

Programsko inženjerstvo
Ak. god. 2023./2024.

DentAll
Dokumentacija, Rev. 2.0

Grupa: *Error 404: Koordinator not found*

Voditelj: *Tomislav Pranjić*

Datum predaje: <19>. <01>. <2024>.

Nastavnik: <*Goran Rajić*>

Sadržaj

1 Dnevnik promjena dokumentacije	2
2 Opis projektnog zadatka	4
3 Specifikacija programske potpore	7
3.1 Funkcionalni zahtjevi	7
3.1.1 Obrasci uporabe	10
3.1.2 Sekvencijski dijagrami	21
3.2 Ostali zahtjevi	25
4 Arhitektura i dizajn sustava	26
4.1 Baza podataka	27
4.1.1 Opis tablica	27
4.1.2 Dijagram baze podataka	29
4.2 Dijagram razreda	30
4.3 Dijagram stanja	36
4.4 Dijagram aktivnosti	37
4.5 Dijagram komponenti	39
5 Implementacija i korisničko sučelje	40
5.1 Korištene tehnologije i alati	40
5.2 Ispitivanje programskog rješenja	41
5.2.1 Ispitivanje komponenti	41
5.2.2 Ispitivanje sustava	44
5.3 Dijagram razmještaja	49
5.4 Upute za puštanje u pogon	50
6 Zaključak i budući rad	58
Popis literature	59
Indeks slika i dijagrama	61
Dodatak: Prikaz aktivnosti grupe	62

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Naslovna stranica popunjena s osnovnim podatcima	Josip Mihelčić	23.10.2023.
0.1.1	Osvježen dnevnik sastajanja.	Ante Sorić Diego Mišetić	31.10.2023.
0.2	Opis projektnog zadatka.	Ante Sorić Diego Mišetić	31.10.2023.
0.3	Dodani ostali zahtjevi.	Ivan Ćorluka	04.11.2023.
0.4	Dodani funkcionalni zahtjevi.	Nikola Perić	05.11.2023.
0.5	Opisi obrazaca uporabe	Tomislav Pranjić Josip Mihelčić	6.11.2023.
0.6	Sekvencijski dijagrami	Diego Mišetić	10.11.2023.
0.7	Dijagram baze podataka	Tomislav Pranjić	12.11.2023.
0.8	Opis tablica baze podataka	Diego Mišetić	14.11.2023.
0.9	Dodan dijagram klasa i ažuriran broj sati	Josip Mihelčić	17.11.2023.
0.10	Sekvencijski dijagrami	Diego Mišetić	10.11.2023.
0.11.1	Započeo dijagrame razreda	Ante Sorić	12.11.2013.
0.11.2	Nastavak dijagrama razreda	Ante Sorić	16.11.2013.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Tomislav Pranjić	17.11.2023.
1.1	Dijagrami stanja, aktivnosti i komponenti	Diego Mišetić	11.01.2024.
1.2	Dijagram razmještaja	Diego Mišetić	12.01.2024.
1.3	Generalna revizija strukture dokumenta	Tomislav Pranjić	13.1.2024.
1.4	Dodane upute za puštanje u pogon	Tomislav Pranjić	17.01.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.5	Dodano ispitivanje	Tomislav Pranjić	19. 01. 2024.
2.0	Konačni tekst predloška dokumentacije	Tomislav Pranjić	19.01.2024.

2. Opis projektnog zadatka

Cilj ovog projekta razvoj je aplikacije za evidenciju i koordinaciju smještaja i prijevoza korisnika usluga zdravstvenog turizma. U današnje vrijeme turizam te sve aktivnosti vezane uz turizam sve su više prisutne u većini država. Mnogi ljudi svojevoljno odlaze u druge države kako bi odradili određeni medicinski postupak, bilo to zbog manje cijene, bolje usluge ili nekog drugog razloga. Ogroman broj klinika pokušava pronaći više klijenta čak i preko granice. Veliki broj stomatoloških klinika u urbanim centrima kao i na Jadranskoj obali povećava obim posla oglašavanjem u inozemstvu i pružanjem usluga strancima.

Ovom aplikacijom pomoglo bi se ne samo turistima, kojima je potrebna usluga zdravstvenog turizma, već i domaćim klinikama kojima bi se obujam posla povećao, što bi značilo i mogućnost za veće plaće zaposlenicima i/ili zapošljavanjem većeg broja ljudi. U aplikaciji bi bila ugrađena kompletna organizacija prijevoza i smještaja, što bi znatno povećalo privlačnost produkta krajnjim korisnicima, a povećala bi se kompetitivnost u odnosu na druge pružatelje sličnih usluga.

Danas možemo vidjeti kako je zdravstveni turizam prisutan u gotovo svakoj državi. Neki od primjera su medicinska rehabilitacija u poznatim toplicama BiH, Spa u Belgiji, razna lječilišta termalni vodama Italije, no isto tako postoje brojni primjeri i u Hrvatskoj. Samo jedan od primjera u Hrvatskoj su stomatološke ordinacije na Jadranu.

Hrvatska privlači turiste zbog svojih ljekovitih termalnih izvora u Toplicama, slikovitih wellness odmarališta na obali Jadranskog mora te renomiranih stomatoloških klinika u Zagrebu. Izvan Hrvatske, Turska nudi putnicima ljekovite termalne kupke u Pamukkaleu, dok Tajland impresionira posjetitelje svojim spa centrima na tropskim otocima, pružajući nezaboravna iskustva u zdravstvenom turizmu.

Korisnici koji bi mogli biti zainteresirani za ovu uslugu su npr. ljudi kojima je potrebna određena zdravstvena usluga kao što su wellness, toplice, stomatološka usluga itd., a koje bi privlačila niža cijena tretmana nego u njihovim državama ili bi jednostavno htjeli razgledati ljepote Hrvatske te usput obaviti zdravstvenu uslugu koja im je potrebna.

Aplikacija koja na izbor daje sve moguće prijevoznike, podatke, rutu putovanja, raspoloživi smještaj i još mnogo toga uvelike bi pojednostavila sam postupak planiranja putovanja korisnicima zdravstvenog turizma. Sve što je potrebno je prijaviti se na aplikaciju, odabratи datum i vrijeme planiranog dolaska, a aplikacija bi se pobrinula za sve ostalo. U par koraka možete se riješiti problema smještaja i prijevoza koji su Vam potrebni kako bi medicinska usluga protekla na najbolji mogući način. U aplikaciji postoje tri uloge:

- Smještajni administrator

- Administrator prijevoznih usluga
- Korisnički administrator

Smještajni administrator unosi podatke o smještaju koji je u to vrijeme raspoloživ. Svaka klinika ima raspoložive smještajne objekte, bilo u najmu ili privatnom vlasništvu. Ova vrsta administratora može mijenjati sve što se tiče samog smještaja, od kapaciteta do vlasništva samog smještajnog objekta. Isto tako smještajni administrator može i obrisati smještajnu jedinicu. Podaci smještaja sastoje se od:

- Tipa stana
 - Veličina i vlasništvo
- Kategorije opremljenosti
 - Broj zvjezdica (1 – najslabije opremljen do 5- najbolje opremljen)
 - Također je opisano sve što stan dodatno posjeduje
- Adrese
 - Adresa zgrade, kat, broj stana
- Vlasništva
 - Privatno ili u najmu
- Vremenski period dostupnosti za korištenje

Također je omogućen grafički prikaz geografskog položaja nekretnine korištenjem Google Maps usluge.

Uloga **administratora prijevoznih usluga** u aplikaciji regulacija je samih prijevoznika. Podatci koje unosi u aplikaciju su:

- osobni podaci prijevoznika: ime, prezime, broj telefona
- tip vozila, kapacitet te registracijska tablica
- radno vrijeme u kojem je prijevoznik dostupan

Prijevoznici se mogu brisati, a njihovi neosnovni podaci mijenjati (kao što su radno vrijeme te radni dani u tjednu).

Za definiciju korisnika medicinske usluge zaslužan je **korisnički administrator** koji unosi njihove osobne podatke te podatke bitne za rad. Ime, prezime i osnovni kontaktni podaci korisnika, vrijeme i mjesto dolaska i odlaska u/iz zemlje te preferencije vezano uz veličinu i kvalitetu smještaja.

Detalji njihovih tretmana ne unoše se ručno već se aplikacijskim sučeljem preuzimaju iz aplikacije za evidenciju medicinskih usluga. Sučelje je realizirano kao umjetno ispitno sučelje, a podaci o tretmanima popunjavaju se korištenjem isitnih primjera.

Korisničkom administratoru iznimno je važno vrijeme i mjesto dolaska određenog korisnika kao i vrsta smještaja za koju je sam korisnik zainteresiran.

Ova vrsta administratora može definirati ostale korisnike te im davati određene uloge. Jednom kada se svi podaci unesu u aplikaciju, aplikacija predlaže određenu smještajnu jedinicu te označava je

zauzetom u tom vremenskom periodu.

Aplikacija periodički provjerava status unosa medicinskih termina komunikacijom s aplikacijom medicinskih usluga. Ako se primi odgovor o zaključanom planu medicinskih usluga s listom medicinskih termina, potrebno je pridijeliti raspoložive prijevoznike svakom od termina (prijevoz od smještaja u ordinaciju te povratak) te označiti prijevoznike zauzetima u tim terminima.

Nakon što je ukupan plan završen, šalje se poruka elektroničke pošte korisniku medicinske usluge podacima ukupnog plana njihovog puta uključujući podatke o prijevozima i smještaju. Također se svakom od prijevoznika šalju posebne poruke s kontaktnim podacima korisnika kao i vremenima i adresama smještaja korisnika.

Ova aplikacija mogla bi se nadograditi i unaprijediti u različitim smjerovima. Neke od ideja za unapređenje aplikacije su:

- Implementacija i unapređenje aplikacije za mobilne uređaje (primarno je samo za ekrane računala) uz pomoć responzivnog dizajna u HTML-u
- Nakon provedenog boravka u nekom od smještaja, korisnici mogu dati određenu ocjenu smještaju i/ili prijevozniku te uz ocjenu ostaviti neobavezni odgovarajući komentar
- U aplikaciji postaviti mogućnost kartičnog plaćanja pri rezervaciji određene zdravstvene usluge
- Mogućnost virtualnog obilaska smještaja prije nego što se klijenti odluče za rezervaciju baš tog smještaja
- Dodavanje opcije za razgled grada u kojem su smješteni (glavne ulice, posebne znamenitosti i slično)
- Unaprijediti web stranicu na više različitih jezika, kako je aplikacija namijenjena za korisnike zdravstvenog turizma to će biti bolje što je više jezika implementirano u aplikaciji

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Zdravstvena ustanova
2. Prijevoznik
3. Vlasnik smještaja
4. Administrator
 - (a) Smještajni administrator
 - (b) Administrator prijevoznih usluga
 - (c) Korisnički administrator
5. Korisnik
6. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:**1. Administrator (inicijator) se može:**

- (a) prijaviti u sustav nakon čega dobija ovlasti ovisno u koje sve vrste administratora spada (smještajni administrator, administrator prijevoznih usluga, korisnički administrator)

2. Smještajni administrator (inicijator) može:

- (a) definirati nove smještajne kapacitete
- (b) mijenjati osnovne podatke smještaja (tip stana, kategorija opremljenosti, adresa, vremenski period dostupnosti)
- (c) obrisati smještaj
- (d) definirati druge korisnike i dodijeliti im uloge (dodavanje novih admina)

3. Administrator prijevoznih usluga (inicijator) može:

- (a) unositi podatke o prijevoznicima (osnovni osobni podatci prijevoznika, kontakt podatci prijevoznika, vrsta i kapacitet prijevoznog sredstva, radno vrijeme u kojem su raspoloživi)
- (b) obrisati prijevoznika
- (c) mijenjati neosnovne podatke o prijevozniku

4. Korisnički administrator (inicijator) može:

- (a) unositi podatke o korisnicima medicinskih usluga (ime, prezime, kontakt, vrijeme i mjesto dolaska/odlaska, preferencije za veličinu i kvalitetu smještaja)

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o administratorima i njihovim ovlastima
- (b) pohranjuje sve podatke o korisnicima, smještaju i prijevoznicima
- (c) nakon unosa novog korisnika od korisničkog admina, pridjeljuje se raspoloživa smještajna jedinica i označava se zauzetom u tom periodu
- (d) periodički provjerava status unosa medicinskih termina komunikacijom s aplikacijom medicinskih usluga
- (e) kada se primi odgovor o zaključanom planu medicinskih usluga s listom termina, pridjeljuju se raspoloživi prijevoznici i označavaju se zauzetim. Zatim se šalju poruke elektronične pošte svakom od prijevoznika s kontaktnim podatcima korisnika kao i o vremenima i adresama smještaja korisnika, te korisniku usluge poruka sa podatcima o prijevoznicima i smještaju

6. API medicinskih usluga (sudionik):

- (a) detalji tretmana se preuzimaju iz aplikacije za evidenciju medicinskih usluga

7. Google Maps (sudionik):

- (a) preko podataka o smještaju omogućava grafički prikaz lokacije smještaja

3.1.1 Obrasci uporabe

UC1 - Prijava

- **Glavni sudionik:** Administrator
- **Cilj:** Prijava administratora u sustav
- **Sudionici:** Administrator i baza podataka
- **Preduvjet:** Nema
- **Opis osnovnog tijeka:**
 1. Otvaranje aplikacije unutar web preglednika
 2. Unos korisničkog imena i lozinke
 3. Podnošenje zahtjeva za prijavu klikom na gumb
 4. Korisnik biva preusmjeren na početnu stranicu
- **Opis mogućih odstupanja:**
 - 2.a Uneseni podaci ne odgovaraju traženom formatu
 1. Ispis upozorenja o krivom formatu i onemogućen gumb za prijavu sve dok podatci ne zadovoljavaju traženi format
 - 4.a Korisnički podatci su neispravni ili nisu prepoznati u bazi podataka
 1. Korisnika ne preusmjeravamo na početnu stranicu već mu samo ispisujemo da prijava nije uspjela.

UC2 - Dodavanje novog administratora

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Dodati korisničke podatke novog administratora i dodijeliti mu odgovarajuće uloge
- **Sudionici:** Smještajni administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju za dodavanje novog administratora
 2. Unosi korisničke podatke novog administratora
 3. Označuje uloge dodijeljene novom administratoru
 4. Podnosi zahtjev za unosom novog administratora u bazu podataka
 5. Sva polja se postavljaju na početne vrijednosti
- **Opis mogućih odstupanja:**
 - 2.a Uneseni podaci ne odgovaraju traženom formatu
 1. Ispis upozorenja o krivom formatu i onemogućen gumb za dodavanje novog administratora sve dok podatci ne zadovoljavaju traženi format
 - 3.a Nije označena ni jedna uloga
 1. Onemogućen gumb za dodavanje novog administratora sve dok nije označena barem jedna uloga novog administratora
 - 4.a Sustav vraća grešku prilikom dodavanja novog administratora
 1. Ispisati tekst greške
 2. Čekati na novi pokušaj podnošenja zahtjeva(Korak 4.)

