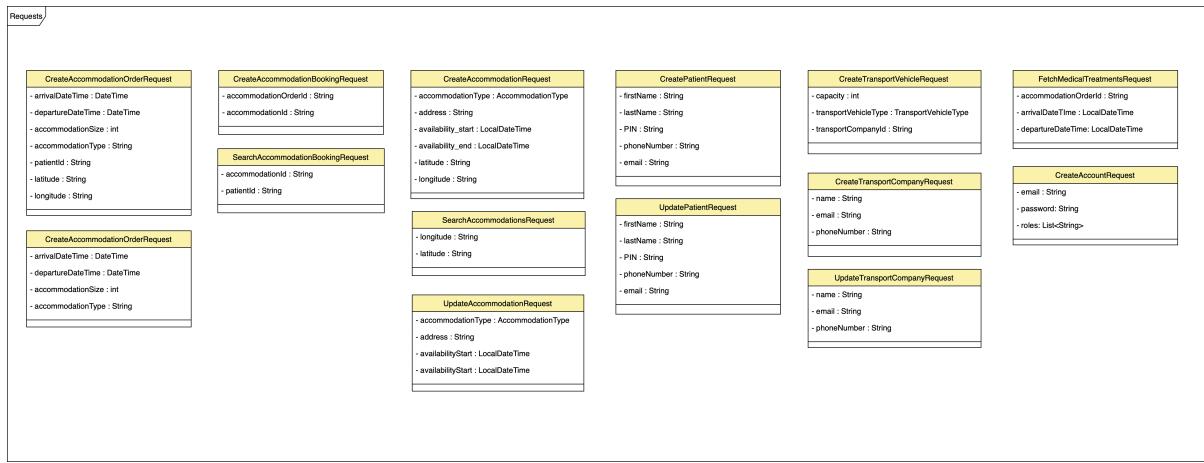
Slika 4.4: Dijagram razreda - dio Service

Servisni sloj u arhitekturi služi kao posrednik između Controllera i domenskih modela, pružajući apstrakciju koja omogućava Controllerima da se usredotoče na obradu HTTP zahtjeva i odgovora, dok Service sloj upravlja poslovnim logikama. Servisni sloj je prikazan na 4.4, gdje su implementacije i sučelja organizirani u skladu s funkcionalnostima koje pružaju. Možemo izdvojiti EmailService i JwtService servise koji pružaju specijalizirane funkcionalnosti – slanje emailova i upravljanje JWT tokenima za autentifikaciju i autorizaciju korisnika. SecurityService sučelje i SecurityServiceImpl su zaduženi za sigurnosne aspekte aplikacije, uključujući autentifikaciju, autorizaciju i upravljanje korisničkim računima.



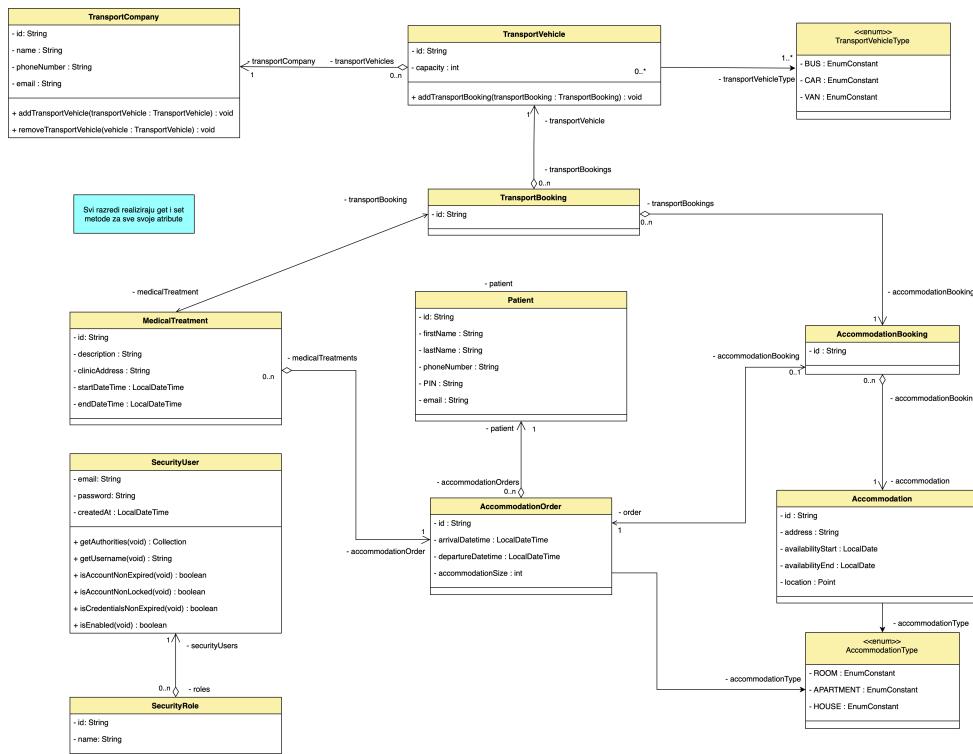
Slika 4.5: Dijagram razreda - dio DTO

DTO razredi su ključni za učinkovit i siguran prijenos podataka između slojeva aplikacije. DTO omogućuje da se modeli ne izlažu direktno klijentima, što smanjuje rizik od manipulacije podataka. Svaki DTO sadrži atribute koji se mapiraju na entitet modela, uz potrebne gettere i setttere za pristup i modifikaciju podataka.



Slika 4.6: Dijagram razreda - dio Requests

Request razredi su specijalizirani razredi koji sadrže sve potrebne informacije za obradu zahtjeva. Oni olakšavaju validaciju ulaznih podataka i smanjuju složenost Controller razreda.



Slika 4.7: Dijagram razreda - dio Model

Domain modeli su osnovni razred nad kojim se obrađuje sve u aplikaciji. Patient, TransportVehicle, Accommodation, i AccommodationOrder su osnovni entiteti.

