Programsko inženjerstvo

Ak. god. 2023./2024.

DentAll

Dokumentacija, Rev. *2*

Grupa: *ZuBit*

Voditelj: *Danko Delimar*

Datum predaje: *19. 1. 2024.*

Nastavnik: *Goran Rajić*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	*	20.10.2023.
0.2	Dodan opis projekta, dionici, funkcionalni zahtjevi i neki use casevi	*	25.10.2023.
0.3	Dodan ostatak *Use Caseva*	*	29.10.2023.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	*	11.09.2013.
1.1	Implementacija i korisničko sučelje	* *	14.1.2024.
1.2	Ispitivanje programskoj rješenja	*	15.1.2024.
1.3	Dijagram razmještaja i upute za puštanje u pogon	*	19.01.2024.
2.0	Konačni tekst predloška dokumentacije	*	19.01.2024.

2. Opis projektnog zadatka

Ideja ovog projekta je napraviti aplikaciju koja olakšava dentalnim klinikama proces ponude potpunog plana liječenja koji uključuje medicinske podatke, smještaj u kojem će se pacijent nalaziti i prijevoz od klinike do smještaja i obratno. Ovo je važna aplikacija za klinike koje žele svoje usluge ponuditi pacijentima iz drugih gradova ili država, tzv. "dentalni turizam", zato što ih mogućnost takve ponude čini puno privlačnijima potencijalnim klijentima koji nisu upoznati s lokacijom na kojoj se klinika nalazi. Time se klijenti, a i klinike mogu fokusirati samo na posao, a naša aplikacija obavlja svu logistiku.

Aplikaciju mogu koristiti sve dentalne klinike kojima je potrebna organizacija smještaja i prijevoza na relaciji smještaj-klinika. Posebice ciljamo na klinike koje žele proširiti posao oglašavanjem u inozemstvu i žele ponuditi potpuni paket potencijalnim klijentima kako bi se istaknule naspram konkurencije. Takvim klinikama je ovakva aplikacija jako potrebna zato što je jedan od glavnih razloga zašto dentalni turizam nije popularan koliko bi mogao biti činjenica da je potencijalnim klijentima težak zadatak samostalno organizirati ovakvo putovanje u stranu državu o kojoj možda ništa ne znaju.

U trenutnom obliku aplikacija cilja dentalne klinike jer su zahvati koje dentalne klinike nude relativno jednostavni za definirati u smislu termina i jer taj sektor medicine ima veliki potencijal za privući ljude iz stranih država zbog značajno jeftinije usluge. Ukoliko je potrebno aplikacija se jednostavno može promijeniti i nadograditi kako bi uslugu mogla pružati cijelom medicinskom sektoru ili bilo kakvoj djelatnosti koja želi ponuditi takav potpuni paket smještaja, prijevoza i usluge.

Aplikacija se može koristiti iz perspektive tri administratora koji predstavljaju tri dijela paketa koje aplikacija spaja, a to su smještajni administrator, prijevozni administrator i korisnički administrator. Krajnji korisnik, tj. pacijent klinike, nema interakciju s aplikacijom nego samo na račun elektroničke pošte koji mu je definiran od strane korisničkog administratora prima informaciju o cijelom paketu smještaja, prijevoza, kliničkih usluga i datuma kada treba stići u državu.

Smještajni administrator ima najveće ovlasti u sustavu i može dodavati nove korisnike i dodjeljivati im uloge. Glavna uloga smještajnog administratora je dodavanje u sustav novih smještaja i popunjavanje osnovnih podataka o tim smještajima. Smještajnim administratorima je omogućen pregled lokacije stana na karti uz pomoć Google Maps API-a. Svi osnovni podatci se mogu mijenjati, a cijeli smještaj se može izbrisati. Osnovni podatci uključuju:

- Tip stana (u vlasništvu klinike ili ne)
- Veličina stana
- Ocjena stana
- Adresa
- Vremenska dostupnost

Prijevozni administrator ima glavnu ulogu u unosu osnovnih podataka o prijevoznicima koji su dostupni u aplikaciji. Osnovni osobni podatci prijevoznika se ne mogu mijenjati, svi ostali se mogu, a cijeli prijevoznik se može izbrisati. Osnovni podatci uključuju:

- Ime i prezime prijevoznika
- Email
- Broj telefona
- Tip prijevoznog sredstva
- Kapacitet prijevoznog sredstva
- Radno vrijeme

Korisnički administrator ima glavnu ulogu u unosu osnovnih podataka o pacijentima klinike. Detalje tretmana ne unose korisnički administratori već se oni dohvaćaju iz postojećeg sustava klinike. Moguće je mijenjati podatke o pacijentima i brisati pacijente. Osnovi podatci o pacijentima su:

- Ime i prezime pacijenta
- Email

- Broj telefona
- Vrijeme dolaska u državu
- Mjesto dolaska u državu
- Preferencija o veličini i ocjeni stana

Svaki korisnički račun može imati više administratorskih uloga. Svaki administrator ima web sučelje unutar kojeg može obavljati potrebne operacije dodavanja, izmjenjivanja i brisanja podataka. Takvo jednostavno web sučelje olakšava posao administratora kada moraju raditi potrebne izmjene.

Kada korisnički administrator unese novog korisnika aplikacija mora provjeriti postoji li dostupan smještaj u danom terminu. Kada se pronađe termin u kojem postoji slobodan smještaj čeka se potvrda od interne aplikacije klinike da je zaključan plan tretmana. Kada se plan tretmana zaključa aplikacija automatski šalje poruku elektroničke pošte klijentu sa cijelim paketom i šalje dostupnim prijevoznicima informacije o klijentu i smještaju.

Aplikacija ne prati podatke o tretmanu pacijenta već podatke mora pratiti interna aplikacija klinike te se tada oni uključuju u kompletan plan iz spomenute aplikacije. Također, aplikacija ne dozvoljava nikakav pristup ili pregled neregistriranim korisnicima s obzirom da je zamišljena za internu uporabu unutra klinike, a ne za vanjsku uporabu od strane korisnika. Sva komunikacija s vanjskim korisnicima kao što su pacijenti ili prijevoznici odvija se automatski putem elektroničke pošte.

S obzirom na to da je glavna funkcionalnost aplikacije smanjenje mentalnog napora vezanog za logistiku, moguća proširenja aplikacije većinom su vezana za dodatno olakšanje logistike. Moguća je implementacija povezanosti s internacionalnim prijevoznicima kako bi klijenti klinika morali učiniti što je manje moguće. Također je moguća integracija sa sustavima vlakova ili avioprijevoznika kako bi se klijentu pružio potpuno organizirani plan puta do destinacije. Druga vrsta mogućeg proširenja je automatsko izračunavanje troška ovakvog plana i automatsko uračunavanje tog troška u cjelokupni trošak tretmana. U taj trošak bili bi uključeni troškovi poput troškova prijevoza i troškova čišćenja stanova u kojima klijenti borave za vrijeme boravka u državi. Povezano s time, u aplikaciju se mogu uključiti i usluge čistača kojima se automatski šalje poruka kada određena smještajna jedinica treba čišćenje.

U našem kratkom istraživanju sličnih aplikacija nismo pronašli aplikaciju koja rješava isti problem. Većina se aplikacija u sektoru dentalnog turizma bavi prikazom tretmana pacijentima, ali ne zadiru u pitanja kako doći do neke klinike ili gdje stanovati za vrijeme obavljanja usluge. Mislimo da zato ova aplikacija ima dobru priliku iskoristiti nišu u tržištu koja trenutno nije jako zastupljena. Također je moguće da za ovakvu funkcionalnost klinike trenutno imaju interna rješenja, no naša aplikacija će omogućiti da svaka zainteresirana klinika ima ovakve mogućnosti uz jednostavan oblik pretplate.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Dentalne klinike
2. Vlasnici smještaja
3. Prijevoznici
4. Administratori koji koriste aplikaciju
5. Klijenti, tj. pacijenti
6. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Smještajni administrator (inicijator) može:
   (a) Pregledavati podatke o smještajima (tip, ocjena, adresa, period dostupnosti)
   (b) Dodavati nove korisnike (ne klijente nego korisnike sustava)
   (c) Korisnicima dodavati nove uloge
   (d) Dodavati, izmjenjivati i brisati podatke o smještaju
   (e) Vidjeti prikaz smještaja na karti