UC3 - Unos raspoloživog smještaja

- **Glavni sudionik:** Smještajni administrator

- **Cilj:** Unos novog smještaja u bazu podataka
- **Sudionici:** Smještajni administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju za dodavanje novog smještaja
 2. Odabir tipa stana
 3. Odabir kategorije stana
 4. Unos maksimalnog kapaciteta stana
 5. Unos adrese na kojoj se stan nalazi(UC4)
 6. Unos podataka o zgradbi(Broj kata, broj stana, dostupnost lifta, opis)
 7. Odabir tipa vlasništva
- **Opis mogućih odstupanja:**
 - 2.a Nije odabran ni jedan tip stana
 1. Onemogućeno podnošenje zahtjeva za dodavanjem smještaja u bazu
 2. Ispis upozorenja o odabiru
 - 3.a Nije odabran ni jedna kategorija stana
 1. Onemogućeno podnošenje zahtjeva za dodavanjem smještaja u bazu
 2. Ispis upozorenja o odabiru
 - 4.a Uneseni podatak nije u odgovarajućem formatu
 1. Onemogućeno podnošenje zahtjeva za dodavanjem smještaja u bazu
 2. Ispis greške o krivom formatu
 - 7.a Odabran tip privatnog vlasništva
 1. Unos podataka vezanih za stan u vlasništvu(UC3.1)
 - 7.b Odabran tip stana u najmu
 1. Unos podataka vezanih za stan u najmu(UC3.2)

UC3.1 -Smještaj u vlasništvu

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Unos podataka o smještaju u vlasništvu
- **Sudionici:** Smještajni administrator i baza podataka
- **Preduvjet:** UC1: Prijava i UC3: Unos raspoloživog smještaja
- **Opis osnovnog tijeka:**
 1. Podnošenje zahtjeva za dodavanje smještaja u bazu podataka
 2. Preusmjeravanje na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Dogodila se greška prilikom dodavanja smještaja
 1. Ispis teksta greške administratoru
 2. Povratak na 1. korak

UC3.2 -Smještaj u najmu

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Unos podataka o smještaju u najmu
- **Sudionici:** Smještajni administrator i baza podataka

- **Preduvjet:** UC1: Prijava i UC3: Unos raspoloživog smještaja
- **Opis osnovnog tijeka:**
 1. Unos vremena dostupnosti stana
 2. Podnošenje zahtjeva za dodavanje smještaja u bazu podataka
 3. Preusmjeravanje na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Nije uneseno vrijeme dostupnosti
 1. Onemogućeno podnošenje zahtjeva za dodavanje smještaja u bazi
 2. Ispis greške o dostupnosti
 - 2.a Dogodila se greška prilikom dodavanja smještaja
 1. Ispis teksta greške administratoru
 2. Povratak na 2. korak

UC4 -Odabir lokacije

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Unos podataka o lokaciji i prikaz lokacije na karti
- **Sudionici:** Smještajni administrator i *Google Maps*
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Unos podataka o lokaciji(Grad, ulica, kućni broj)
 2. Prikaz unesene lokacije na krati pomoću servisa *Google Maps*
- **Opis mogućih odstupanja:**
 - 1.a Podatci ne odgovaraju traženom formatu
 1. Ne upućujemo zahtjev na *Google Maps*
 2. Onemogućen odabir lokacije
 3. Ispis greške u podacima
 - 2.a Dogodila se greška prilikom prikaza lokacije
 1. Onemogućen odabir lokacije
 2. Ispis greške dobivene od *Google Mapsa*

UC5 -Pregled podataka u bazi

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled svih unesenih podataka u bazi
- **Sudionici:** Administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Administrator odabire koju grupu podataka hoće vidjeti
 2. Ako administrator odabere filter ili specificira sortiranje podataka podatci se sortiraju i ponovno prikažu(UC5.1)
 3. Ako administrator odabere brisanje određenog unosa koristi se UC6
 4. Ako administrator odabere uređivanje određenog unosa koristi se UC7

UC5.1 - Filtriranje i sortiranje podataka za prikaz

- **Glavni sudionik:** Administrator

- **Cilj:** Pregled sortiranih i filtriranih podataka
- **Sudionici:** Administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Odabir ponuđenih ograničenja na skup podataka (raspon vrijednosti, podaci specifično grupirani i slično)
 2. Odabir načina sortiranja podataka (uzlazno, silazno)
 3. Prikaz podataka

UC6 -Brisanje unosa

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje podataka iz baze podataka
- **Sudionici:** Administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Prikaz odabranog unosa kojeg treba obrisati
 2. Administrator potvrđuje brisanje unosa
 3. Prikaz ažuriranih podataka
- **Opis mogućih odstupanja:**
 - 2.a Administrator ne potvrđuje brisanje
 1. Povratak na prikaz podataka bez brisanja

UC7 -Promjena podataka

- **Glavni sudionik:** Administrator
- **Cilj:** Promjena podataka u bazi podataka
- **Sudionici:** Administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Prikaz promjenjivih podataka odabranog unosa
 2. Administrator izmjenjuje potrebne podatke
 3. Podnošenje zahtjeva za izmjenom podataka
- **Opis mogućih odstupanja:**
 - 2.a Novi podatci ne odgovaraju traženom formatu
 1. Ispis upozorenja o krivom formatu
 2. Onemogućeno podnošenje zahtjeva za promjenom sve dok format nije zadovoljen
 - 3.a Administrator odustaje od izmjene podataka
 1. Povratak na prikaz podataka bez slanja zahtjeva za izmjenom podataka

UC8 -Unos podataka o prijevoznicima

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Stvoriti novi zapis u bazi podataka s podacima o prijevozniku
- **Sudionici:** Prijevozni administrator i baza podataka
- **Preduvjet:** UC1: Prijava

- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju za dodavanje novog prijevoznika
 2. Unos osobnih podataka, kontaktnih podataka i podataka o vozilu
 3. Odabir radnog vremena i radnih dana u tjednu
 4. Podnošenje zahtjeva za upisom prijevoznika u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Podatci ne odgovaraju traženom formatu
 1. Ispis upozorenja o krivom formatu
 2. Onemogućeno podnošenje zahtjeva za upisom podataka
 - 3.a Nije ispravno odabранo radno vrijeme
 1. Ispis upozorenja o krivom radnom vremenu
 2. Onemogućeno podnošenje zahtjeva za upisom podataka
 - 4.a Dogodila se greška prilikom upisa podataka u bazu
 1. Ispis teksta greške

UC9 -Unos podataka o korisnicima

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Stvoriti novi zapis u bazi podataka s podacima o korisniku
- **Sudionici:** Korisnički administrator i baza podataka
- **Preduvjet:** UC1: Prijava
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju za dodavanje novog korisnika
 2. Unos osobnih i kontaktnih podataka
 3. Odabir lokacije dolaska korisnika(UC4)
 4. Odabir vremena dolaska korisnika
 5. Odabir lokacije odlaska korisnika(UC4)
 6. Odabir vremena odlaska korisnika
 7. Odabir preferencija korisnika
 8. Podnošenje zahtjeva za upisom korisnika u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Podatci ne odgovaraju traženom formatu
 1. Ispis upozorenja o krivom formatu
 2. Onemogućeno podnošenje zahtjeva za upisom podataka
 - 3.a i 5.a Nije odabrana lokacija
 1. Onemogućeno podnošenje zahtjeva za upisom podataka
 - 4.a i 6.a Nije ispravno odabranо vrijeme dolaska
 1. Ispis upozorenja o krivom unosu vremenu
 2. Onemogućeno podnošenje zahtjeva za upisom podataka
 - 8.a Dogodila se pogreška prilikom upisa podataka
 1. Ispis teksta greške

UC10 -Usklađivanje podataka

- **Glavni sudionik:** Baza podataka

- **Cilj:** Uskladiti podatke iz baze podataka s podacima u bazi podataka pružatelja medicinske usluge
- **Sudionici:** Baza podataka i API medicinske usluge
- **Preduvjet:** -
- **Pokretač:** Prošlo je određeno vrijeme od zadnjeg usklađivanja
- **Opis osnovnog tijeka:**
 1. Baza podataka šalje upit API-ju s podacima korisnika za koje nema zabilježen ID
 2. API odgovara s traženim popisom
 3. Baza uskladjuje ID-eve iz popisa
 4. Baza podataka šalje upit tražeći popis zaključanih medicinskih termina zabilježenih nakon određenog datuma(Datum zadnjeg usklađivanja podataka)
 5. API odgovara s traženim popisom
 6. Baza podataka obrađuje sve zaključane termine(UC11)
- **Opis mogućih odstupanja:**
 - 1.a Nema korisnika s nezabilježenim ID-em u bazi podataka
 1. Ne šalje se upit
 2. Prelazimo na korak 4.
 - 2.a i 5.a Nismo dobili odgovor
 1. Nakon 10 sekundi pokušamo ponovno korak 1./4.
 2. U slučaju 3 ili više neuspjeha, šaljemo obavijest smještajnom administratoru(UC10.1)
 3. Prekidamo proces usklađivanja podataka

UC10.1 - Slanje obavijesti smještajnom administratoru

- **Glavni sudionik:** Baza podataka
- **Cilj:** Obavijestiti smještajnog administratora o nemogućnosti usklađivanja s API-em medicinskih usluga
- **Sudionici:** Baza podataka, smještajni administrator i API medicinskih usluga
- **Preduvjet:** UC1: Prijava
- **Pokretač:** API ne odgovara na upit 3 ili više puta kod usklađivanja podataka (UC10)
- **Opis osnovnog tijeka:**
 1. Slanje obavijesti smještajnom administratoru na mail s uključenim vremenom i sadržajem upita API-u medicinskih usluga

UC11 -Zaključan plan usluge

- **Glavni sudionik:** Baza podataka
- **Cilj:** Slanje poruke u obliku emaila korisniku i prijevozniku sa svim potrebnim podacima
- **Sudionici:** Baza podataka, korisnik i prijevoznik
- **Preduvjet:** -
- **Pokretač:** Dobiven zapis o zaključanom terminu tretmana tokom UC10: Usklađivanje podataka
- **Opis osnovnog tijeka:**
 1. Odabir prijevoznika dostupnog u terminu dolaska korisnika
(prijevoznik1)

2. Odabir prijevoznika dostupnog u terminu tretmana ±2h
(prijevoznik2)
3. Odabir prijevoznika dostupnog u terminu odlaska korisnika
(prijevoznik3)
4. Slanje poruke svakom od prijevoznika s potrebnim podacima
5. Slanje poruke korisniku s podacima ukupnog plana njihovog puta i kontaktnim podacima prijevoznika

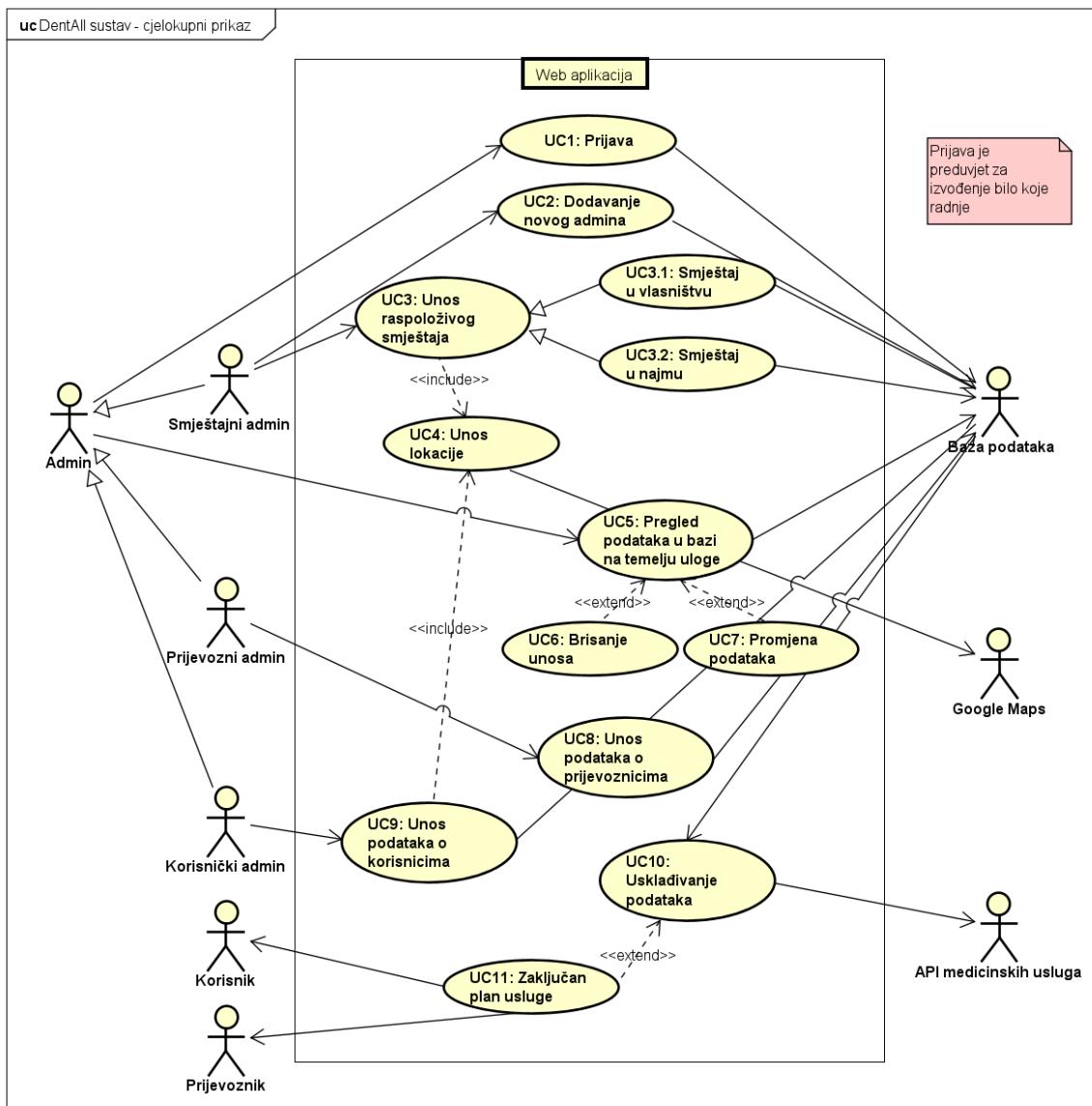
- **Opis mogućih odstupanja:**

- 1.a, 2.a i 3.a Ne postoji prijevoznik dostupan u tom terminu
 1. Šaljemo obavijest prijevoznom administratoru o nemogućnosti uparivanja prijevoznika s korisnikom
 2. Označavamo taj termin nedovršenim
 3. Čekamo prijevoznog administratora da poduzme akciju(UC11.1)
- 4.a Nisu sva tri prijevoznika različita
 1. Ne šaljemo prijevozniku više od jedne poruke već sve podatke vezane za tog prijevoznika grupiramo u jednu poruku

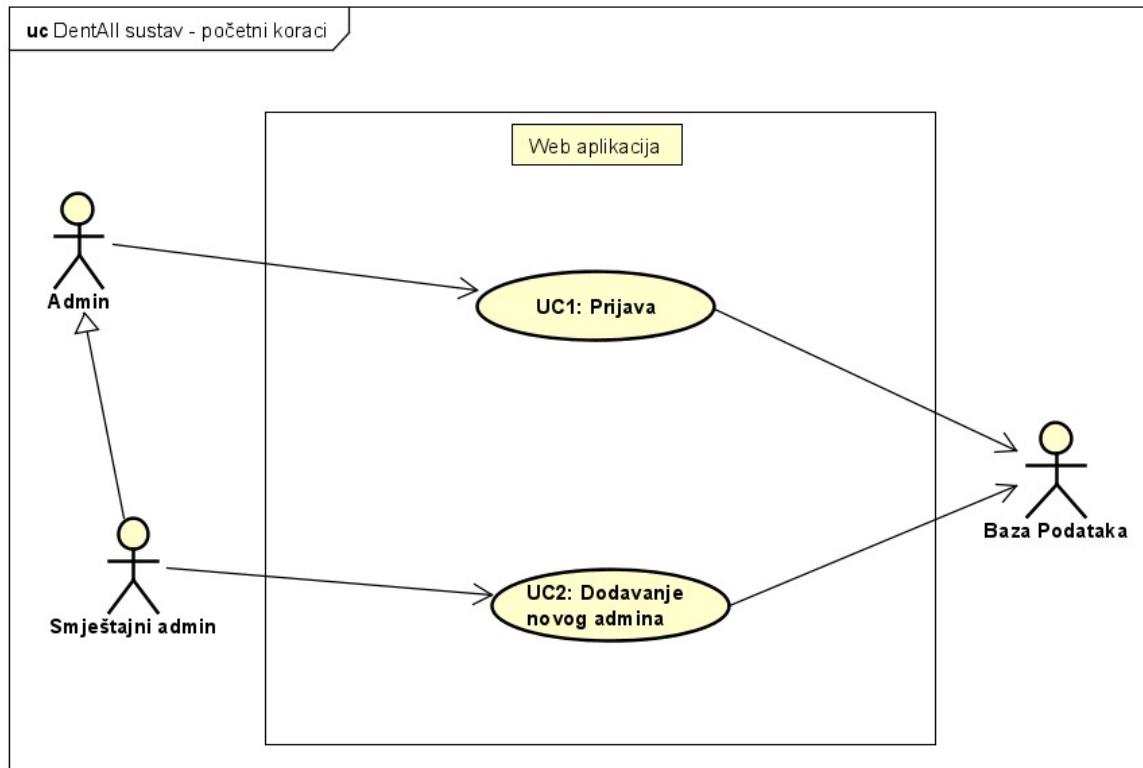
UC11.1 - Slanje obavijesti prijevoznom administratoru

- **Glavni sudionik:** Baza podataka
- **Cilj:** Obavijestiti prijevoznog administratora o nepostojanju prijevoznika s odabranim terminom
- **Sudionici:** Baza podataka, prijevozni administrator
- **Preduvjet:** UC1: Prijava
- **Pokretač:** Poslana obavijest prijevoznom administratoru
- **Opis osnovnog tijeka:**
 1. Slanje obavijesti prijevoznom administratoru na mail s uključenim detaljima o terminu (vrijeme i mjesto tretmana)
 2. Prijevozni administrator upisuje prijevoznika za odabrani termin
- **Opis mogućih odstupanja**
 - 2.a Prijevozni administrator ne upisuje prijevoznika za odabrani termin
 1. Obavijestiti korisnika na mail o nepostojanju prijevoznika za odabrani termin