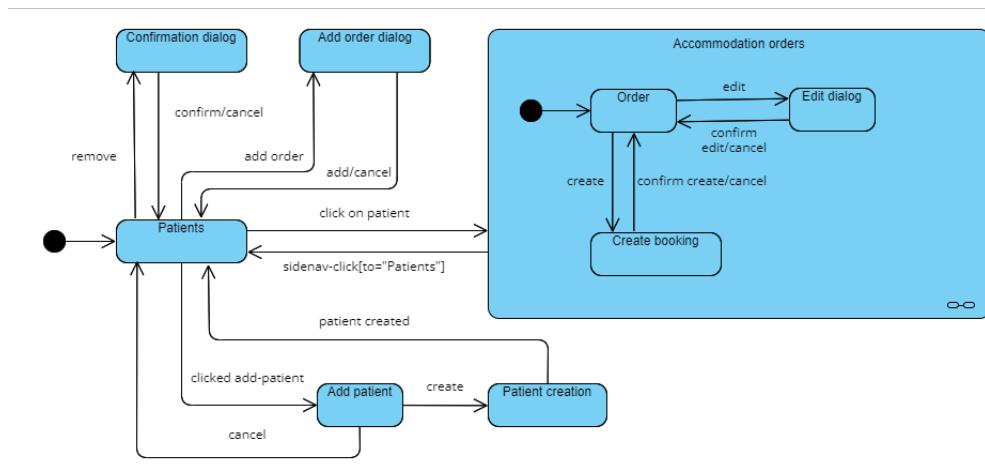
teti koji se međusobno povezuju. Odnosi između entiteta su jasno definirani s odgovarajućim vezama (npr. jedan pacijent može imati više medicinskih tretmana). SecurityUser i SecurityRole modeliraju sigurnosni aspekt aplikacije, omogućavajući upravljanje pristupom i autorizacijom.

4.3 Dijagram stanja

Unutar **Patients dashboard** stanja, sustav nudi nekoliko interakcija:

- Klikom na *Add patient* otvara se modal za unos informacija o pacijentu. Iz ovog modala, moguće su dvije akcije:
 1. Klikom na *Add*, sustav prelazi u stanje izrade pacijenta i nakon toga se vraća na **Patients dashboard**.
 2. Klikom na *Cancel*, sustav se odmah vraća na **Patients dashboard** bez kreiranja novog pacijenta.
- Za brisanje pacijenata, klikom na *Remove* pojavljuje se modal potvrde. Nakon akcije, sustav se uvijek vraća na **Patients dashboard**.
- Dodavanje *accommodation ordera* za pacijente ostvaruje se klikom na *Add accommodation order*, što otvara odgovarajući modal. Iz ovog modala, akcijama *add* ili *cancel* sustav se vraća na **Patients dashboard**.
- Klikom na svakog pacijenta, sustav prelazi u listu narudžbi smještaja, unutar kojeg se može:
 - Kreirati *accommodation booking*.
 - Urediti postojeću narudžbu.

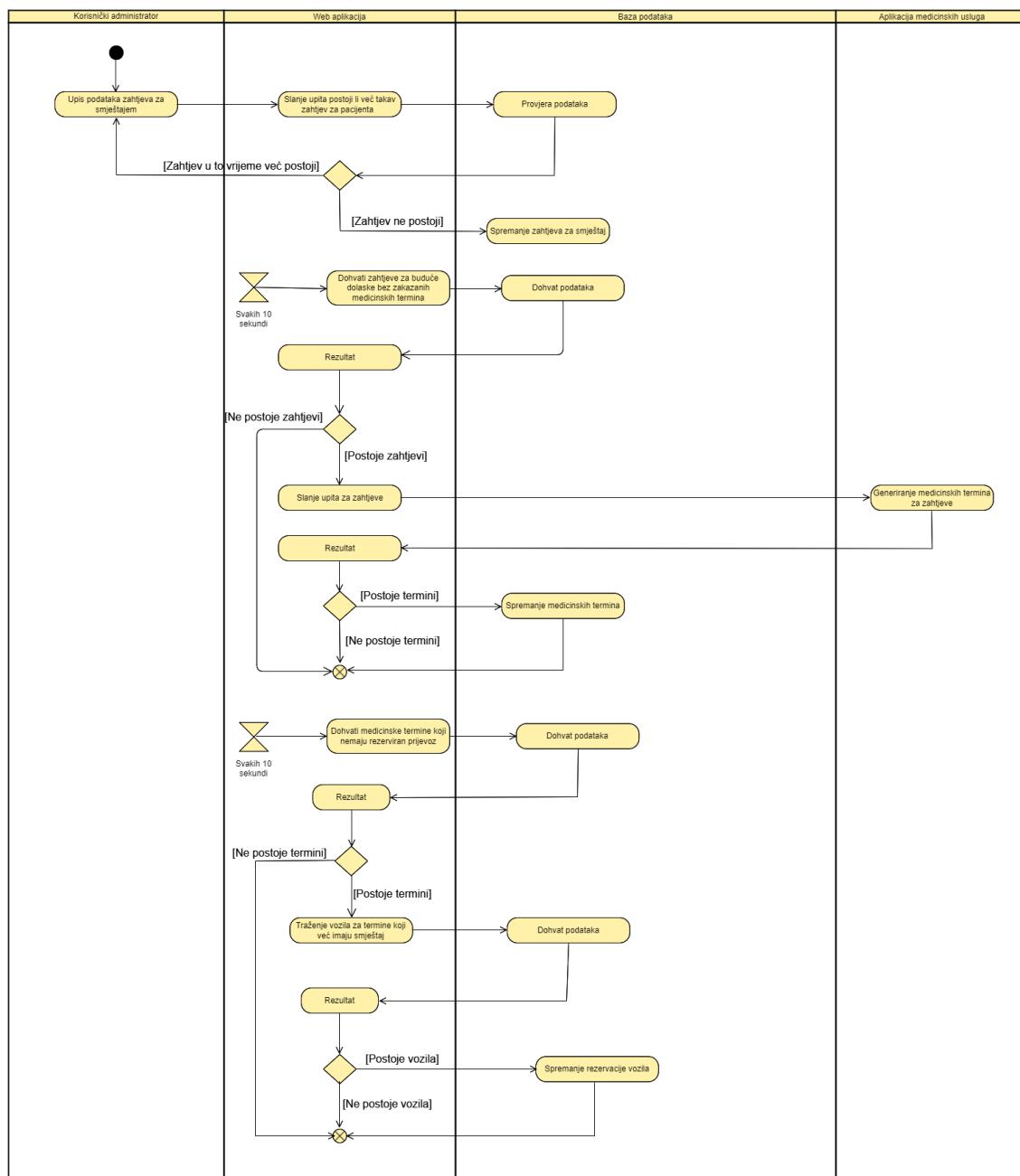
Oba stanja, nakon odrđene akcije, vraćaju na listu narudžbi za navedeni smještaj.