2. Prijevozнички administrator (inicijator) može:
   (a) Pregledavati podatke o prijevoznicima
   (b) Dodati osnovne osobne podatke prijevoznika
   (c) Dodati kontaktne podatke i podatke o vrsti i kapacitetu vozila
   (d) Izmjenjivati neosnovne podatke (kontakt, vrsta i kapacitet vozila)
   (e) Izbrisati prijevoznika
3. Korisnički administrator (inicijator) može:
   (a) Dodati podatke o klijentima (osobni podatci, kontakt, preferencije o smještaju)
4. Baza podataka (sudionik) može:
   (a) Pohranjuje sve podatke o prijevoznicima
   (b) Pohranjuje sve podatke o smještaju
   (c) Pohranjuje nemedicinske podatke o klijentima

UC1 - Prijava smještajnog administratora

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Prijaviti se u sustav kao smještajni administrator
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke te odabir uloge smještajnog administratora
  2. Potvrda o postojanju računa i ispravnosti podataka
  3. Pristup funkcijama smještajnog administratora
- **Opis mogućih odstupanja:**

  2.a Uneseni podatci nisu u točnom formatu
  1. Obavijestiti korisnika koji podatci nisu u točnom formatu
  2. Korisnik mijenja potrebne podatke i pokušava opet

  2.b Korisnički račun ne postoji, lozinka nije točna ili korisničko ime nije točno
  1. Obavijestiti korisnika da su korisničko ime ili lozinka netočni
  2. Korisnik mijenja potrebne podatke i pokušava opet

UC2 - Prijava prijevoznog administratora

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Prijaviti se u sustav kao prijevozni administrator
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke te odabir uloge prijevoznog administratora
  2. Potvrda o postojanju računa i ispravnosti podataka
  3. Pristup funkcijama prijevoznog administratora
- **Opis mogućih odstupanja:**

  2.a Uneseni podatci nisu u točnom formatu
  1. Obavijestiti korisnika koji podatci nisu u točnom formatu
  2. Korisnik mijenja potrebne podatke i pokušava opet

  2.b Korisnički račun ne postoji, lozinka nije točna ili korisničko ime nije točno
  1. Obavijestiti korisnika da su korisničko ime ili lozinka netočni
  2. Korisnik mijenja potrebne podatke i pokušava opet

UC3 - Prijava korisničkog administratora

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Prijaviti se u sustav kao korisnički administrator
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke te odabir uloge korisničkog administratora
  2. Potvrda o postojanju računa i ispravnosti podataka
  3. Pristup funkcijama korisničkog administratora
- **Opis mogućih odstupanja:**

  2.a Uneseni podatci nisu u točnom formatu
  1. Obavijestiti korisnika koji podatci nisu u točnom formatu
  2. Korisnik mijenja potrebne podatke i pokušava opet

  2.b Korisnički račun ne postoji, lozinka nije točna ili korisničko ime nije točno
  1. Obavijestiti korisnika da su korisničko ime ili lozinka netočni
  2. Korisnik mijenja potrebne podatke i pokušava opet

UC4 - Dodavanje novog korisnika

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Dodati novog korisnika sustava
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava smještajnog administratora
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke novog korisnika te odabir svih uloga koje će korisnik imati
  2. Provjera postoji li već korisnik s takvim imenom
  3. Upis novog korisnika u bazu podataka
- **Opis mogućih odstupanja:**

  2.a Polja korisničko ime i/ili lozinka su prazni
  1. Obavijestiti korisnika koji su podatci prazni
  2. Korisnik mijenja potrebne podatke i pokušava opet

  2.b Postoji korisnički račun s tim imenom
  1. Obavijestiti korisnika da postoji račun s tim imenom
  2. Korisnik mijenja korisničko ime i pokušava opet

2.c Nije odabrana niti jedna uloga
    1. Obavijestiti korisnika da račun mora imati neku ulogu
    2. Korisnik dodaje jednu ili više uloga i pokušava opet