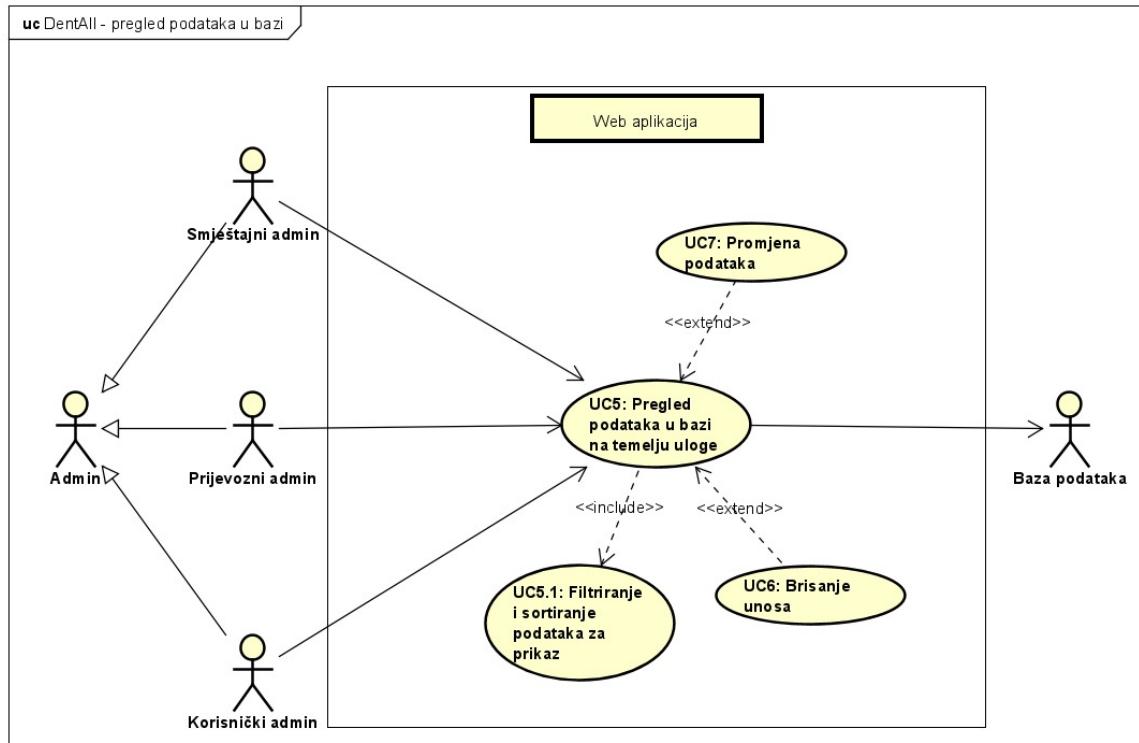
Dijagrami obrazaca uporabe



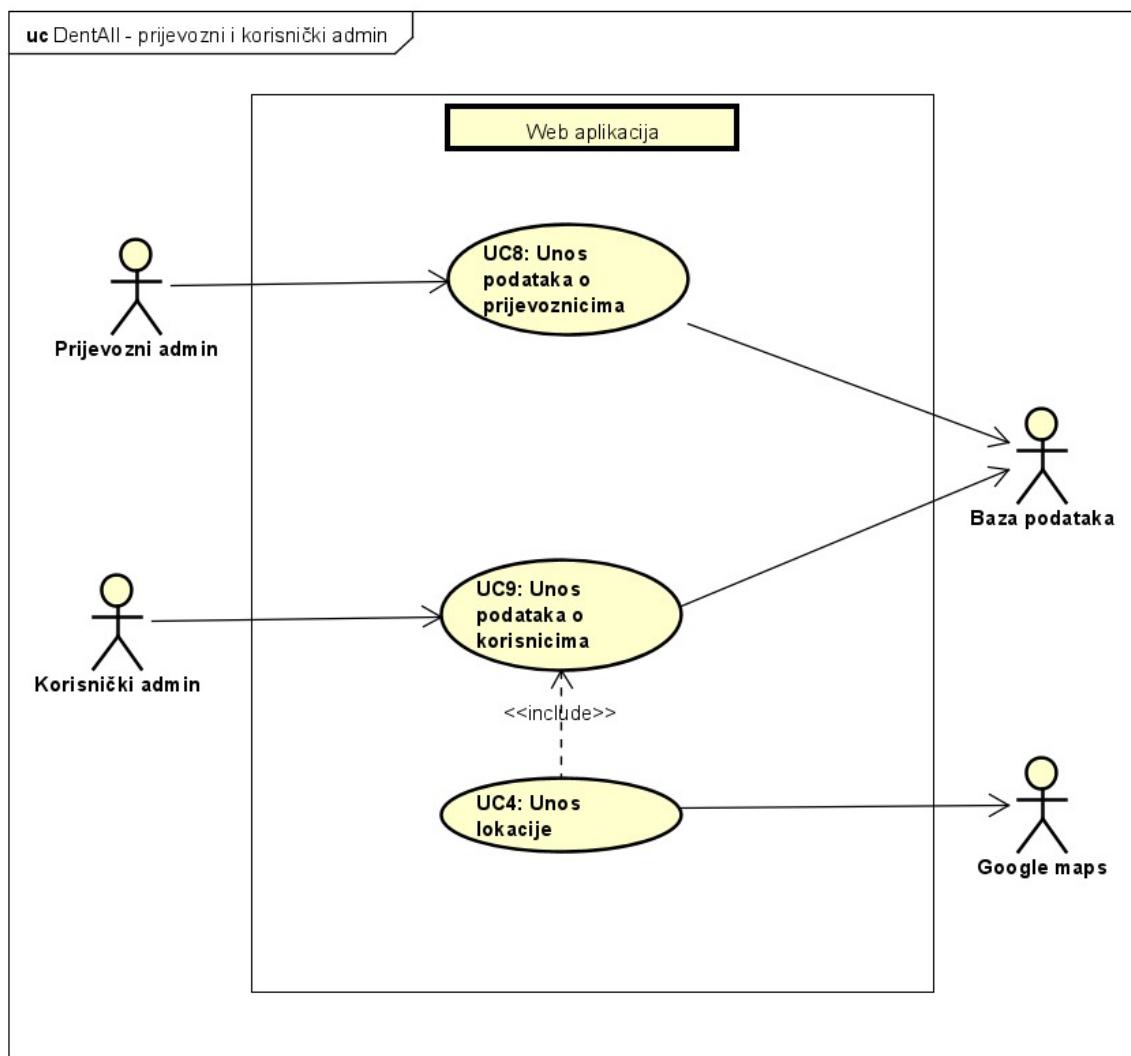
Slika 3.1: Sveobuhvatni prikaz



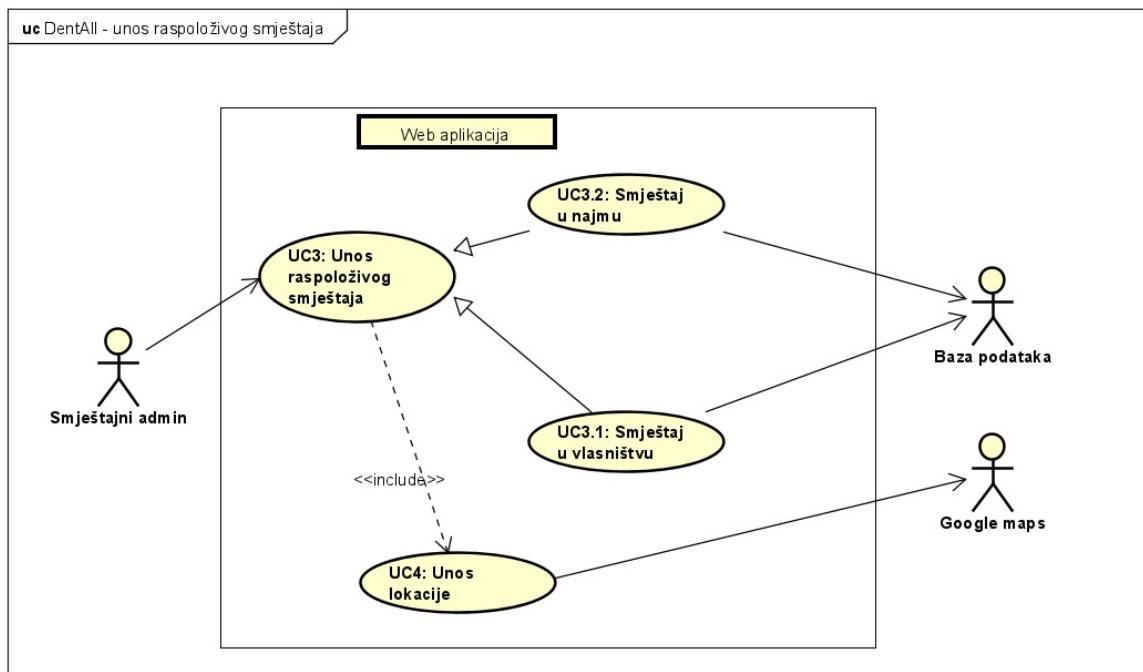
Slika 3.2: Početni koraci



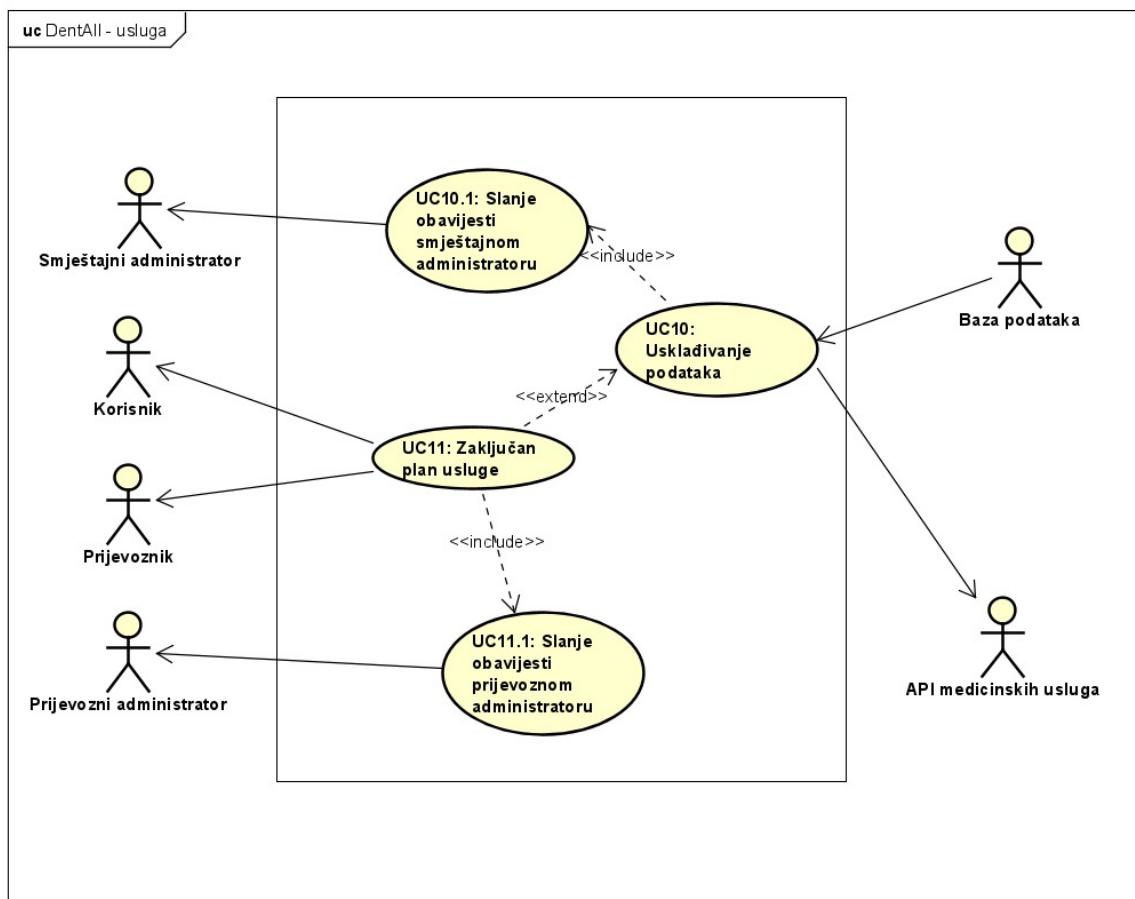
Slika 3.3: Pregled podataka u bazi podataka



Slika 3.4: Prikaz uloga korisničkog i prijevoznog administratora



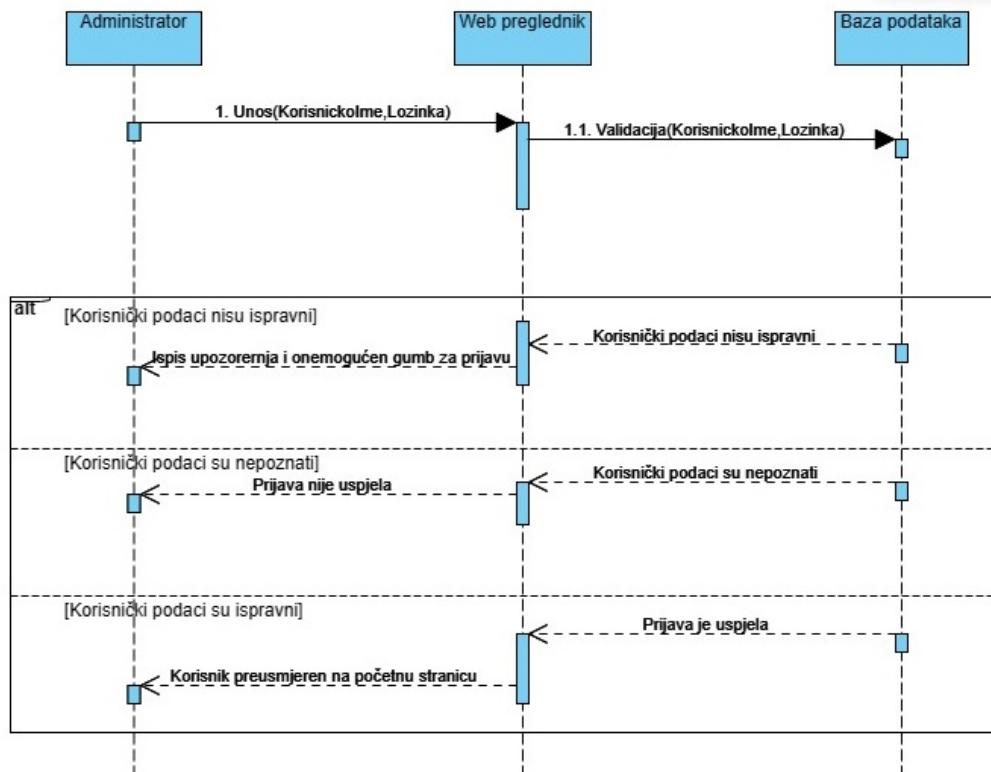
Slika 3.5: Prikaz unosa raspoloživog smještaja



Slika 3.6: Prikaz korisničke strane

3.1.2 Sekvencijski dijagrami

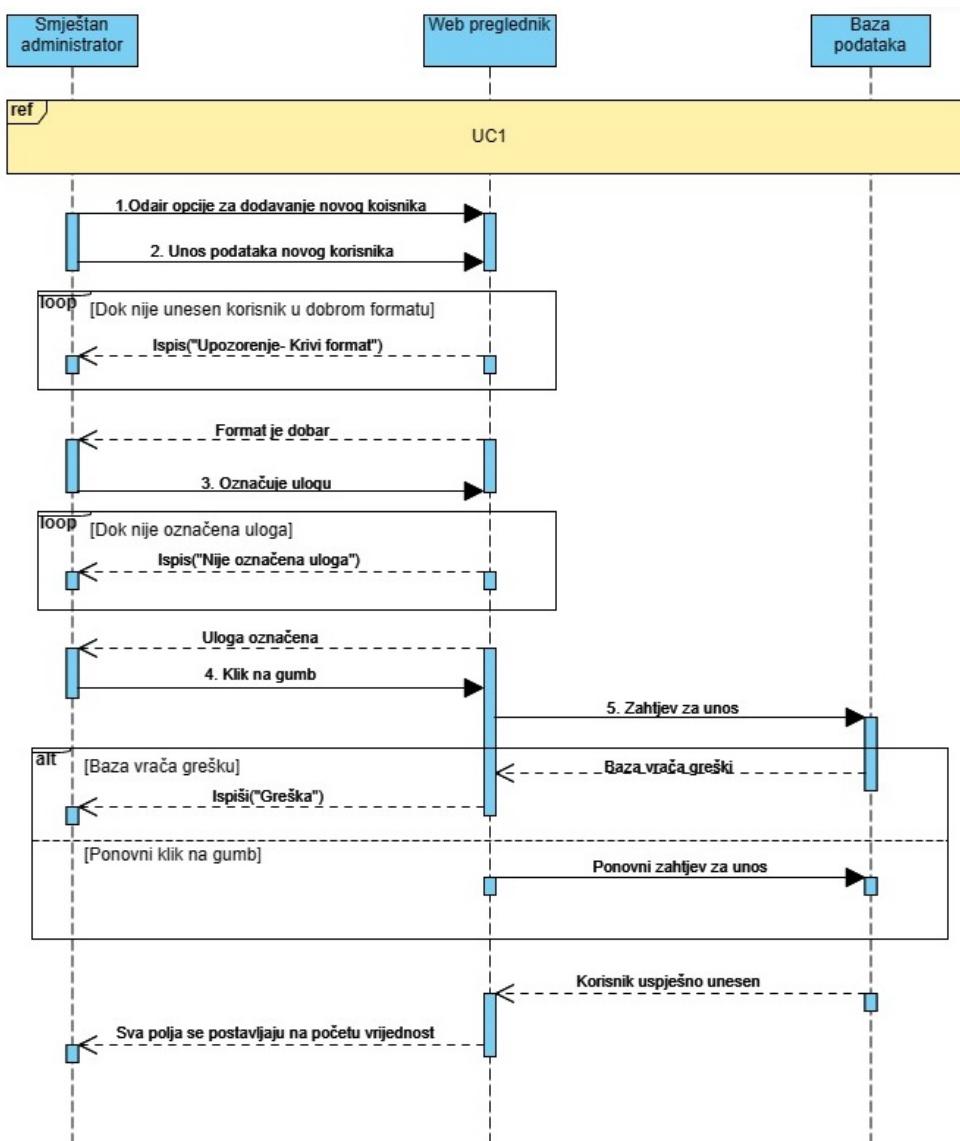
UC1-Prijava Administrator unosi korisničko ime i lozinku. Web preglednik to upućuje bazi podataka na validaciju. Ako korisnički podaci nisu ispravni ili su nepoznati, korisnikova prijava nije uspjela te se ispisuje greška. Ako je prijava uspjela korisnik je preusmjeren na početnu stranicu.