Slika 4.8: Dijagram stanja

4.4 Dijagram aktivnosti

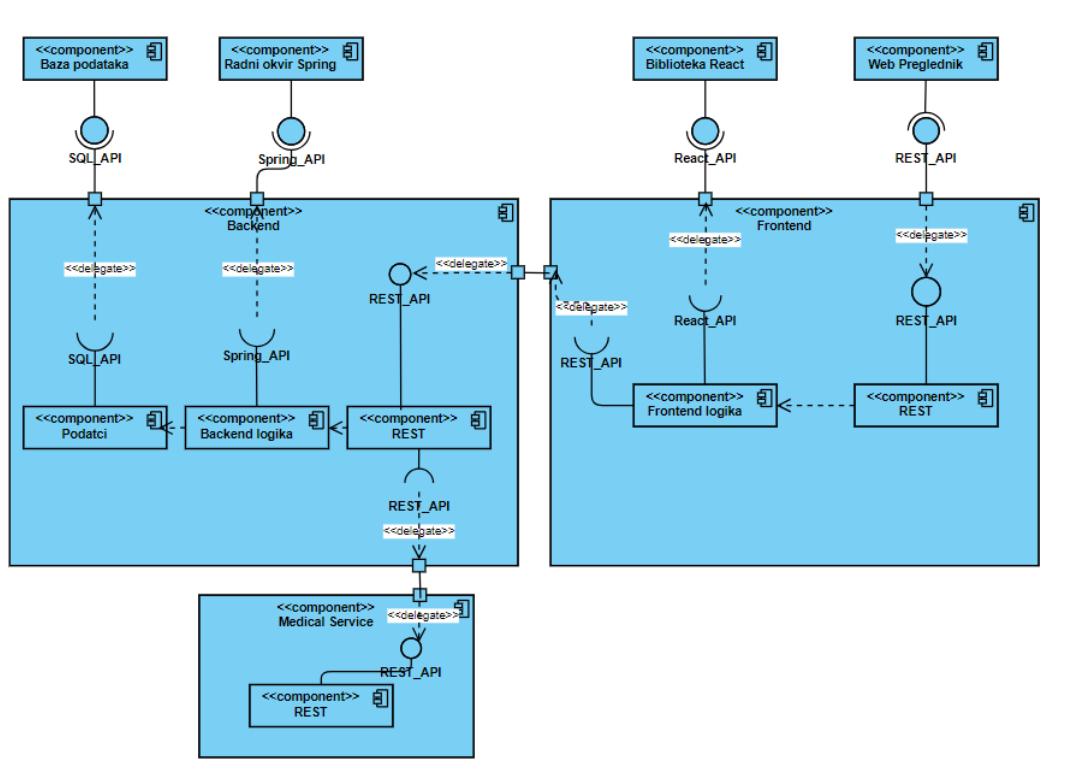
Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Ne upotrebljava se za modeliranje događajima poticanog ponašanja. Na dijagramu aktivnosti 4.9 prikazan je proces kreiranja zahtjeva za smještaj, dohvaćanja medicinskih termina te rezervacije vozila za termine. Korisnički administrator unosi zahtjeve za smještaj u sustav se prijavi u sustav. U sustavu se svakih 10 sekundi provjerava postoji li zahtjev za smještaj (dolazak pacijenta) koji još nema medicinske termine. Za sve koji nemaju medicinske termine zove se Aplikacija za medicinske usluge (external API) koji vraća za sve zahtjeve nasumično 0, 1, 2 ili 3 medicinska termina unutar zadano vremena dolaska i odlaska pacijenta definiranih u zahtjevu. Također, svakih 10 sekundi se provjerava i postoje li i medicinski termini koji još nemaju rezervirano vozilo za prijevoz. Za sve koji nemaju dodjeljuje se vozilo, ako postoji u terminu medicinske usluge, s uvjetom da pacijent već ima rezerviran smještaj.



Slika 4.9: Dijagram aktivnosti

4.5 Dijagram komponenti

Na donjoj slici 4.9 prikazan je dijagram komponenti koji vizualno prikazuje ovisnost komponenti. Aplikacija DentAll sastoji se od tri komponente: frontend, backend i Medical Service. Komponenta frontenda napravljena je korištenjem biblioteke React, a komponenta backenda izgrađena je pomoću Java Spring Boota. Na slici je označeno da korisnik preko web preglednika pristupa aplikaciji, komponente frontend i backend međusobno razmjenjuju podatke preko REST API-ja, baza podataka komunicira sa backendom preko SQL upita, a komponenta Medical Service komunicira sa backendom putem REST API-ja.



Slika 4.10: Dijagram aktivnosti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Kao dogovoren način razmjenjivanja ideja i dogovaranja oko projekta odabранa je aplikacija Whatsapp <https://www.whatsapp.com/>. Github <https://github.com/> je odabran kao sustav za upravljanje cijelokupnim kodom projekta, a za izradu svih dijagrama (sekvencijskih dijagrama, dijagrama razreda, dijagrama komponenti i sl.) korišten je alat Visual Paradigm <https://online.visual-paradigm.com/>.

Za razvojnu okolinu odabran je Visual Studio Code <https://code.visualstudio.com/> iz razloga što je jednostavan za korištenje jer pruža mogućnost pametnog dovršavanja na temelju vrsta varijabli i definicija funkcija za koju je zaslužan IntelliSense. Uz to sve, VSC je prilagodljiva okolina s puno ekstenzija, a i lako se povezuje s GitHubom.

Aplikacija je razvijena pomoću Java Spring Boot <https://spring.io/projects/spring-boot/>, a korišten programski jezik je Java <https://www.java.com/en/>, verzija 17. Za frontend je korišten programski jezik JavaScript <https://www.javascript.com/>, preciznije, jezik TypeScript <https://www.typescriptlang.org/> koji je nadogradnja već postojećeg JavaScripta te biblioteka React <https://react.dev/>. Korištenjem TypeScripta omogućeno je dodavanje tipova i dodatnih mogućnosti nego samo koristeći JavaScript. Upotrebljena je i biblioteka ReactQuery koja omogućuje automatsko ažuriranje podataka - implementira logiku za osvježavanje i predmemoriranje podataka, kao i za njihovu invalidaciju. Uz sve navedeno, korištene su i biblioteke UI komponenti ChakraUI i TailwindCSS za stiliziranje i dizajn korisničkog sučelja.