Slika 3.7: Sekvencijski dijagram UC1-Prijava

UC2-Dodavanje novog administratora

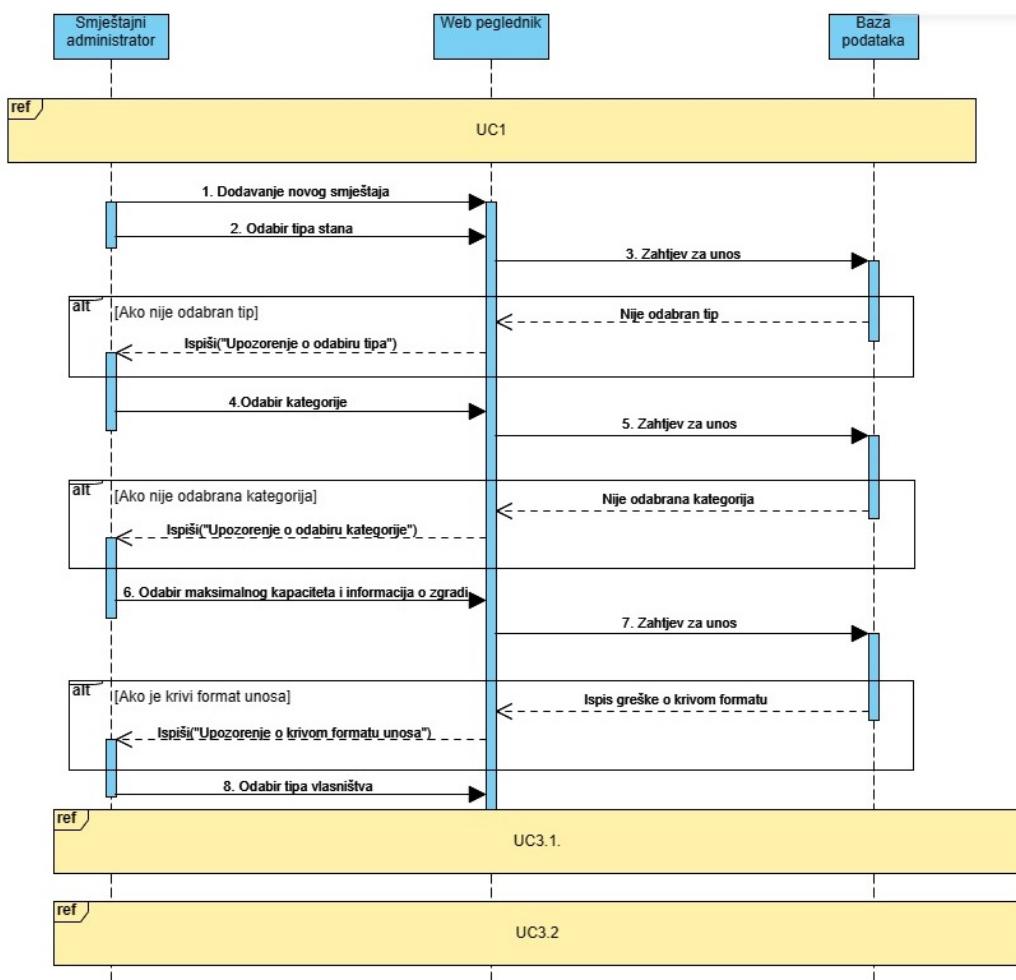
Smještajni administrator prvo se mora prijaviti (UC1-Prijava). Nakon toga odabire opciju za dodavanje novog korisnika te upisuje njegove podatke. Dok nije unesen korisnik u dobrom formatu, korisniku (smještajnom administratoru) se ispisuje upozorenje o krivom formatu. Nakon što se unese dobar format, smještajni administrator unosi ulogu. Dok nije označena uloga ispisuje se upozorenje o neoznačenoj ulozi. Nakon što je uloga označena, smještajni administrator moram kliknuti na gumb pri čemu se šalje zahtjev za unos u bazu podataka. Baza podataka može vratiti grešku te se ispisuje poruka o grešci korisniku ili se može ponovno poslati zahtjev za unos u bazu podataka. Nakon što je korisnik uspješno unesen, smještajnom administratoru se sva polja vraćaju na početnu vrijednost.



Slika 3.8: Sekvencijski dijagram UC2-Dodavanje novog administratora

UC3-Unos razpoloživog smještaja

Smještajni administrator se prvo se mora prijaviti(UC1-Prijava). Nakon toga smještajni administrator odabire opciju dodavanja novog smještaja te odabire tip stana. Web preglednik pošalje zahtjev bazi podataka za unos te baza ako nije odabran tip šalje povratnu informaciju pregledniku, a preglednik korisniku ispiše upozorenje o odabiru tipa. Ako je uspješno odabran tip , sljedeće se unosi kategorija, te ponovno šalje zahtjev za unos, ako nije odabrana kategorija ispisuje se upozorenje o odabiru kategorije. Nakon što je kategorija uspješno odabrana, smještajni administrator odabire maksimalan kapacitet i unosi informacije o zgradama,web preglednik ponovno pošalje zahtjev za unos, te baza podataka može poslati grešku nazad u slučaju ako je krivi format unosa. Na kraju smještajni administrator odabire tip vlasništva, čime se detaljnije bave UC3.1 -Smještaj u vlasništvu i UC3.2 - Smještaj u najmu.



Slika 3.9: Sekvencijski dijagram UC3-Unos raspoloživog smještaja

3.2 Ostali zahtjevi

- Aplikacija mora biti u mogućnosti grafički prikazati geografski položaj nekretnine korištenjem Google Maps/Open Maps usluge.
- Aplikacijsko sučelje mora imati mogućnost preuzimanja detalja korisnikovih tretmana iz aplikacije za evidenciju medicinskih usluga.
- Sustav treba biti jednostavan za korištenje.
- Korištenje sustava ne smije narušavati njegovu funkcionalnost i rad.
- Svi privatni podatci u sustavu moraju biti zaštićeni.
- Lozinke moraju biti sigurno spremljene u bazi.
- Nadogradnja sustava ne smije narušavati njegove postojeće funkcije.
- Sustav treba podržavati istovremeni rad više korisnika.

4. Arhitektura i dizajn sustava

- Aplikacija se sastoji od backend-a, frontend-a i baze podataka.

Backend je pozadinski dio aplikacije koji se izvršava na serveru. Za razvoj backend-a odabran je programski jezik Java i okvir Spring što čini backend prenosiv i jednostavan za korištenje. Sastoji se od rest kontrolera koji primaju HTTP zahtjeve i vraćaju JSON podatke, Service komponenti koje provjeravaju točnost upita i valjanost njihovih vrijednosti te konačno Repository komponenti koji omogućuju upisivanje u bazu i čitanje iste.

Frontend je grafičko sučelje koje omogućuje prijavu u aplikaciju, prikaz traženih podataka te lakše korištenje aplikacije. Za razvoj frontend-a odabran je programski jezik TypeScript u kojem je korišten React te React router biblioteka. Taj odabir omogućuje razvoj stranice uz pomoć tzv. komponenti što čini kod čitljivije i dopušta lakše ponovno korištenje. Za izgled stranice korišten je Bootstrap kako bi stranica imala moderan izgled.

Baza podataka pisana je u programskom jeziku SQL zbog njegove široke upotrebe i visoke kompatibilnosti.

- Ulaskom na aplikaciju korisnik (administrator) je preusmjeren na stranicu za prijavu. Nakon unosa svog korisničkog imena i lozinke, podatci se provjeravaju, i ako su ispravni, administratora se preusmjerava na administratorsku stranicu. Ondje su prikazani različiti podaci ovisno o tome koji administrator pristupa stranici, na primjer smještajni administrator vidi samo podatke vezane uz smještaj.

Sustav dodatno komunicira sa vanjskim API-em kako bi dobio podatke koje mu trebaju te ih zatim sprema u bazu podataka iz koje ih dobavlja svaki sljedeći put.

- Frontend je organiziran u jednoj mapi (*IzvorniKod/error404-fe/error404*) te dvije podmape, u mapi se nalaze paketi i dodaci potrebni za pokretanje frontend-a te podmape *src* i *public*. Unutar podmape *src* nalazi se sam kod frontend dijela aplikacije te u mapama *assets* i *components* se redom nalaze *namespace* u .svg obliku, te dodatne komponente koje omogućavaju rad frontend-a.

Backend je organiziran u jednoj mapi (*IzvorniKod/error404-be*), te dvije podmape, *src* i *docker*. U mapi *src/main* nalazi se sav kod za backend, on je podjeljen u dvije podmape. Unutar mape *java/dentall* nalazi se kod koji je pisan u programskom jeziku Javi, on je podjeljen na više podmapa u kojima se nalaze dijelovi MVC arhitekture, u mapi *dao* nalaze se Repository komponente koji služi dohvaćanju podataka iz baze, u mapi *service* nalaze se Service komponente, u mapi *rest* nalaze se kontroleri, u mapi *domain* nalaze se ostale klase koje sadrže objektne reprezentacije podataka iz baze. Unutar mape *resources* nalazi se mapa baza podataka *db* koja se sastoji od 2 podmape *sql* i *changelog*, gdje se redom nalaze sam SQL kod baze podataka te datoteke pisane u XML-u (*eXtensive Markup Language*) koje služe boljem povezivanju baze podataka sa backend-om. U mapi *docker* nalazi se *Dockerfile* koji sadrži programski kod potreban za izgradnju (*build*) same aplikacije.

4.1 Baza podataka

Baza podataka modelirana je tabličnim modelom u "Structured Query Language" (SQL) jeziku. Ona se sastoji od osam entiteta. Entiteti baze podataka su tablice u kojima su zadržani podaci potrebni za rad cijelog sustava. Tablice su redom **Korisnik**, **Smještaj**, **Adresa**, **Vozilo**, **Vozač**, **Admin**, **Administrator** i **Uloga**. Unutar baze je moguće dodavati (*insert*), mijenjati (*update*) i brisati (*delete*) podatke koje ona sadrži, također je moguće izvršavati *select* i agregatne upite koji će prikazivati traženi sadržaj koji se nalazi unutar baze podataka.

4.1.1 Opis tablica

Entitet **Korisnik** sadržava informacije o korisniku. Atributi od kojih se sastoje entitet su: IDKor koji je ujedno i identifikacijski ključ korisnika, Ime, Prezime, DatDol odnosno vrijeme dolaska, DatOdl vrijeme odlaska, IdSmj, RegVoz i OdvoziRez. Ovaj entitet je u vezi One-to-One s entitetom Smještaj preko atributa IDSmj, te je u vezi One-to-One s entitetom Vozilo preko atributa RegVoz i OdlaziRegVoz.

Korisnik		
IDKor	INT	Identifikacijski ključ korinika
Ime	VARCHAR	Ime korisnika
Prezime	VARCHAR	Prezime korisnika
DatDol	TIMESTAMP	Datum dolaska korisnika
DatOdl	TIMESTAMP	Datum odlaska korisnika
IdSmj	INT	Identifikacijski ključ smještaja
RegVozila	VARCHAR	Identifikacijski ključ vozila u dolasku
OdvoziReg	VARCHAR	Identifikacijski ključ vozila u odlasku

Entitet **Smještaj** opisuje smještaj u kojem će boraviti korisik. Sadrži atribute: IDSmj (ujedno i identifikacijski ključ), vrsta smještaja te IDAdr. Entitet Smještaj povezan je sa vezom Many-to-Many s entitetom Adresa preko atributa IDAdr.

Smještaj		
IDSmj	INT	Identifikacijski ključ smještaja
Vrsta	VARCHAR	Vrsta smještaja
IDAdr	INT	Identifikacijski ključ adrese smještaja

Entitet **Adresa** govori o samoj adresi smještaja i to s atributima: IDAdr, Mjesto, Ulica i Broj. Identifikacijski ključ ovog entiteta je IDAdr.

Adresa		
IDAdr	INT	Identifikacijski ključ adrese
Mjesto	VARCHAR	Mjesto u adresi smještaja
Ulica	VARCHAR	Ulica u adresi smještaja
Broj	INT	Kućni broj u adresi smještaja

Entitet **Vozilo** sadrži informacije o samom vozilu. Sadrži atribute: RegVozila (registracija vozila), Model i Boja. Identifikacijski ključ entiteta je registracija vozila (RegVoz)

Vozilo		
RegVozila	VARCHAR	Identifikacijski ključ vozila
Model	VARCHAR	Model vozila
Boja	VARCHAR	Boja vozila

Entitet **Vozač** opisuje vozača koji vozi i odvozi korisnika u i iz njegovog privremenog smještaja. Sadrži atribute: IDVoz (identifikacijski ključ), Ime, Prezime, Brradsat(broj radnih sati) te RegVozila. Povezan je s entitetom Vozilo sa vezom Many-to-Many preko atributa RegVozila.

Vozač		
IDVoz	INT	Identifikacijski ključ vozača
Ime	VARCHAR	Ime vozača
Prezime	VARCHAR	Prezime vozača
Brradsat	INT	Broj radnih sati vozača
RegVozila	VARCHAR	Identifikacijski ključ vozila

Entitet **Admin** sadrži informacije o adminu profilu te se sastoji od atributa: UserName i lozinka.

Admin		
UserName	VARCHAR	Identifikacijski ključ admina
Lozinka	VARCHAR	Lozinka admina

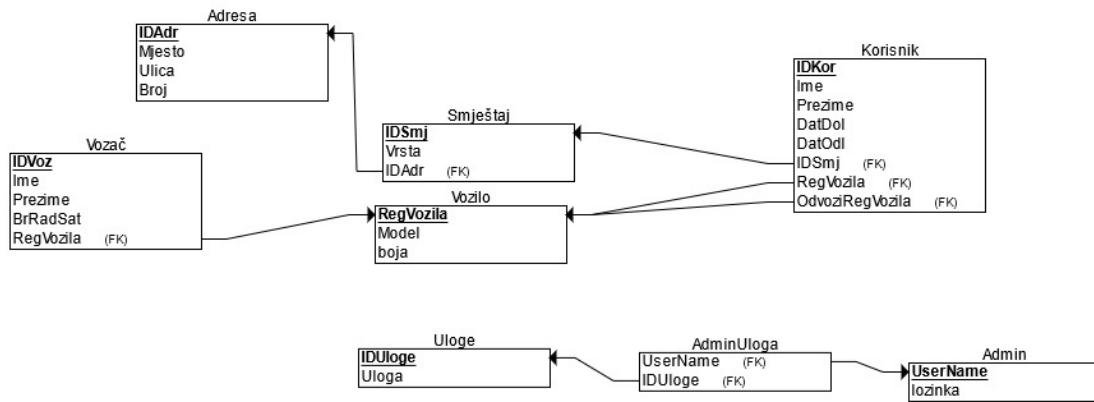
Entitet **AdminUloga** povezuje admina s njegovom ulogom. Sastoji se od atributa: UserName i IDUloge. Povezan je s entitetom Admin s vezom Many-to-Many te preko atributa UserName, te je spojen također s vezom Many-to-Many s entitetom Uloga preko atributa IDUloge.

AdminUloga		
UserName	VARCHAR	Obrisničko ime admina
IDUloge	VARCHAR	ID Uloge admina

Entitet **Uloge** sadrži informacije o samoj ulozi. Aributi od kojih se sastoje su: IDUloge(identifikacijski ključ) te Uloga

Uloge		
IDUloge	VARCHAR	Identifikacijski ključ uloge admina
Uloga	VARCHAR	Uloga admina

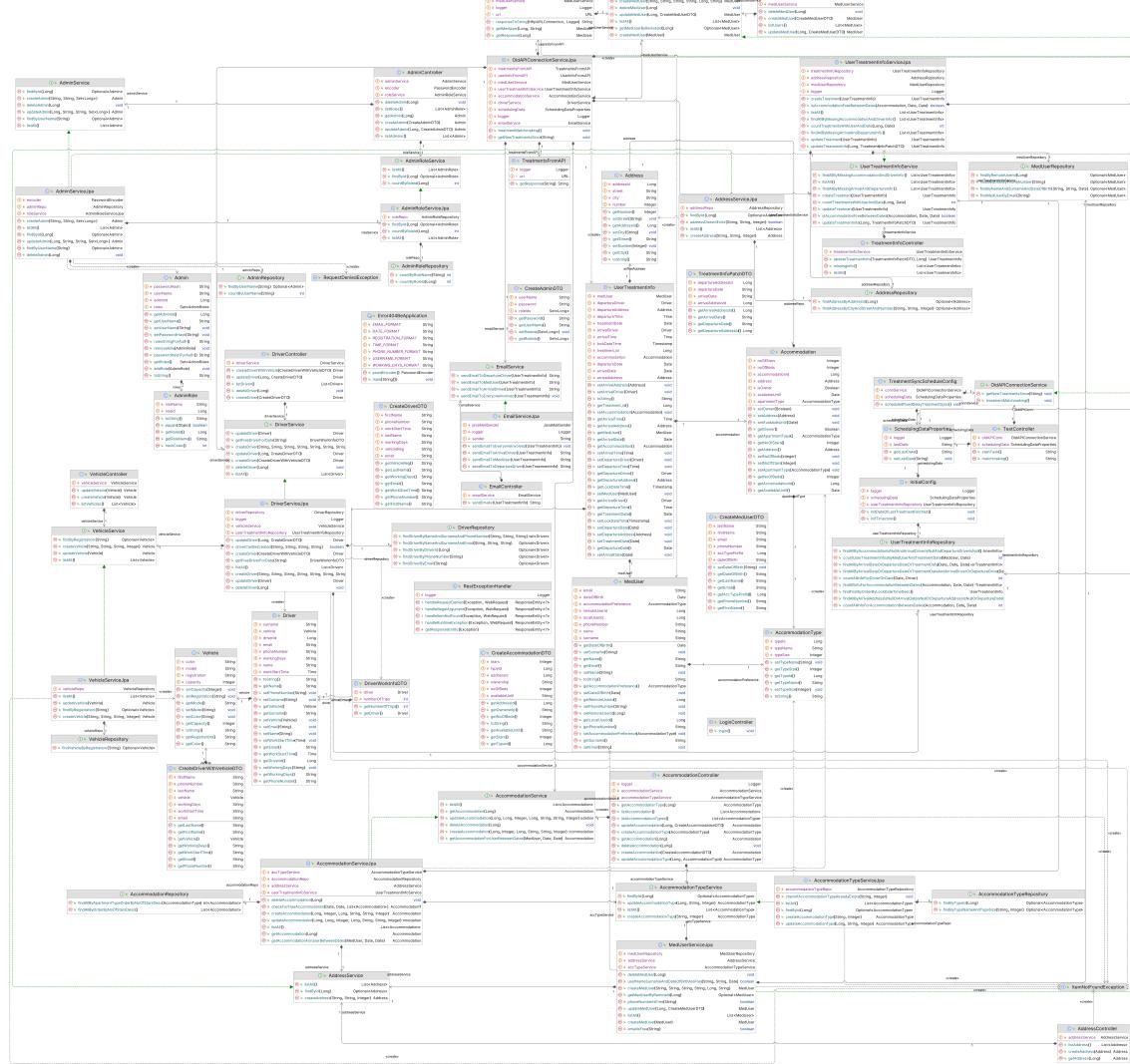
4.1.2 Dijagram baze podataka



Slika 4.1: Dijagram baze podataka

4.2 Dijagram razreda

Na sljedećim slikama prikazani su dijagrami razreda koji se odnose na *backend* dio aplikacije. Na slici 4.2 prikazan je cjelokupni dijagram razreda, a na ostalima su razdvojeni u smislene cjeline. Za razvoj aplikacije korišten je MVC arhitektturni obrazac u *Springu*. Struktura ovoga arhitektturnog obrasca sastoji se od 3 glavna dijela: *Model* (ostvaren u slojevima *Service* i *Repository - dao*, model podataka, izvršavanje upita nad bazom), *Pogled* (klijentska strana, komunicira sa *Springom* putem HTTP zahtjeva i odgovora) te *Nadglednik* (ostvaren u sloju *Controller* - u našoj strukturi aplikacije direktorij naziva *rest*, zadužen za primanje i provjeravanje zahtjeva). U nastavku bit će pojašnjeni detaljnije svaki od njih.



Slika 4.2: Cjelokupan dijagram razreda

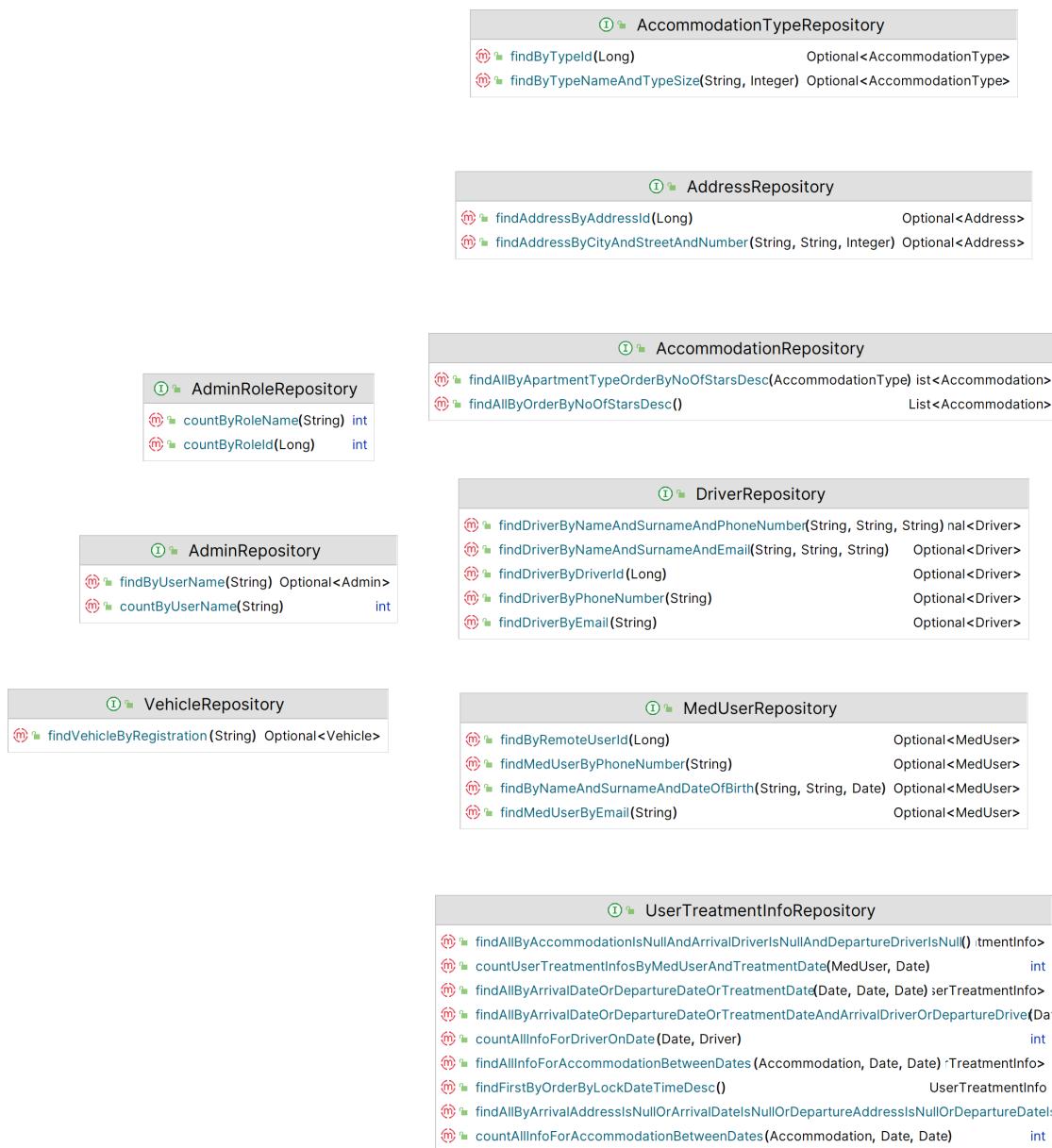


Slika 4.3: Dijagram razreda - Controller

Navedene klase nasljeđuju REST Controller koji je zadužen za rukovanje HTTP zahtjevima i za pružanje odgovarajućih odgovora. REST Controller vraća podatke u JSON formatu.

DTO objekti služe za transfer podataka između slojeva aplikacije, pogotovo između klijenta i servera. CreateAdminDTO je *Data transfer Object* koji je zadužen za stvaranje administratora, CreateDriverDTO je *Data transfer Object* koji je zadužen za stvaranje vozača itd.

Klase unutar *security* poddirektorija služe za baratanje HTTP zahtjevima te vraćanje odgovarajućih odgovora.

Slika 4.4: Dijagram razreda - *Repository*

Navedena sučelja nasljeđuju *JPARepository* koji pruža generičke metode za operacije s podatcima, poput spremanja, ažuriranja, brisanja i slično.

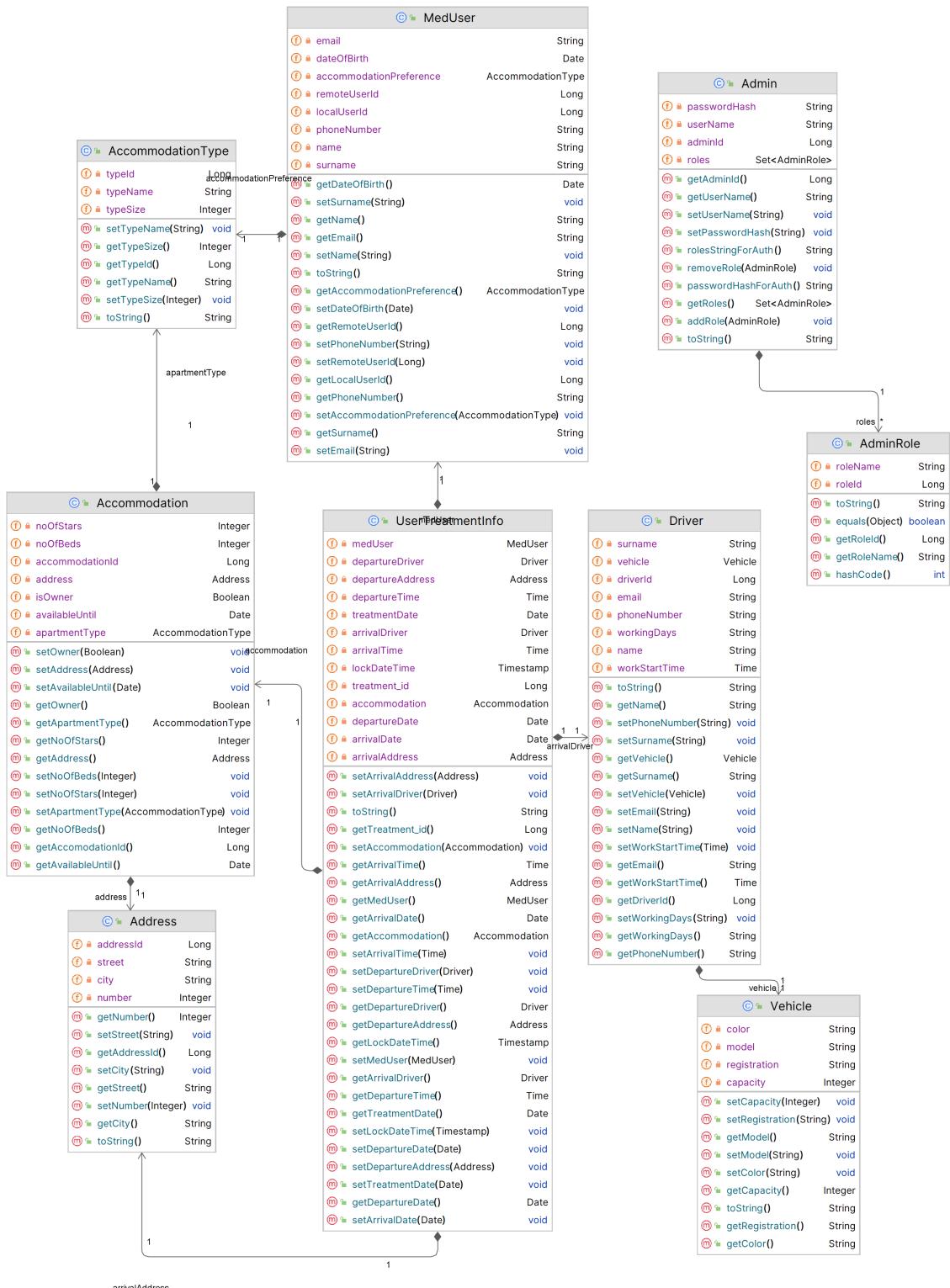


Slika 4.5: Dijagram razreda - *Service*

Navedena Service sučelja sadrže poslovnu logiku aplikacije, tj. odgovorni su za obradu podataka, implementaciju algoritama itd. Ponašaju se kao sloj između *Controllera* i *Repositoryja*.

AdminService sadrži metodu *createAdmin* za stvaranje administratora. Postoji provjera raznih svojstava, poput duljine lozinke, duljine nadimka, postoji li već neki administrator s tim nadimkom, postojanje navedenih uloga itd. Ako je sve uspješno spremi se novi administrator u *AdminRepository*. Slične funkcionalnosti nalaze se i u ostalim klasama.

Implementacije svih sučelja nazale se unutar poddirektorija *impl*

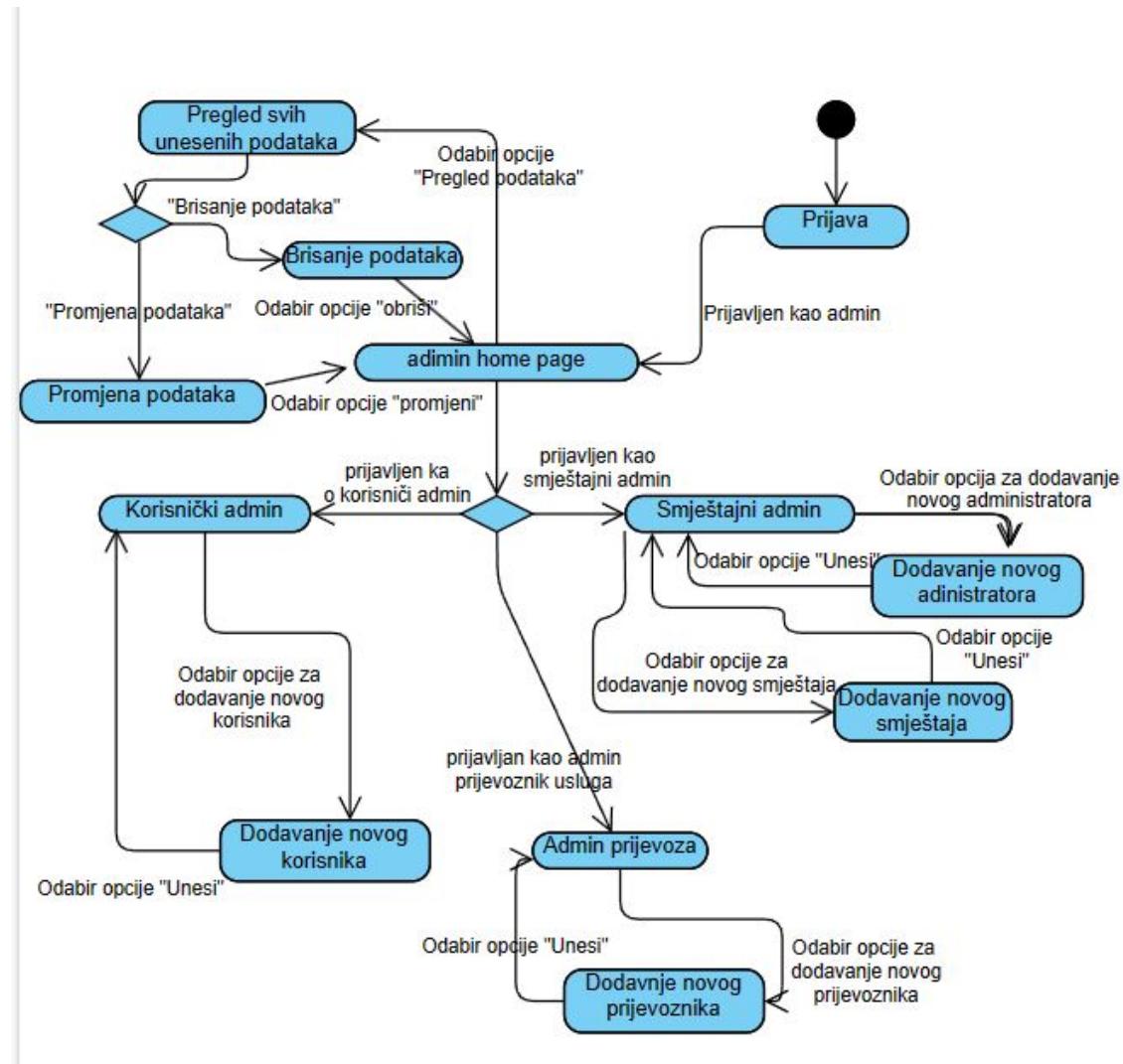
Slika 4.6: Dijagram razreda - *Domain*

Modeli predstavljaju strukturu baze podataka u našoj aplikaciji. Tako imamo klase: *Admin*, *MedUser*, *Accommodation*, *Driver*, *Vehicle* itd. sa svojim privatnim atributima te javnim metodama.

dama. Tako na primjer *Admin* ima svoj ID, nadimak, ime te pripadajuće uloge, koje mogu biti smještajni administrator(ima najveće ovlasti), korisnički administrator te prijevozni administrator, što je sadržano u *AdminRole*. *Accommodation* sadrži sve podatke vezane uz smještaj te vrstu smještaja, što je sadržano u *AccommodationType* i adresu smještaja (zasebna klasa *Address*), a *Driver* sve potrebno za definiranje usluge prijevoza, odnosno osobni podaci vozača te vrstu vozila koju koristi, što možemo vidjeti unutar *Vehicle* klase. *MedUser* sadrži sve potrebno za definiranje korisnika medicinskih usluga, njihove osobne podatke, vrijeme dolaska i odlaska, preferencije vezane uz smještaj i slično.