Za bazu podataka korišten je Postgres <https://www.postgresql.org/> koja se pokreće pomoću Dockera <https://www.docker.com/>, odnosno Docker containera.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti ostvareno je korištenjem JUnit, Mockito i RestAssured alata za ispitivanje. U nastavku su opisani provedeni testovi, te su priloženi kodovi.

Testovi testiraju AccommodationController kao i druge komponente o kojima AccommodationController ovisi.

Inicijalizacija izgleda ovako:

```

1  @BeforeEach
2  void setUp() {
3      RestAssured.port = port;
4      Mockito.when(authenticationManager.authenticate(Mockito.any()))
5          .thenReturn(new TestingAuthenticationToken("admin", "password", "ACCOMMODATION",
6              "TRANSPORT", "PATIENT"));
7  }

```

U ovom ispitnom slučaju ispitan je pokušaj kreacije novog smještaja, a očekivani rezultat je u isti taj smještaj u JSON obliku te HTTP status kod 200.

```

1  @Test
2  @WithMockUser(username = "admin", roles = {"ACCOMMODATION", "TRANSPORT", "PATIENT"})
3  void testCreateAccommodation() {
4      CreateAccommodationRequest request = new CreateAccommodationRequest();
5      request.setAddress(UUID.randomUUID().toString());
6      request.setLatitude("45");
7      request.setLongitude("15");
8      request.setAccommodationType("ROOM");
9      request.setAddress(UUID.randomUUID().toString());
10     request.setAvailabilityEnd(LocalDate.of(2021, 12, 31));
11     request.setAvailabilityEnd(LocalDate.of(2022, 12, 31));
12
13     Response response = given()
14         .contentType("application/json")
15         .body(request)
16         .when()
17         .post("/accommodations")
18         .then()
19         .statusCode(200)
20         .extract()
21         .response();
22
23     AccommodationDto dtoResponse = response.as(AccommodationDto.class);
24     assertEquals(request.getAccommodationType(), dtoResponse.getAccommodationType()
25                 .toString());
26     assertEquals(request.getAddress(), dtoResponse.getAddress());
27     assertEquals(request.getLatitude() + ".0", dtoResponse.getLatitude());
28     assertEquals(request.getLongitude() + ".0", dtoResponse.getLongitude());

```

```

28     assertEquals(request.getAvailabilityStart(), dtoResponse.getAvailabilityStart());
29     ;
30     assertEquals(request.getAvailabilityEnd(), dtoResponse.getAvailabilityEnd());
31
32     ArgumentCaptor<Accommodation> captor = ArgumentCaptor.forClass(Accommodation.
33         class);
34     Mockito.verify(accommodationRepository).save(captor.capture());
35     Accommodation accommodation = captor.getValue();
36
37     assertEquals(request.getAccommodationType(), accommodation.getAccommodationType
38         ());
39     assertEquals(request.getAddress(), accommodation.getAddress());
40     assertEquals(request.getLatitude() + ".0", String.valueOf(accommodation.
41         getLocation().getY()));
42     assertEquals(request.getLongitude() + ".0", String.valueOf(accommodation.
43         getLocation().getX()));
44     assertEquals(request.getAvailabilityStart(), accommodation.getAvailabilityStart
45         ());
46     assertEquals(request.getAvailabilityEnd(), accommodation.getAvailabilityEnd());
47 }
```

U drugom ispitnom slučaju testiran je pokušaj brisanja smještaja, kada taj smještaj postoji.

```

1  @Test
2  @WithMockUser(username = "admin", roles = {"ACCOMMODATION", "TRANSPORT", "PATIENT"})
3  void testDeleteAccommodation_whenExists() {
4      String id = UUID.randomUUID().toString();
5      Mockito.when(accommodationRepository.existsById(id)).thenReturn(true);
6
7      Response response = given()
8          .contentType("application/json")
9          .when()
10         .delete("/accommodations/" + id)
11         .then()
12         .statusCode(200)
13         .extract()
14         .response();
15
16     assertEquals("Successfully deleted", response.asString());
17
18     Mockito.verify(accommodationRepository).deleteById(id);
19 }
```

U trećem ispitnom slučaju testiran je pokušaj brisanja smještaja kada smještaj NE postoji. Očekuje se iznimka.

```

1  @Test
2  @WithMockUser(username = "admin", roles = {"ACCOMMODATION", "TRANSPORT", "PATIENT"})
3  void testDeleteAccommodation_whenDoesNotExist_shouldThrow() {
4      String id = UUID.randomUUID().toString();
5      Mockito.when(accommodationRepository.existsById(id)).thenReturn(false);
```

```
6      Response response = given()
7      .contentType("application/json")
8      .when()
9      .delete("/accommodations/" + id)
10     .then()
11     .statusCode(400)
12     .extract()
13     .response();
14
15
16     ExceptionResponse exceptionResponse = response.as(ExceptionResponse.class);
17
18     assertEquals("Accommodation with id: '" + id + "' not found!", exceptionResponse
19     .getMessage());
}
```

U četvrtom ispitnom slučaju testiran je pokušaj ažuriranja smještaja kada on postoji.

```
1 @Test
2 @WithMockUser(username = "admin", roles = {"ACCOMMODATION", "TRANSPORT", "PATIENT"})
3 void testUpdateAccommodation_whenDoesExist() {
4     String id = UUID.randomUUID().toString();
5     UpdateAccommodationRequest request = UpdateAccommodationRequest.builder()
6         .accommodationType(AccommodationType.APARTMENT)
7         .address(UUID.randomUUID().toString())
8         .availabilityStart(LocalDate.of(2021, 12, 31))
9         .availabilityEnd(LocalDate.of(2022, 12, 31))
10        .build();
11
12     Mockito.when(accommodationRepository.findById(id)).thenReturn(Optional.of(
13         Accommodation.builder()
14             .id(id)
15             .accommodationType(AccommodationType.ROOM)
16             .address(UUID.randomUUID().toString())
17             .availabilityStart(LocalDate.of(2021, 1, 1))
18             .availabilityEnd(LocalDate.of(2021, 1, 31))
19             .location(toPoint("15", "45"))
20             .build()));
21
22     Response response = given()
23         .contentType("application/json")
24         .body(request)
25         .when()
26         .put("/accommodations/" + id)
27         .then()
28         .statusCode(200)
29         .extract()
30         .response();
31
32     AccommodationDto dtoResponse = response.as(AccommodationDto.class);
33
34     assertEquals(request.getAccommodationType().toString(), dtoResponse.
```

```

34     assertEquals(request.getAccommodationType().toString());
35     assertEquals(request.getAddress(), dtoResponse.getAddress());
36     assertEquals(request.getAvailabilityStart(), dtoResponse.getAvailabilityStart());
37     ;
38     assertEquals(request.getAvailabilityEnd(), dtoResponse.getAvailabilityEnd());
39
40     Mockito.verify(accommodationRepository).findById(id);
41
42     ArgumentCaptor<Accommodation> captor = ArgumentCaptor.forClass(Accommodation.
43         class);
44
45     Mockito.verify(accommodationRepository).save(captor.capture());
46     Accommodation accommodation = captor.getValue();
47
48     assertEquals(request.getAccommodationType().toString(), accommodation.
49         getAccommodationType().toString());
50     assertEquals(request.getAddress(), accommodation.getAddress());
51     assertEquals(request.getAvailabilityStart(), accommodation.getAvailabilityStart
52         ());
53     assertEquals(request.getAvailabilityEnd(), accommodation.getAvailabilityEnd());
54 }
55
56     private static Point toPoint(String longitude, String latitude) {
57         GeometryFactory geometryFactory = new GeometryFactory();
58         return geometryFactory.createPoint(new Coordinate(Double.parseDouble(longitude),
59             Double.parseDouble(latitude)));
60     }