4.3 Dijagram stanja

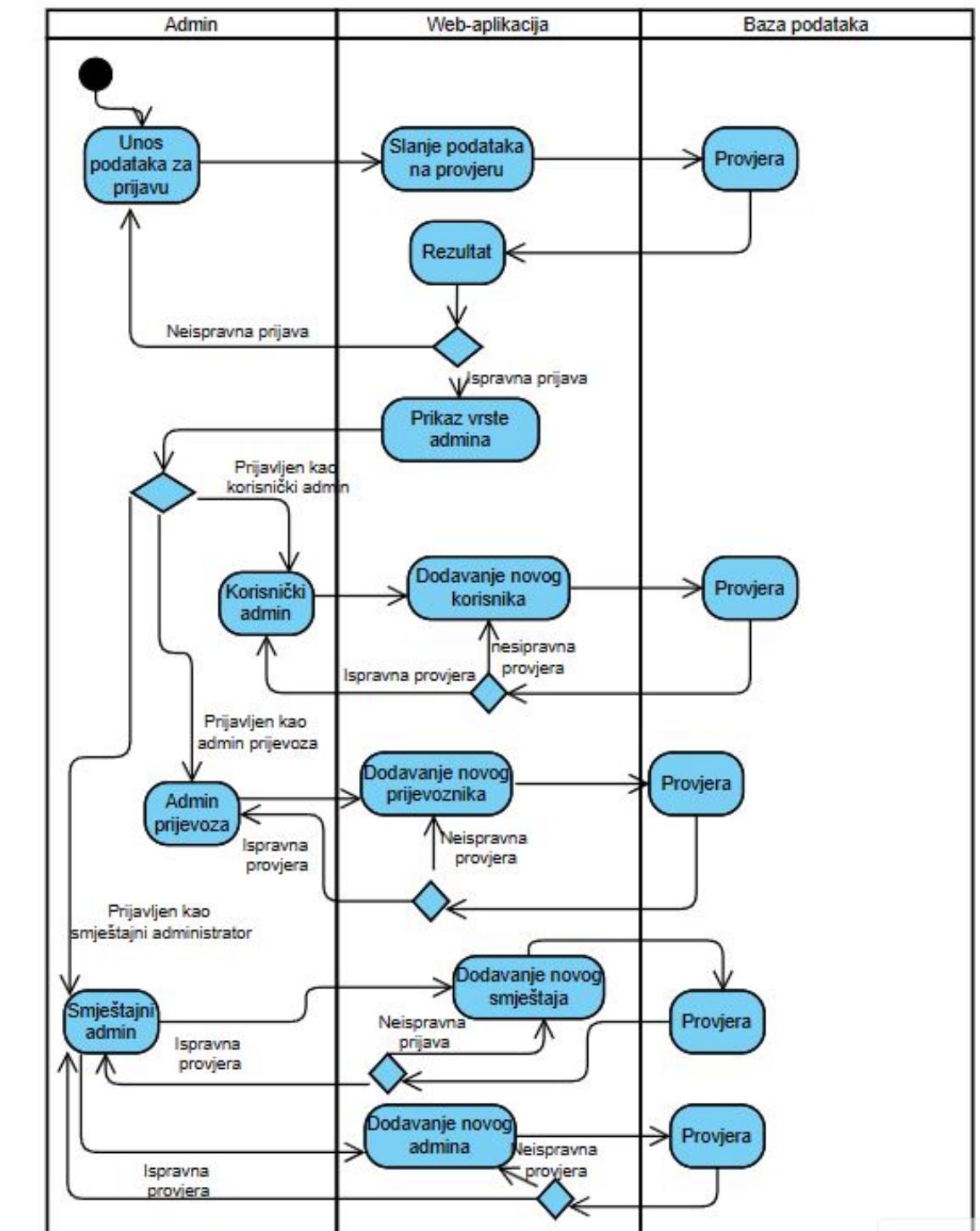
Na slici 4.7 prikazan je dijagram stanja. Prvo na što admin nađe je prijava, te nakon toga mu se prikaže web stranica za admina. Na toj stranici može odabratи opciju pregleda svih unesenih podataka te izbrisati ili promjeniti iste. Ako je admin prijavljen kao korisnički admin onda on odabirom opcije za dodavanje novog korisnika može dodati novog korisnika. Dok ako je admin prijavljen kao admin prijevoza, može dodavati nove prijevoznike. Smještajni admin može odabirom opcije za dodavanje novog administratora, dodati novog administratora, ali isto tako odabirom opcije za dodavanje novog smještaja, može dodati novi smještaj.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

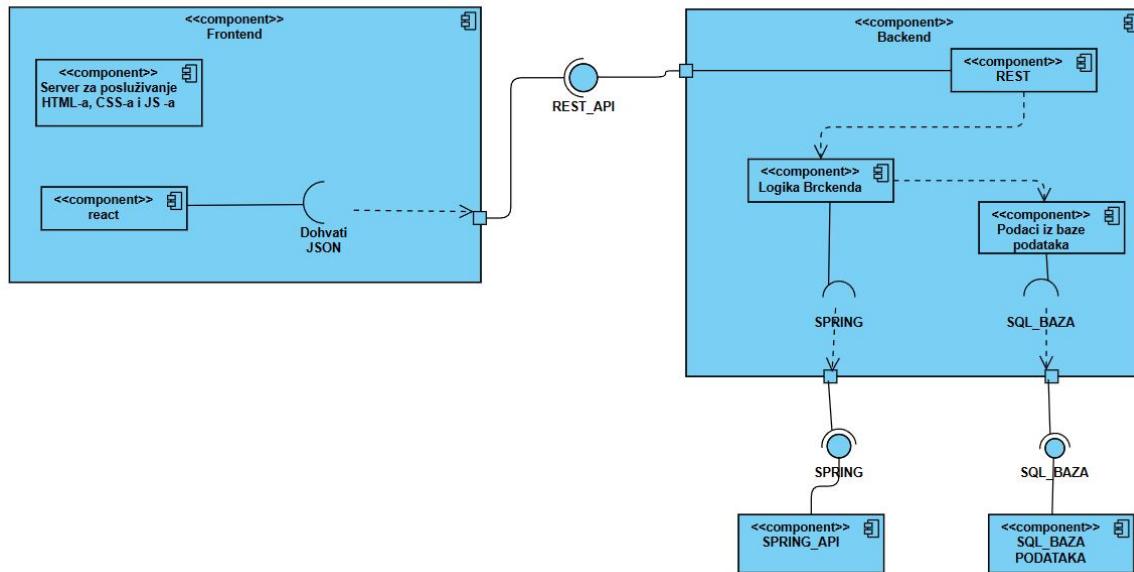
Na slici 4.8 prikaza je dijagram aktivnosti. Sve počinje prijavom. Uneseno korisničko ime i lozinka se provjeravaju s bazom podataka. Ako je neispravna prijava, admin se vraća na obrazac za unos podataka za prijavu, dok ako je prijava uspješna, pojavljuje se prikaz vrsta admina. Ako je prijavljen korisnički admin, on može dodavati nove korisnike. Adim prijevoza može dodavati nove prijevoznike. Dok smještajni admin može dodavati novi smještaj te dodavati nove admire.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Na slici 4.9 prikazan je dijagram komponenti koji pokazuje međuvisnosti između frontenda, backenda i baze podataka. Frontend ima poseban server za HTML, CSS i JS datoteke koje služe za strukturu i dizajn web stranice. REST API komponenti pristupa se preko sučelja za dohvat JSON podataka te poslužuje podatke backendu. Cijeli backend je napravljen na Springu te backend komunicira s bazom podataka slanjem SQL upita.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Tehnologije i alati koji su se koristili u izradi aplikacije grupirane su s obzirom na mjesto primjene: backend, frontend, baza podataka, dokumentacija i ispitivanje. Za komunikaciju među članovima korištena je aplikacija WhatsApp koju održava tvrtka Meta, te platformu Microsoft Teams, koja se također koristila za komunikaciju s asistentom i demonstratorom, a koju održava Microsoft. Za upravljanje izvornog koda korišten je alat Git koji je otvorenog koda na platformi GitHub kojeg razvija tvrtka istoga imena, a čija je roditeljska tvrtka Microsoft, i konačno za puštanje aplikacije u pogon korišten je poslužitelj Render.

Backend

Korišteno je radno okruženje IntelliJ IDEA koje je razvila tvrtka JetBrains za koji postoji komercijalna, Apache 2 licansa, a radno okruženje se koristilo preko fakultetskog računa. Za provjeru funkcionalnosti prilikom izrade koristila se aplikacija Postman koja ima mogućnost slanja HTTP zahtjeva backend-u. Za razvoj su korištene tehnologije Spring i programski jezik Java.

Frontend

Korišteno je radno okruženje VSC (Visual Studio Code) kojeg je razvila tvrtka Microsoft. Radno okruženje je služilo pisanju koda. Nadalje korišteni su preglednici Safari kojeg je razvila tvrtka Apple, Firefox kojeg je razvila tvrtka Mozilla te Chrome kojeg je razvila tvrtka Google, preglednici su korišteni u svrhu provjere izgleda frontend-a. Za sam razvoj su korištene biblioteke tehnologije React te programski jezik TypeScript, a za izgled tehnologija Bootstrap

Baza podataka

Baza podataka je napisana u SQL jeziku, u tu svrhu poslužio je upravljački sustav PostgreSQL te alat naziva, pgAdmin, oba su otvorena koda. Dijagram baze podataka napravljen je pomoću online alata EDRPlus koji služi za modeliranje dijagrama baze podataka.

Dokumentacija

Dokumentacija je pisana pomuću okruženja TeXstudio, LaTex jezikom, dok su UML dijagrami izrađeni dijelom programom Astah kojeg je razvila tvrtka Change Vision, a dijelom online okolinom Visual Paradigm kojeg je razvila tvrtka istoga imena.

Ispitivanje

Za ispitivanje komponenti korišten je alat JUnit 5 u okolini Spring Boot, te pisan u programskom jeziku Java. Ispitivanje sustava provedeno je pomoću dodataka za preglednik Selenium IDE koji je otvorenog koda i preglednika Firefox.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti ostvareno je pomoću JUnit 5 alata za ispitivanje u okolini Spring Boot, a pisan u programskom jeziku Java.

Na donjoj slici prikazan je izvorni kod klase *AdminTest*, ona provjerava stvara li sustav objekt *Admin* sa oba konstruktora te može li se pristupiti tim podacima. Test **getAdminInfoDef()** testira inicijalizaciju objekta *Admin* kojemu su u slučaju ne postavljanja argumenata sve vrijednosti jednake **NULL**. Zatim test **getAdminInfoTest** testira kreiranje objekta *Admin* sa nekim danim vrijednostima, u testnom primjeru je to (*username = "tomi"*) te *passwordHash = "password"*.

```
class AdminTest {
    Admin adminDef;
    Admin adminTest;

    @BeforeEach
    @DisplayName("Setting up info")
    void setUp() {
        adminDef = new Admin();
        adminTest = new Admin("tomi", "password");
    }

    @Test
    @DisplayName("Testing w default info")
    void getAdminInfoDef() {
        assertNull(adminDef.getUserName(), "Username should be null for default constructor");
        assertNull(adminDef.passwordHashForAuth(), "Passwordhash should be null for default constructor");
    }

    @Test
    @DisplayName("Testing w testing info")
    void getAdminInfoTest() {
        assertEquals("tomi", adminTest.getUserName(), "Getter should return correct username");
        assertEquals("password", adminTest.passwordHashForAuth(), "Getter should return correct passwordhash");
    }
}
```

Slika 5.1: Kod klase AdminTest

Nadalje, provedeno je ispitivanje klase *Accommodation* koja obrađuje informacije o smještaju u sustavu. Na donjoj slici prikazan je izvorni kod klase *AccommodationTest* koji se sastoji od funkcije **setUp** koji stvara potrebne informacije, testa **testGetDefault** koji ispituje inicijalizaciju te provjerava njezinu ispravnost, inicijalna vrijednost za svaki atribut treba biti **NULL**. Test **testGetCorrect** testira stvaranje objekta kada su mu zadane neke vrijednosti, npr. adresa je *address = Unska ulica 23, Zagreb*, tip smještaja je *accommodationType = stan*, broj zvjezdica je *noOfStars = 3* i broj kreveta *noOfBeds = 5* te povrat i spremanje danih vrijednosti.

```

class AccommodationTest {
    Address adresa;
    Date datum;
    AccommodationType accommodationType;
    Accommodation accommodationEmpty;
    Accommodation accommodationTest;

    @BeforeEach
    @DisplayName("Setting up info")
    void setUp() {
        accommodationEmpty = new Accommodation();
        adresa = new Address("Zagreb", "Unska Ulica", 23);
        datum = new Date(0);
        accommodationType = new AccommodationType("stan", 200);
        accommodationTest = new Accommodation(accommodationType, 3, adresa, true, datum, 5);
    }

    @Test
    @DisplayName("Getting default apartment info")
    void testGetDefault() {
        assertNull(accommodationEmpty.getApartmentType(), "Default apartment type should be null");
        assertNull(accommodationEmpty.getAddress(), "Default address should be null");
        assertNull(accommodationEmpty.getAvailableUntil(), "Default available date should be null");
        assertNull(accommodationEmpty.getNoOfStars(), "Default number of stars should be null");
        assertNull(accommodationEmpty.getNoOfBeds(), "Default number of beds should be null");
        assertNull(accommodationEmpty.getOwner(), "Default owner should be null");
    }

    @Test
    @DisplayName("Getting correct apartment info")
    void testGetCorrect() {
        assertEquals(accommodationType, accommodationTest.getApartmentType(), "Getter should return correct type.");
        assertEquals(adresa, accommodationTest.getAddress(), "Getter should return correct address");
        assertEquals(datum, accommodationTest.getAvailableUntil(), "Getter should return correct date");
        assertEquals(3, accommodationTest.getNoOfStars(), "Getter should return correct number of stars");
        assertEquals(5, accommodationTest.getNoOfBeds(), "Getter should return correct number of beds");
        assertEquals(true, accommodationTest.getOwner(), "Getter should return correct owner");
    }
}

```

Slika 5.2: Kod klase AccommodationTest

Za kraj provedeno je ispitivanje klase *UserTreatmentInfo*, na isti način kao i prethodne dvije klase, testiran je konstruktor bez zadanih vrijednosti i sa zadanim vrijednostima, jedina razlika je u tome da ova klasa ima dva konstruktora koji traže vrijednosti. Prvo inicijaliziramo tri instance klase **UserTreatmentInfo**, jedno bez zadanih vrijednosti, drugu u koju upisujemo vrijednosti: *treatment_id* = 123L, zatim za datum dolaska i odlaska isti, onaj koji jest u trenutku pokretanja, dalje za adresu dolaska i odlaska istu vrijednost *address* = "Zagreb, Savska, 132", istoga vozača na odlaku i povratku (čije vrijednosti su *name*="Marko", *surname*="Popić", *email*="mail@mail.com", *phoneNumber*="0992583695", vozilo crveni *Porcshe*, vrijeme početka rada trenutno, a za dane rada *string*="NUCP"), smještaj s istim vrijednostima kao i u ispitivanju klase *Accommodation*, datum usluge isti kao već spomenuti, trenutni (onaj u trenutku pokretanja testa) *Timestamp* za trenutak zaključavanja i korisnik sa podacima *name*="Luke", *surname*="Skywalker", *email*="lukeskywalker@force.com", *phoneNumber* = "036958721" već spomenuti *accommodationType* te datum rođenja jednak onom pri pokretanju što se sprema u varjablu **userInfoTestOne**. Treći konstruktor traži samo tri vrijednosti datum liječenja što je u testnom primjeru postavljeno na jednak onom na dan testiranja, trenutak zaključavanja koji je postavljen na jednak način i korisnika usluge, on je postavljen jednakom kao i u prethodnom konstruktoru, a sve je spremljeno u varijablu **userInfoTestTwo**. Inicijalizaciju podataka pokazuje slika dolje.

```

class UserTreatmentInfoTest {
    UserTreatmentInfo userInfoDef;
    UserTreatmentInfo userInfoTestOne;
    UserTreatmentInfo userInfoTestToo;
    UserTreatmentInfo userInfoFault;
    Date date;
    Address adresa;
    Driver vozac;
    Accommodation accomodation;
    Timestamp trenutakZakljuc;
    MedUser korisnik;

    @BeforeEach
    @DisplayName("Setting up necesery info")
    void setUp() {
        userInfoDef = new UserTreatmentInfo();
        date = new Date();
        adresa = new Adresa("Zagreb", "Savska", 132);
        vrijeme = new Time();
        vozilo = new Vehicle("red", "Porsche 911", "red", 2);
        vozac = new Driver("Marko", "Popić", "mail@mail.com", "0992583695", vozilo, vrijeme, "NUCP");
        accommodationType = new AccommodationType(stan, 200);
        accomodation = new Accommodation(accommodationType, 3, adresa, true, datum, 5);
        korisnik = new MedUser("Luke", "skywalker", "lukeskywalker@force.com", "036958721", accomodation, date);
        trenutakZakljuc = new Timestamp();
        userInfoTestOne = new UserTreatmentInfo(123, date, date, adresa, adresa, vozac, vozac, accomodation, date, trenutakZakljuc, korisnik);
        userInfoTestToo = new UserTreatmentInfo(date, trenutakZakljuc, korisnik);
        userInfoFault = new UserTreatmentInfo();
    }
}

```

Slika 5.3: Kod klase UserTreatmentTest, funkcija **setUp**

Konstruktor bez zadanih vrijednosti treba za svaki atribut postaviti NULL što provjerava test **getUserInfoDef**. Prvi konstruktor treba postaviti vrijednosti oblika varijable **userInfoTestOne** što provjerava test **getUserInfoOne** što je prikazano na slici dolje.

```

@Test
void getUserInfoDef() {
    assertNull(userInfoDef.getArrivalDate(), "Should be null");
    assertNull(userInfoDef.getDepartureDate(), "Should be null");
    assertNull(userInfoDef.getArrivalAddress(), "Should be null");
    assertNull(userInfoDef.getDepartureAddress(), "Should be null");
    assertNull(userInfoDef.getArrivalDriver(), "Should be null");
    assertNull(userInfoDef.getDepartureDriver(), "Should be null");
    assertNull(userInfoDef.getAccommodation(), "Should be null");
    assertNull(userInfoDef.getArrivalTime(), "Should be null");
    assertNull(userInfoDef.getDepartureTime(), "Should be null");
    assertNull(userInfoDef.getTreatmentDate(), "Should be null");
    assertNull(userInfoDef.getLockDateTime(), "Should be null");
    assertNull(userInfoDef.getMedUser(), "Should be null");
}

@Test
void getUserInfoOne() {
    assertEquals(date, userInfoTestOne.getArrivalDate(), "Getter should return correct info");
    assertEquals(date, userInfoTestOne.getDepartureDate(), "Getter should return correct info");
    assertEquals(adresa, userInfoTestOne.getArrivalAddress(), "Getter should return correct info");
    assertEquals(adresa, userInfoTestOne.getDepartureAddress(), "Getter should return correct info");
    assertEquals(vozac, userInfoTestOne.getArrivalDriver(), "Getter should return correct info");
    assertEquals(vozac, userInfoTestOne.getDepartureDriver(), "Getter should return correct info");
    assertEquals(accomodation, userInfoTestOne.getAccommodation(), "Getter should return correct info");
    assertEquals(date, userInfoTestOne.getTreatmentDate(), "Getter should return correct info");
    assertEquals(date, userInfoTestOne.getLockDateTime(), "Getter should return correct info");
    assertEquals(korisnik, userInfoTestOne.getMedUser(), "Getter should return correct info");
    assertNull(userInfoDef.getArrivalTime(), "Should be null");
    assertNull(userInfoDef.getDepartureTime(), "Should be null");
}

```

Slika 5.4: Kod klase UserTreatmentTest, test **getUserInfoDef** i test **getUserInfoOne**

Posljednji konstruktor prima vrijednosti oblika varijable **userInfoTestToo**, a ispravnost postavljanja provjerava test **getUserInfoToo** čiji je izvorni kod prikazan na slici dolje.

```

@Test
void getUserInfoToo() {
    assertNull(userInfoTestToo.getArrivalDate(),"Should be null");
    assertNull(userInfoTestToo.getDepartureDate(), "Should be null");
    assertNull(userInfoTestToo.getArrivalAddress(),"Should be null");
    assertNull(userInfoTestToo.getDepartureAddress(),"Should be null");
    assertNull(userInfoTestToo.getArrivalDriver(),"Should be null");
    assertNull(userInfoTestToo.getDepartureDriver(),"Should be null");
    assertNull(userInfoTestToo.getAccommodation(),"Should be null");
    assertNull(userInfoTestToo.getArrivalTime(),"Should be null");
    assertNull(userInfoTestToo.getDepartureTime(),"Should be null");
    assertNull(userInfoTestToo.getArrivalTime(),"Should be null");
    assertNull(userInfoTestToo.getDepartureTime(),"Should be null");
    assertEquals(date, userInfoTestToo.getTreatmentDate(), "Getter should return correct info");
    assertEquals(date, userInfoTestToo.getTreatmentDate(), "Getter should return correct info");
    assertEquals(korisnik, userInfoTestToo.getMedUser(), "Getter should return correct info");
}

```

Slika 5.5: Kod klase UserTreatmentTest, test **getUserInfoToo**

5.2.2 Ispitivanje sustava

Zbog broja ne implementiranih funkcija na sustavu testiranje je ograničeno.

Provjerava se dodavanje novog korisnika (*user*), vozača (*driver*), administratora (*admin*) te smještaja (*accommodation*). Za početak, dodavanje novog korisnika nije moguće preko tipke koja je za tu funkciju predviđena. Očekivani rezultat bi bio prikaz dodanog korisnika, ali korisnik se ne dodaje te ne prikazuje.

1	✓ open	/	
2	✓ set window size	778x821	
3	✓ type	id=loginUsername	owner
4	✓ type	id=loginPassword	pass
5	✓ click	css= btn	
6	✓ click	css= btn-secondary	
7	✓ click	linkText=Transportation	

Slika 5.6: Snimka kako bi dodavanje novog korisnika trebalo izgledati

Running 'Adding a new user'

1. open on / **OK**
2. setWindowSize on 778x821 **OK**
3. type on id=loginUsername with value owner **OK**
4. type on id=loginPassword with value pass **OK**
5. click on css=.btn **OK**
6. Trying to find css=.btn-secondary... **OK**
7. click on linkText=Transportation **OK**

'Adding a new user' completed successfully

Slika 5.7: Rezultati testa

Nadalje, klik na gumb za koji je predviđeno dodavanje novog vozača otvara formu predviđenu za to. Nakon upisa podataka i klika na gumb za dodati novog vozača, novi vozač bi se trebao prikazati u listi, ali to se ne događa.

1	✓ open	/
2	✓ set window size	584x816
3	✓ type	id=loginUsername
4	✓ type	id=loginPassword
5	✓ click	css=.btn
6	✓ click	linkText=Transportation
7	✓ mouse over	css=.btn-secondary
8	✓ click	css=.btn-secondary
9	✓ click	name=name
10	✓ type	name=name
11	✓ click	name=surname
12	✓ type	name=surname
13	✓ click	name=email

Slika 5.8: Snimka kako bi dodavanje novog vozača trebalo izgledati

24. type on name=vehicle.registration with value SB1234Ja **OK**
25. click on name=vehicle.model **OK**
26. type on name=vehicle.model with value Opel Astra **OK**
27. click on name=vehicle.color **OK**
28. type on name=vehicle.color with value grey **OK**
29. click on name=vehicle.capacity **OK**
30. type on name=vehicle.capacity with value 5 **OK**
31. click on css=.btn-primary **OK**
32. Trying to find css=tr:nth-child(7).btn-outline-primary... **OK**

'Adding a new driver' completed successfully

Slika 5.9: Rezultati testa

Treći test je dodavanje novog administratora, pri pritsku na tipku koja je za to predviđena otvara se forma koja omogućuje upis novog administratora, ali on se ne prikazuje, kao ni jedan upisani administrator u sustavu.

1	✓ open	/
2	✓ set window size	1120x816
3	✓ type	id=username
4	✓ type	id=password
5	✓ click	css=.btn
6	✓ click	linkText=Accommodation
7	✓ click	css=.btn:nth-child(2)
8	✓ double click	name=username
9	✓ type	name=username
10	✓ click	name=password
11	✓ click	name=roleIds
12	✓ type	name=roleIds
13	✓ click	css=.btn:nth-child(4)

Slika 5.10: Snimka kako bi dodavanje novog korisnika trebalo izgledati

Running 'Adding a new user'

1. open on / **OK**
2. setWindowSize on 778x821 **OK**
3. type on id=loginUsername with value owner **OK**
4. type on id=loginPassword with value pass **OK**
5. click on css=.btn **OK**
6. Trying to find css=.btn-secondary... **OK**
7. click on linkText=Transportation **OK**

'Adding a new user' completed successfully

Slika 5.11: Rezultati testa

Posljednji test je dodavanje novog smještaja, pri pritisku na tipku koja je zato predviđena otvara se forma koja omogućuje upis novog smještaja, ali on se ne prikazuje s ostalim, već upisanim smještajima.

1	✓ open	/	
2	✓ set window size	584x816	
3	✓ type	id=loginUsername	owner
4	✓ type	id=loginPassword	pass
5	✓ click	css=.btn	
6	✓ click	linkText=Accommodation	
7	✓ click	css=.btn:nth-child(6)	
8	✓ click	name=type.typeName	
9	✓ type	name=type.typeName	stan
10	✓ click	name=address.city	
11	✓ type	name=address.city	Zagreb
12	✓ click	name=address.street	
13	✓ type	name=address.street	Vlaška
14	✓ click	name=address.number	

Slika 5.12: Snimka kako bi dodavanje novog smještaja trebalo izgledati

8. click on name=type.typeName **OK**
9. type on name=type.typeName with value stan **OK**
10. click on name=address.city **OK**
11. type on name=address.city with value Zagreb **OK**
12. click on name=address.street **OK**
13. type on name=address.street with value Vlaška **OK**
14. click on name=address.number **OK**
15. type on name=address.number with value 11 **OK**
16. click on name=noOfBeds **OK**
17. type on name=noOfBeds with value 5 **OK**
18. click on css=.btn:nth-child(3) **OK**

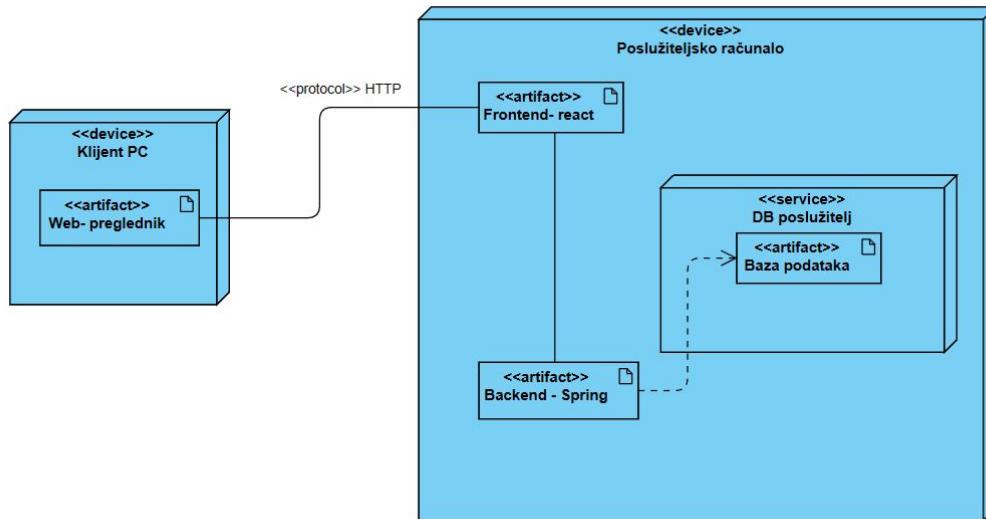
'Adding a new accommodation' completed successfully

Slika 5.13: Rezultati testa

Testiranje sustava pokazuje da on nije u mogućnosti ostvariti komunikaciju između frontenda i backenda, ali da je u mogućnosti prikazati prethodno upisane vrijednosti u bazi podataka.

5.3 Dijagram razmještaja

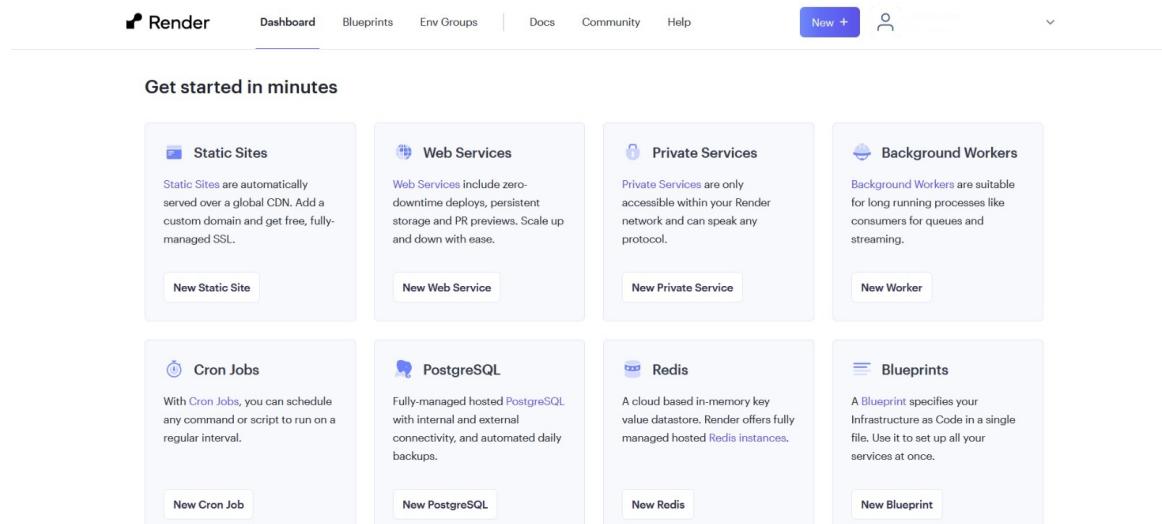
Na slici prikazan je dijagram razmještaja. Cjelokupan sustav je baziran na arhitekturi "klijent-poslužitelj". Klijentsko računalo i poslužitelj za frontend povezani su protokolom HTTP. Na poslužiteljskom računalu nalaze se Frontend-react, Backend-Spring i DB poslužitelj. Spring iz backenda dohvata podatke iz SQL baze podataka.



Slika 5.14: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Upute za puštanje u pogon su podijeljene u tri dijela jer svaki od tri dijela aplikacije (frontend, backend, baza podataka) zahtjeva posebnu pripremu za postavljanje na server. Početni koraci su jednaki za sve komponente, potrebno je prijaviti se na <https://render.com/> (u slučaju prvog korištenja registrirati se) te na izborniku kao na slici odabratи što se želi pustiti u pogon.



Slika 5.15: Izbornik sustava za puštanje u pogon na Renderu

Baza podataka

Puštanje u pogon započinje postavljanjem baze podataka, to činimo odabirom opcije **New PostgreSQL** u izborniku. Taj odabir će otvoriti prozor prikazan na slici.

New PostgreSQL

Name

A unique name for your PostgreSQL instance.

Database Optional

The PostgreSQL `dbname`

User Optional**Region**

The [region](#) where your PostgreSQL instance runs.

PostgreSQL Version

Slika 5.16: Izgled izbornika za puštanje baze podataka u pogon.

Za puštanje u pogon, potrebno je iz izbornika za regiju (*Region*) odabrati opciju Frankfurt, upisati ime baze podataka (npr. Error404-DentAll-BazaPodataka) u polje *Name* što se može vidjeti na slici iznad. Zatim odabrati besplatnu opciju u izborniku tipa (*Instance type*), prikazano na slici dolje. Nапослјетку је потребно pritisnuti gumb "Create Database" također prikazan na slici dolje. Sva ostala polja mogu ostati prazna.

Instance Type

For hobby projects

Free \$0 / month	256 MB (RAM) 0.1 CPU 1 GB (Storage)	Upgrade to enable more features Free instances do not support backups. A free database expires 90 days after creation. Only one free PostgreSQL instance can be active for any given Render user or team.
----------------------------	---	---

For professional use
Select an instance type for your PostgreSQL instance. Currently, we don't support downgrading PostgreSQL instances. Make sure to pick the instance type that works for you.

Starter \$7 / month	256 MB (RAM) 0.1 CPU 1 GB (Storage)	Standard \$20 / month	1 GB (RAM) 1 CPU 16 GB (Storage)
Pro \$95 / month	4 GB (RAM) 2 CPU 96 GB (Storage)	Pro Plus \$185 / month	8 GB (RAM) 4 CPU 256 GB (Storage)

Need a custom instance type? We support up to 512 GB RAM, 64 CPUs, and 5 TB storage.

i Access more features like [Point-in-Time Recovery](#) and [High Availability](#) by upgrading to a team plan.

[Create a team](#)

[Create Database](#)

Slika 5.17: Izgled izbornika za puštanje baze podataka u pogon.

Frontend i Backend

Puštanje u pogon frontenda i backenda započinje jednakom, odabirom opcije **New Web Service** u već prikazanom izborniku. Zatim je potrebno povezati vlastiti GitHub račun sa Renderom te upisati url repozitorija (upisati <https://github.com/Tomislav-Pranjic/Error404-DentAll> u prostor za javne repozitorije) prikazano na donjoj slici.

Public Git repository

Use a **public repository** by entering the URL below. Features like [PR Previews](#) and [Auto-Deploy](#) are not available if the repository has not been configured for Render.

[Continue](#)

Slika 5.18: Mjesto za upis URL-a mape u kojoj se nalazi web servis

Za početak će biti opisano puštanje u pogon backenda.

1. korak

Prvo ćemo odabrati ime web servisa, primjer "Error404-be", zatim odabrati istu regiju kao i za bazu podataka (Frankfurt), nadalje potrebno je odabrati granu koda sa koje puštamo backend u pogon, ime grane je "main", onda je potrebno upisati korijensku mapu (*IzvorniKod/error404-be*), te konačno od raznih vrsta web servisa potrebno odabrati pravi *Runtime*, za Backend je to **Docker**, kako prozor

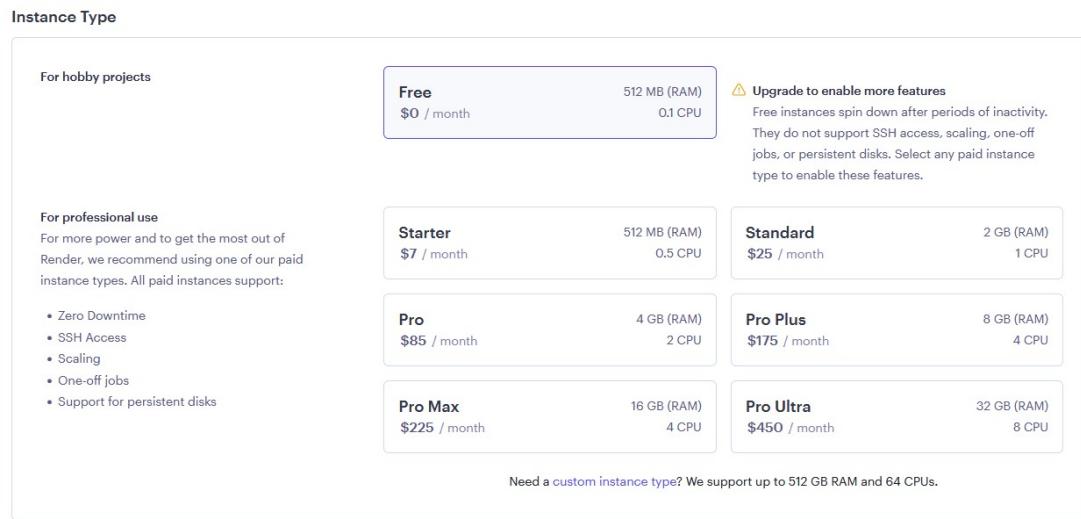
izgleda na kraju ovog koraka vidi se na slici.