```

U petom ispitnom slučaju testiran je pokušaj ažuriranja smještaja kada on NE postoji.

```

1     @Test
2     @WithMockUser(username = "admin", roles = {"ACCOMMODATION", "TRANSPORT", "PATIENT"})
3     void testUpdateAccommodation_whenDoesNotExist_shouldThrow() {
4         String id = UUID.randomUUID().toString();
5         UpdateAccommodationRequest request = UpdateAccommodationRequest.builder()
6             .accommodationType(AccommodationType.APARTMENT)
7             .address(UUID.randomUUID().toString())
8             .availabilityStart(LocalDate.of(2021, 12, 31))
9             .availabilityEnd(LocalDate.of(2022, 12, 31))
10            .build();
11
12         Mockito.when(accommodationRepository.existsById(id)).thenReturn(false);
13
14         Response response = given()
15             .contentType("application/json")
16             .body(request)
17             .when()
18             .put("/accommodations/" + id)
19             .then()
20             .statusCode(400)
21             .extract()
22             .response();

```

```
23  
24     ExceptionResponse exceptionResponse = response.as(ExceptionResponse.class);  
25  
26     assertEquals("Accommodation with id: '" + id + "' not found!", exceptionResponse  
27         .getMessage());  
    }
```

5.2.2 Ispitivanje sustava

Ispitivanje sustava izvedeno je u programskom jeziku Python koristeći Selenium WebDriver. Priloženi kodovi su u nastavku, ali prije toga je izdvojen zajednički kod koji koriste svi testovi.

```
def login(driver, email, password):  
    driver.get(f"{FRONTEND_URL}/login")  
  
    email_input = driver.find_element(by=By.NAME, value="email")  
    password_input = driver.find_element(by=By.NAME, value="password")  
    submit_button = driver.find_element(by=By.NAME, value="submit")  
  
    email_input.send_keys(email)  
    password_input.send_keys(password)  
    submit_button.click()  
  
    sleep(1)
```

Slika 5.1: Zajednički kod testova

Prvi test ispituje pokušaj prijave s krivim podatcima. Očekivani je da se putanja ne promijeni te da se pojavi obavijest u obliku *Toast-a* koja daje više informacija o greški.

```
def test1() -> bool:  
    """  
    Test with wrong credentials.  
    The page should stay the same and a toast should appear.  
    :return:  
    """  
  
    driver = webdriver.Firefox()  
    driver.get(f"{FRONTEND_URL}/login")  
  
    login(driver, email="admin@admin.com", password="paradajzz27")  
  
    testPassed = driver.current_url == f"{FRONTEND_URL}/login" and driver.find_element(by=By.CLASS_NAME,  
                                         value="chakra-alert") is not None  
  
    driver.close()  
    return testPassed
```

Slika 5.2: Izvorni kod 1. testa

Drugi test ispituje pokušaj prijave s ispravnim podatcima. Očekivano ponašanje je da se putanja promijeni te da gumb "Logout" bude vidljiv.

```
def test2() -> bool:  
    """  
    Test with correct credentials.  
    The page should change and a login button should be visible.  
    :return:  
    """  
  
    driver = webdriver.Firefox()  
  
    login(driver, email="admin@admin.com", password="admin")  
    sleep(1)  
  
    testPassed = driver.current_url != f"{FRONTEND_URL}/login" and driver.find_element(by=By.LINK_TEXT,  
                                         value="Logout") is not None  
  
    driver.close()  
    return testPassed
```

Slika 5.3: Izvorni kod 2. testa

Treći test ispituje stvaranje transportne kompanije. Očekivano ponašanje je povećanje broja redaka u tablici koja prikazuje transportne kompanije za 1.

```
def test3() -> bool:
    """
    Test create a new transport company.
    A new transport company should be created.
    :return:
    """

    driver = webdriver.Firefox()

    login(driver, email: "admin@admin.com", password: "admin")
    driver.get(f"{FRONTEND_URL}/transport-companies")
    sleep(3)

    num_of_companies = len(driver.find_elements(by=By.TAG_NAME, value="tr"))

    driver.find_element(by=By.NAME, value="add-transport-company-button").click()

    inputs = driver.find_elements(by=By.TAG_NAME, value="input")
    inputs[0].send_keys("Selenium company")
    inputs[1].send_keys("selenium@gmail.com")
    inputs[2].send_keys("42069420")
    driver.find_element(by=By.NAME, value="submit").click()

    testPassed = len(driver.find_elements(by=By.TAG_NAME, value="tr")) == num_of_companies + 1
    driver.close()
    return testPassed
```

Slika 5.4: Izvorni kod 3. testa

Četvrти test ispituje brisanje transportne kompanije. Očekivano ponašanje je smanjenje broja redaka u tablici koja prikazuje transportne kompanije za 1

```
def test4() -> bool:
    """
    Test delete a transport company.
    A transport company should be deleted.
    :return:
    """

    driver = webdriver.Firefox()
    login(driver, email="admin@admin.com", password="admin")
    driver.get(f"{FRONTEND_URL}/transport-companies")
    sleep(1)

    num_of_companies = len(driver.find_elements(by=By.TAG_NAME, value="tr"))

    rows = driver.find_elements(by=By.TAG_NAME, value="tr")
    # ignore the first row since that's a table header
    for row in rows[1:]:
        cells = row.find_elements(by=By.TAG_NAME, value="td")
        if cells[0].text == "Selenium company":
            cells[4].find_element(by=By.TAG_NAME, value="button").click()
            driver.find_element(by=By.NAME, value="confirm").click()

    testPassed = len(driver.find_elements(by=By.TAG_NAME, value="tr")) == num_of_companies - 1
    driver.close()
    return testPassed
```

Slika 5.5: Izvorni kod 4. testa

Peti test ispituje odjavljivanje korisnika. Očekivano ponašanje je promjena putanja na "/login".

```

def test5() -> bool:
    """
    Test logout functionality.
    :return:
    """

    driver = webdriver.Firefox()
    login(driver, email: "admin@admin.com", password: "admin")
    driver.find_element(by=By.NAME, value="logout").click()

    testPassed = driver.current_url == f"{FRONTEND_URL}/"
    driver.close()
    return testPassed

```

Slika 5.6: Izvorni kod 5. testa

Kod koji prikazuje kako su testovi pokrenuti.

```

def run_test(test_number: int) -> bool:
    testPassed = eval(f"test{test_number}()")
    print(f"Test {test_number} passed: {testPassed}")
    return testPassed

if __name__ == '__main__':
    run_test(1)
    run_test(2)
    run_test(3)
    run_test(4)
    run_test(5)

```

Slika 5.7: Izvršavanje testova

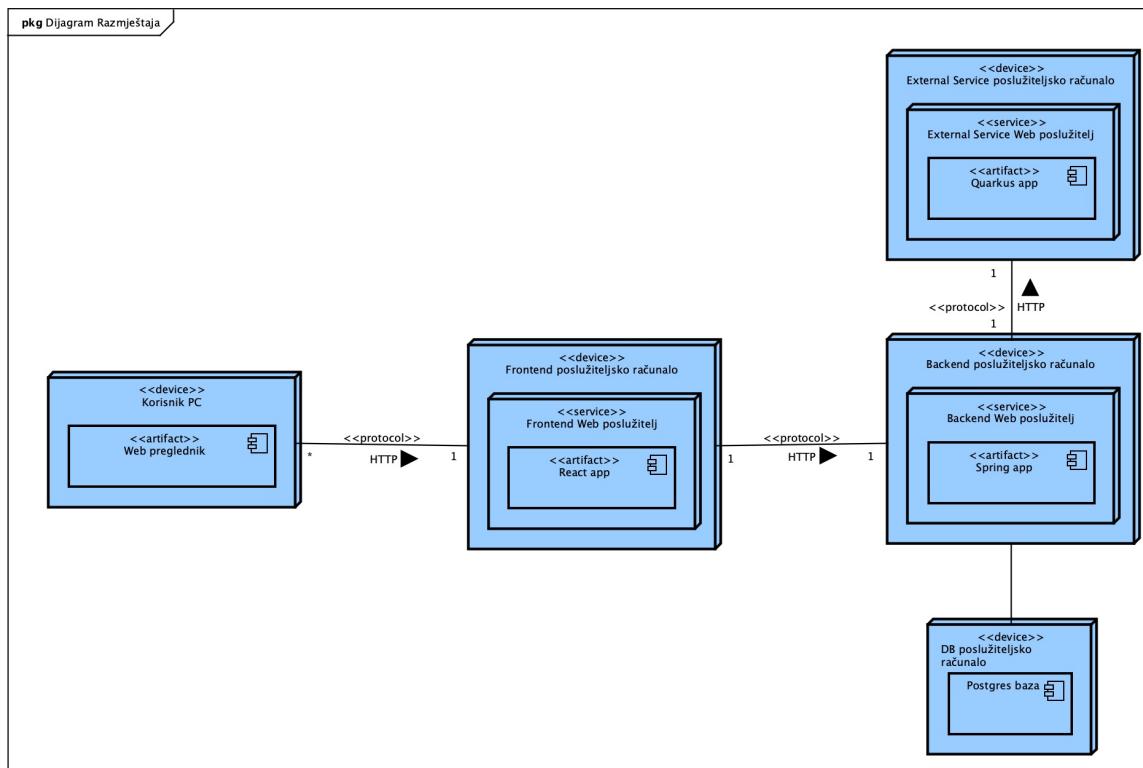
Rezultati izvršenih testova.

```
Test 1 passed: True
Test 2 passed: True
Test 3 passed: True
Test 4 passed: True
Test 5 passed: True
```

Slika 5.8: Rezultati testova

5.3 Dijagram razmještaja

Dijagram razmještaja na slici 5.9 prikazuje topologiju sklopolja i programsku potporu web-aplikacije. Sustav je baziran na arhitekturi "klijent-poslužitelj". Komunikacija između računala korisnika i frontend poslužiteljskog računala, kao i između frontend i backend poslužiteljskog računala, odvija se preko HTTP veze. Korisnici pristupaju web-aplikaciji koristeći web preglednik te im frontend poslužiteljsko računalo, na kojemu se nalazi frontend web poslužitelj, daje odgovarajući prikaz. Na backend poslužiteljskom računalu se nalazi backend web poslužitelj te Spring aplikacija, a Postgres baza nalazi se na poslužiteljskom računalu baze podataka koje je povezano s backend poslužiteljskim računalom. Backend poslužiteljsko računalo također je spojeno na poslužiteljsko računalo vanjskog servisa, koje je postavljeno na sličan način.



Slika 5.9: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Unutar korisničkog sučelja Render web aplikacije, nakon registracije, slijedite ove korake:

1. Kliknite na “**New**” i zatim “**Web service**” za kreiranje nove backend aplikacije, odnosno “**PostgreSQL**” za kreiranje nove aplikacije za bazu podataka.
2. Odaberite opciju za povezivanje s repozitorijem na GitHubu. Preporučljivo je koristiti GitHub zbog integracije koju Render nudi.
3. Nakon povezivanja s GitHub računom, odaberite repozitorij u kojem se nalazi vaš projekt.
4. Unutar sljedećeg prozora, potrebno je:
 - Odrediti ime aplikacije.
 - Odabrati regiju servera na kojoj će aplikacija biti upogonjena.
 - Odrediti granu GitHub repozitorija.
 - Napisati putanju do izvorne mape.
 - Navesti putanju do `.dockerfile` datoteke potrebne Renderu za uspješno upogonjenje.
5. Odredite varijable okoline (*environment variables*), koje su za ovaj projekt u sljedećem obliku:

```
spring.datasource.url=${DB_URL}  
spring.datasource.username=${DB_USERNAME}  
spring.datasource.password=${DB_PASSWORD}
```

6. Nakon što je sve konfiguirano, pokrenite proces upogonjenja. Render će automatski preuzeti kod iz GitHub repozitorija, izgraditi aplikaciju prema definiranim komandama i pokrenuti aplikaciju.
7. Za povezivanje s bazom podataka, Render nudi integraciju s različitim bazama podataka kao servisom. Možete kreirati novu instancu baze direktno preko Render sučelja i povezati je s vašom aplikacijom kroz varijable okoline.