Name A unique name for your web service.	Error404DentAll-be
Region The region where your web service runs.	Frankfurt (EU Central)
Branch The repository branch used for your web service.	main
Root Directory <small>Optional</small> Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.	IzvorniKod/error404-be
Runtime The runtime for your web service.	Docker

Slika 5.19: Mjesto za odabira imena, regije, grane, mape te vrste web servisa

2. korak

U ovom koraku odabiremo besplatnu verziju tipa servera (*Instance type*), isto kao i kod baze podataka, zatim stvaramo dvije varijable okoline
`OWNER_PASS = "$2a$10$6djQ3MpO06d5dgvp9ijBee6ZKKj5L5iJVrrDPQjtD4I6em2p7JjMu"`, i `PORT = 8080`, koje će služiti kako bismo mogli povezati backend i frontend.



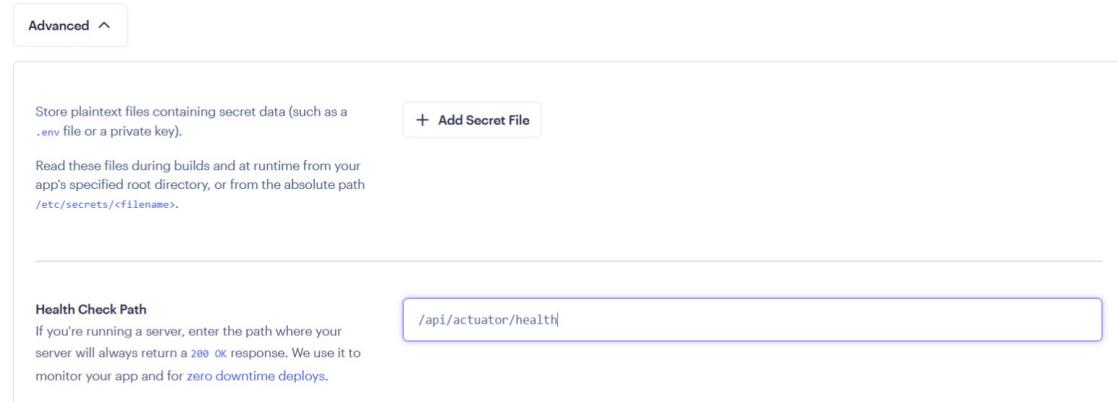
Environment Variables Optional
Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

OWNER_PASS	\$2a\$10\$6djQ3Mp006d5dgvp9ijBee6ZKKj5L5iJVrrDPQjtD4I6em2 p7JjMu	
PORT	8080	

Slika 5.20: Mjesto za odabira tipa, i upis varijabli okoline

3. korak

Dalje odabiremo opciju *Advanced* koja nam dopušta dodatne opcije uređivanja poput specificiranja *Health Check Path*, stranice koja će uvijek vraćati HTTP kod "200 OK", a služi za provjera rada backend-a.



Slika 5.21: Mjesto za specificiranja Health Check puta

4. korak

Dalje, potrebno je upisati lokaciju *Dockerfile-a*, relativno korijenskoj mapi (`/docker/maven/Dockerfile`), te konačno pritisnuti gumb **Create Web Service**, ostale postavke ostaviti kako su inicijalizirane.

Docker Build Context Directory

Docker build context directory. This is relative to your repository root. Defaults to the root.

IzvorniKod/error404-be/

Dockerfile Path

Path to your Dockerfile relative to the repository root. This is *not* relative to your Docker build context. For example, `./subdir/Dockerfile`.

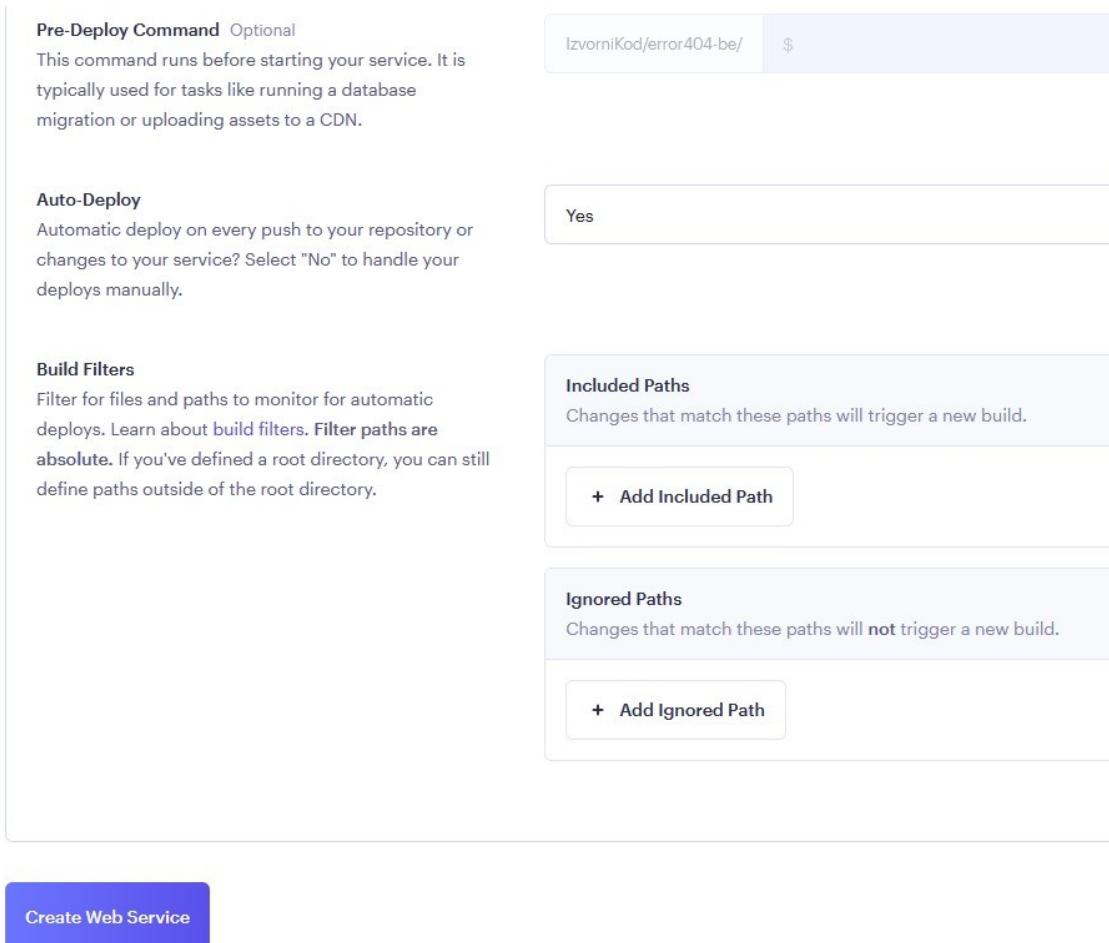
IzvorniKod/error404-be/

/docker/maven/Dockerfile

Docker Command

Add an optional command to override the Docker `CMD` for this service. This will also override the `ENTRYPOINT` if defined in your Dockerfile. Examples: `./start.sh --type=worker` or `./bin/bash -c cd /some/dir && ./start.sh`

Slika 5.22: Mjesto za specificiranja docerfile lokacije



Slika 5.23: Gumb za stvaranje web servisa

Dalje će biti opisan postupak puštanja frontenda u pogon.

1. korak

Za početak, kao i svagdje, odabiremo ime servisa, neka to za primjer bude "Error404-DentAll-fe", zatim odabiremo već poznatu regiju (Frankfurt), te istu granu kao i za backend ("main"), korijenska mapa ovoga je puta *IzvorniKod/error404-fe* te se upisuje na isto mjesto kao i kod backenda, ali za razliku od backenda, *Runtime* je **Node**. Kao *Build Command* upisujemo **yarn build**, a kao *Start Command* **yarn start-prod**.

Runtime

The runtime for your web service.

Node

Build Command

This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

IzvorniKod/error404-fe/

\$ yarn build

Start Command

This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

IzvorniKod/error404-fe/

\$ yarn start-prod

Slika 5.24: Konfiguracija frontenda

2. korak

Za razliku od backenda ne koristimo nikakve proširene mogućnosti nego samo stvaramo 2 varijable okoline, $API_URL = "https://error404dentall-be.onrender.com/api"$, i $PORT = 8080$, koje će služiti kako bismo mogli povezati frontend i backend. Nakon toga jednostavno se pritisne gumb **Create Web Service** i novi servis je stvoren.

Environment Variables Optional
Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

API_URL	https://error404dentall-be.onrender.com/api	
PORT	8080	
+ Add Environment Variable		

Slika 5.25: Konfiguracija varijabli okoline

6. Zaključak i budući rad

Zadatak naše grupe je bio stvoriti web aplikaciju "DentAll" koja bi omogućila tvrtkama koje omogućuju medicinskim turistima prijevoz i smještaj da vide i organiziraju te podatke. Nakon 12 tjedana rada, zadatak je djelomično ispunjen. Moguće je dodavati nove elemente preko backenda u bazu podataka, ali to nije moguće učiniti s frontend-a. Sustav na frontendu ispisuje upisane korisnike, vozače i smještaj, ali ne prikazuje upisane administratora. Moguće je otvoriti formu za upis novih administratora, vozača, smještaja, ali nije moguće otvoriti formu za unos novog korisnika. Nije moguće promijeniti upisane podatke niti ih isbrisati.

Projekt se može podjeliti na dvije faze. U prvoj fazi radili smo na podjeli rada, u početku smo imali samostalnu podjelu zadatka, a nakon otprilike tri tjedna rada konačno smo se podjelili u tri podtima, podtim za frontend koji je brojio tri člana, podtim za backend i bazu podataka koji su oba brojila po dva člana. Prva faza završava prvom predajom projekta.

U drugoj fazi ostaje podjela na tri podtima s tim da je podtim koji je radio na bazi podatka preuzeo sav rad na dokumentaciji. Naglasak u ovoj fazi bio je na implementaciji projekta, a faza je trajala do konačne predaje projekta.

Aplikaciju je moguće proširiti na mnoge načine, jedna ideja bila bi omogućiti samim korisnicima zdravstvenih usluga da mogu provjeriti informacije o svome posjetu. Rad na projektu dakako je vrijedno iskustvo za sve članove tima, svima nama je ovo bio prvi puta rad na ovakvoj aplikaciji i u ovakovom formatu, te se većina nas do sad nije susrela s tehnologijama poput LaTex-a i Git-a. Konflikata u timu nije bilo, ali nedostatak koordinatora i neiskustvo voditelja dovelo je do ponekad manjka komunikacije među članovima i konačno djelomično ispunjenje zadatka.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzb>
2. Learn Spring Boot, <https://www.baeldung.com/spring-boot>
3. Astah Community, <http://astah.net/editions/uml-new>
4. Visual Paradigm Online, <https://online.visual-paradigm.com/>
5. Upute za deploy na Render, <https://github.com/progi-devops>
6. Stack overflow, <https://stackoverflow.com/>
7. React reference overview, <https://react.dev/reference/react>
8. Get started with Bootstrap, <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
9. Java Networking, <https://docs.oracle.com/javase/8/docs/technotes/guides/>
10. Render, <https://render.com/>
11. JUnit 5, <https://junit.org/junit5/>
12. Selenium IDE, <https://www.selenium.dev/selenium-ide/>

Indeks slika i dijagrama

3.1 Sveobuhvatni prikaz	17
3.2 Početni koraci	18
3.3 Pregled podataka u bazi podataka	18
3.4 Prikaz uloga korisničkog i prijevoznog administratora	19
3.5 Prikaz unosa raspoloživog smještaja	20
3.6 Prikaz korisničke strane	21
3.7 Sekvencijski dijagram UC1-Prijava	22
3.8 Sekvencijski dijagram UC2-Dodavanje novog administratora	23
3.9 Sekvencijski dijagram UC3-Unos raspoloživog smještaja	24
4.1 Dijagram baze podataka	29
4.2 Cjelokupan dijagram razreda	30
4.3 Dijagram razreda - <i>Controller</i>	31
4.4 Dijagram razreda - <i>Repository</i>	32
4.5 Dijagram razreda - <i>Service</i>	33
4.6 Dijagram razreda - <i>Domain</i>	34
4.7 Dijagram stanja	36
4.8 Dijagram aktivnosti	38
4.9 Dijagram komponenti	39
5.1 Kod klase AdminTest	41
5.2 Kod klase AccommodationTest	42
5.3 Kod klase UserTreatmentTest, funkcija setUp	43
5.4 Kod klase UserTreatmentTest, test getUserInfoDef i test getUserInfoOne	43
5.5 Kod klase UserTreatmentTest, test getUserInfoToo	44
5.6 Snimka kako bi dodavanje novog korisnika trebalo izgledati	44
5.7 Rezultati testa	45
5.8 Snimka kako bi dodavanje novog vozača trebalo izgledati	45
5.9 Rezultati testa	46
5.10 Snimka kako bi dodavanje novog korisnika trebalo izgledati	46
5.11 Rezultati testa	47
5.12 Snimka kako bi dodavanje novog smještaja trebalo izgledati	47
5.13 Rezultati testa	48
5.14 Dijagram razmještaja	49
5.15 Izbornik sustava za puštanje u pogon na Renderu	50
5.16 Izgled izbornika za puštanje baze podataka u pogon.	51
5.17 Izgled izbornika za puštanje baze podataka u pogon.	52

5.18 Mjesto za upis URL-a mape u kojoj se nalazi web servis	52
5.19 Mjesto za odabira imena, regije, grane, mape te vrste web servisa	53
5.20 Mjesto za odabira tipa, i upis varijabli okoline	54
5.21 Mjesto za specificiranja Health Check puta	54
5.22 Mjesto za specificiranja docerfile lokacije	55
5.23 Gumb za stvaranje web servisa	56
5.24 Konfiguracija frontenda	57
5.25 Konfiguracija varijabli okoline	57

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 18. listopada 2023.
- Prisustvovali: T. Pranjić, J. Mihelčić, A. Mesić, D. Mišetić, A. Sorić, I. Čorluka, N. Perić
- Teme sastanka:
 - Formiranje tima
 - Odabir voditelja

2. sastanak

- Datum: 25. listopada 2023.
- Prisustvovali: T. Pranjić, J. Mihelčić, A. Mesić, D. Mišetić, A. Sorić, I. Čorluka, N. Perić
- Teme sastanka:
 - Razjašnjavanje pojedinosti oko projektnog zadatka

3. sastanak

- Datum: 25. listopada 2023.
- Prisustvovali: A. Sorić, D. Mišetić
- Teme sastanka:
 - Opis projektnog zadatka

4. sastanak

- Datum: 08. siječnja 2024.
- Prisustvovali: T. Pranjić, J. Mihelčić, A. Mesić, D. Mišetić, A. Sorić, I. Čorluka, N. Perić
- Teme sastanka:
 - Plan rada za drugi ciklus

Tablica aktivnosti

	Tomislav Pranjić	Ante Sorić	Diego Mišetić	Josip Mihelčić	Antonia Mesić	Ivan Čorluka	Nikola Perić
Upravljanje projektom	10			8			
Opis projektnog zadatka		8	8				
Funkcionalni zahtjevi							8
Opis pojedinih obrazaca	1			2			
Dijagram obrazaca	3			3			
Sekvencijski dijagrami			6				
Opis ostalih zahtjeva						2	
Arhitektura i dizajn sustava	2						1
Baza podataka	3		3				
Dijagram razreda		5		2			
Dijagram stanja			6				
Dijagram aktivnosti			5				
Dijagram komponenti			5				
Korištene tehnologije i alati	2						
Ispitivanje programskog rješenja	6						
Dijagram razmještaja			5				
Upute za puštanje u pogon	4						
Dnevnik sastajanja							
Zaključak i budući rad	2						
<i>Backend: Generičke funkcionalnosti i sigurnost</i>		16		18			
<i>Frontend: Izrada početne stranice</i>							8
<i>Frontend: Izrada stranice za upravljanje prijevoznikom</i>							17
<i>Frontend: Izrada stranice za upravljanje smještajem</i>					14		

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Tomislav Pranjić	Ante Šorić	Diego Mišetić	Josip Mihelčić	Antonia Mesić	Ivan Čorluka	Nikola Perić
<i>Frontend: Izrada stranice za upravljanje korisnicima</i>						16	
<i>Baza podataka: Izrada baze podataka</i>	4						
<i>Frontend: Izrada stranice za prijavu</i>							9
<i>Baza podataka: Izrada baze podataka</i>	4						
<i>Backend: Spajanje s bazom podataka</i>			8				
<i>Backend: Implementacija funkcionalnosti</i>		15		30			
<i>Backend: Povezivanje s frontendom</i>				30			
<i>Puštanje sustava u pogon</i>	20			15			

Dijagrami pregleda promjena