8. Nakon što je aplikacija pokrenuta, Render će pružiti URL na kojem je vaša aplikacija dostupna.
9. Za promjene ili ažuriranje aplikacije, ažurirajte kod na GitHubu. Ukoliko ste tako odabrali u postavkama, Render će automatski detektirati promjene i ponovno upogoniti aplikaciju. Isti proces je moguć i ručno te u bilo kojem trenutku možete preusmjeriti render na neku drugu granu GitHub-a.
10. Proces puštanja vanjskog servisa u pogon je identičan.

Puštanje *frontend*-a u pogon.

1. Na servisu (<https://vercel.com/>) je potrebno napraviti korisnički račun.
2. Potrebno je napraviti novi *hobby* projekt te ga povezati sa GitHub repozitorijem u kojem se nalazi vaš kod.
3. Svaki push na granu "master" će pokrenuti novi deployment na servisu Vercel koji će izgenerirati novu poveznicu gdje će biti dostupan vaš frontend.
4. U root *frontend*/ foldera je potrebno dodati datoteku po imenu .env
Ta datoteka će imati jednu varijablu

VITE_BACKEND_URL=...

koja će reći pokazivati na URL backend-a za potrebe dohvaćivanje podataka putem REST sučelja.

5. Svaki puta kada želite pustiti u pogon novu verziju frontend-a potrebno je samo pushati kod u master granu.

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije pod nazivom “*DentAll*”. Ideja je bila razviti rješenje za učinkovitim upravljanjem smještajem i prijevozom pacijenata zdravstvenog turizma. Nakon 12 tjedana timskog rada, ostvarili smo zadani cilj i projekt je završen. Projekt je imao tri faze.

Prva faza je uključivala okupljanje tima, razgovor o idejama za aplikaciju, izražavanje pojedinačnih interesa i želja za radom u prvom ciklusu predaje. Naglasak je bio na radu dokumentiranja zahtjeva te razvoju backenda. Kvalitetno oblikovanje obrazaca i dijagrama (uključujući obrasce uporabe, sekvensijske dijagrame, model baze podataka i dijagrame razreda) značajno je olakšalo daljnji implementacijski razvoj aplikacije. Timovi su paralelno radili na dokumentaciji i backendu, dok su u području frontenda definirane ideje i implementirani osnovni dijelovi.

Druga faza je počela puno intenzivnije te je naglasak više bio na implementaciji same aplikacije. Jedan tim radio je na frontendu te drugi na backendu. Temeljito izrađena prva faza projekta pridonijela je uštedi vremena tijekom izrade aplikacije, izbjegavajući moguće pogreške i povećavajući učinkovitost timskog rada. U ovoj fazi uspješno smo dovršili implementaciju cijele aplikacije.

U trećoj i konačnoj fazi našeg projekta fokus je bio na dokumentaciji i izradi raznolikih UML dijagrama, detaljno ispitivanje sustava, identifikaciju i ispravak grešaka te implementaciji preostalih funkcionalnosti. U ovoj fazi svi članovi su radili na dokumentaciji. Ova je faza trajala do završetka projekta, prije konačne predaje i kolokviranja drugog ciklusa.

Aplikaciju je moguće proširiti na mnogo načina. Jedna od mogućnosti bi bila dodavanje interakcije s pacijentima to jest omogućiti ocjenjivanje te komentiranje usluga.

Sudjelovanje u ovom projektu bilo je korisno iskustvo za sve članove tima. Iznimno smo zadovoljni postignutim rezultatima i timskim radom koji je pridonio ostvarenju tih rezultata.

Popis literature

Kontinuirano osvježavanje

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

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

Indeks slika i dijagrama

3.1 Dijagram obrasca uporabe, osnovne funkcionalnosti administratora	22
3.2 Dijagram obrasca uporabe, funkcionalnosti smještajnog administratora	23
3.3 Dijagram obrasca uporabe, funkcionalnosti administratora prijevoznih usluga	24
3.4 Dijagram obrasca uporabe, funkcionalnosti korisničkog administratora	25
3.5 Sekvencijski dijagram za UC1	26
3.6 Sekvencijski dijagram za UC4, UC5, UC6, UC9, UC11, UC12	28
3.7 Sekvencijski dijagram za UC15, UC16, UC18, UC19	30
3.8 Sekvencijski dijagram za UC20, UC21, UC22, UC23, UC24, UC25 .	32
4.1 Arhitektura sustava	34
4.2 ER Dijagram baze podataka	41
4.3 Dijagram razreda - dio Controllers	42
4.4 Dijagram razreda - dio Service	43
4.5 Dijagram razreda - dio DTO	44
4.6 Dijagram razreda - dio Requests	45
4.7 Dijagram razreda - dio Model	45
4.8 Dijagram stanja	47
4.9 Dijagram aktivnosti	49
4.10 Dijagram aktivnosti	50
5.1 Zajednički kod testova	56
5.2 Izvorni kod 1. testa	57
5.3 Izvorni kod 2. testa	57
5.4 Izvorni kod 3. testa	58
5.5 Izvorni kod 4. testa	59
5.6 Izvorni kod 5. testa	60
5.7 Izvršavanje testova	60
5.8 Rezultati testova	61

5.9 Dijagram razmještaja	62
6.1 Grafovi promjena	73

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 25. listopada 2023.
- Prisustvovali: Filip Buljan, Roko Gligora, Neven Lukić, Karla Pišonić, Karla Šmuk
- Teme sastanka:
 - dogovor o korištenim tehnologijama
 - dogovor podjеле zadataka
 - komentiranje osnovnih zahtjeva projekta

2. sastanak

- Datum: 1. studenoga 2023.
- Prisustvovali: Filip Buljan, Roko Gligora, Neven Lukić, Maksim Madžar, Karla Pišonić, Karla Šmuk
- Teme sastanka:
 - korištene i organizacija GitHuba
 - podjela zadataka
 - pregled modela baze podataka
 - pregled dizajna korisničkog sučelja

3. sastanak

- Datum: 8. studenoga 2023.
- Prisustvovali: Filip Buljan, Roko Gligora, Neven Lukić, Maksim Madžar, Karla Pišonić, Karla Šmuk
- Teme sastanka:
 - podjela zadataka
 - dogovor oko backend funkcionalnosti
 - razgovor o idejama za puštanje u pogon

4. sastanak

- Datum: 15. studenoga 2023.

- Prisustvovali: Filip Buljan, Roko Gligora, Neven Lukić, Maksim Madžar, Karla Pišonić, Ante Vujčić, Karla Šmuk
- Teme sastanka:
 - podjela zadataka za dovršetak prve revizije
 - razgovor o puštanje u pogon

5. sastanak

- Datum: 10. siječnja 2024.
- Prisustvovali: Filip Buljan, Roko Gligora, Neven Lukić, Maksim Madžar, Karla Pišonić, Ante Vujčić, Karla Šmuk
- Teme sastanka:
 - podjela zadataka za dovršetak druge revizije
 - status razvoja frontenda i backenda
 - dogovor oko podjele dokumentacije

Tablica aktivnosti

Kontinuirano osvježavanje

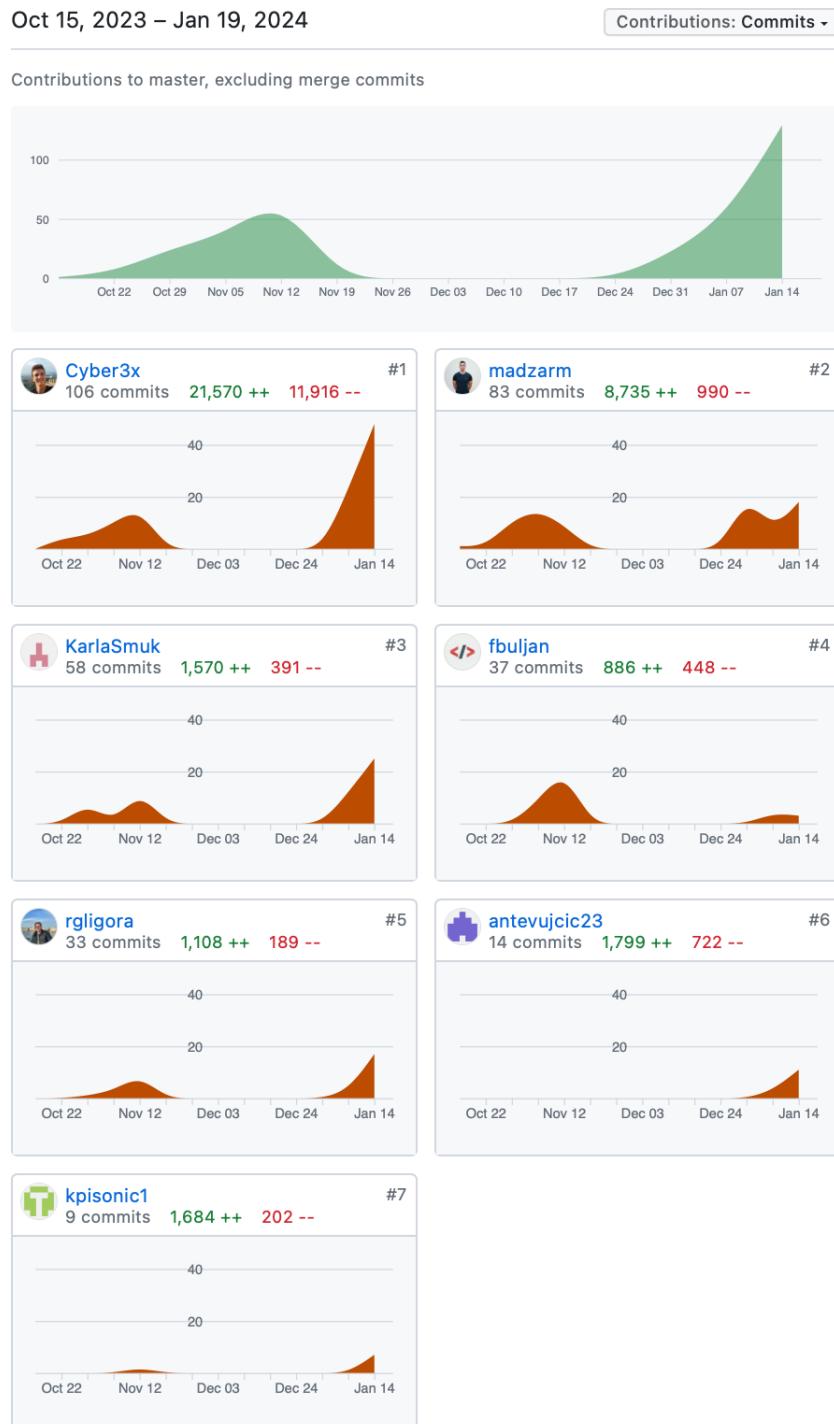
	Neven Lukić	Filip Buljan	Roko Gligora	Maksim Madžar	Karla Pišonić	Karla Šmuk	Ante Vujčić
Upravljanje projektom	6						
Opis projektnog zadatka						5	
Funkcionalni zahtjevi						3	
Opis pojedinih obrazaca	2				4		
Dijagram obrazaca						5	
Sekvencijski dijagrami						10	
Opis ostalih zahtjeva		1					
Arhitektura i dizajn sustava	5						
Baza podataka	10		4	3			
Dijagram razreda			10	4			
Dijagram stanja			1				3
Dijagram aktivnosti						2	
Dijagram komponenti					3		
Korištene tehnologije i alati					2		
Ispitivanje programskog rješenja	1			3			
Dijagram razmještaja		2					
Upute za puštanje u pogon	1	2	1				
Dnevnik sastajanja	2						

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Neven Lukić	Filip Buljan	Roko Gligora	Maksim Madžar	Karla Pišonić	Karla Šmuk	Ante Vujčić
Zaključak i budući rad						1	
Popis literature							
Backend		20	20	45			
Frontend	60		2		8	30	20
Puštanje u pogon	5			3			
Istraživanje informacija/tehnologija	5	15	4		4	7	

Dijagrami pregleda promjena



Slika 6.1: Grafovi promjena