UC5 - Pregled postojećih smještaja

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Pregledati postojeće smještaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava smještajnog administratora
- **Opis osnovnog tijeka:**
    1. Smještajni administrator odabire opcije ”Postojeći smještaji”
    2. Aplikacija u obliku liste prikazuje sve postojeće smještaje i kartu s njihovim adresama

UC6 - Dodavanje novog smještaja

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Dodati novi smještaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava smještajnog administratora
- **Opis osnovnog tijeka:**
    1. Unos potrebnih podataka za dodavanje novog smještaja
    2. Provjera jesu li uneseni svi podatci
    3. Upis novog smještaja u bazu podataka
- **Opis mogućih odstupanja:**

2.a Neki od podataka nije unesen
    1. Obavijestiti korisnika koji podatci su prazni
    2. Korisnik mijenja potrebne podatke i pokušava opet

UC7 - Promjena podataka smještaja

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Promijeniti podatke smještaja
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava smještajnog administratora
- **Opis osnovnog tijeka:**
    1. Smještajni administrator odabire ”Promijeni” opciju na smještaju
    2. Smještajni administrator mijenja podatke o smještaju

3. Smještajni administrator sprema promjene
4. Pregledava se ispravnost podataka
5. Mijenjaju se podatci u bazi podataka

- **Opis mogućih odstupanja:**

3.a Smještajni administrator izlazi bez spremanja
1. Podatci se ne spremaju u bazu podataka

4.a Neki od podataka nije unesen
1. Obavijestiti smještajnog administratora koji podatci su prazni
2. Smještajni administrator mijenja potrebne podatke i pokušava opet

UC8 - Brisanje smještaja

- **Glavni sudionik:** Smještajni administrator
- **Cilj:** Promijeniti podatke smještaja
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava smještajnog administratora
- **Opis osnovnog tijeka:**
  1. Smještajni administrator odabire ”Izbriši” opciju na smještaju
  2. Aplikacija šalje ”jeste li sigurni?” upit smještajnom administratoru
  3. Ako je, smještaj se briše iz baze, u protivnom se ne desi ništa

UC9 - Pregled postojećih prijevoznika

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Pregledati postojeće prijevoznike
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava prijevoznog administratora
- **Opis osnovnog tijeka:**
  1. Prijevozni administrator odabire opcije ”Postojeći prijevoznici”
  2. Aplikacija u obliku liste prikazuje sve postojeće prijevoznike

UC10 - Dodavanje novog prijevoznika

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Dodati novog prijevoznika
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava prijevoznog administratora
- **Opis osnovnog tijeka:**
  1. Unos potrebnih podataka za dodavanje novog prijevoznika

2. Provjera jesu li uneseni svi podatci
3. Upis novog prijevoznika u bazu podataka

- **Opis mogućih odstupanja:**

2.a Neki od podataka nije unesen

1. Obavijestiti prijevoznog administratora koji podatci su prazni
2. Prijevozni administrator mijenja potrebne podatke i pokušava opet

UC11 - Promjena podataka o prijevozniku

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Promijeniti podatke o prijevozniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava prijevoznog administratora
- **Opis osnovnog tijeka:**
  1. Prijevozni administrator odabire ”Promijeni” opciju na postojećem prijevozniku
  2. Prijevozni administrator mijenja podatke o prijevozniku
  3. Prijevozni administrator sprema promijene
  4. Pregledava se ispravnost podataka
  5. Mijenjaju se podatci u bazi podataka
- **Opis mogućih odstupanja:**

3.a Prijevozni administrator izlazi bez spremanja

1. Podatci se ne spremaju u bazu podataka

4.a Neki od podataka nije unesen

1. Obavijestiti prijevoznog administratora koji podatci su prazni
2. Prijevozni administrator mijenja potrebne podatke i pokušava opet

UC12 - Brisanje prijevoznika

- **Glavni sudionik:** Prijevozni administrator
- **Cilj:** Promijeniti podatke smještaja
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava prijevoznog administratora
- **Opis osnovnog tijeka:**
  1. Prijevozni administrator odabire ”Izbriši” opciju na prijevozniku
  2. Aplikacija šalje ”jeste li sigurni?” upit prijevoznom administratoru
  3. Ako je, prijevoznik se briše iz baze, u protivnom se ne desi ništa

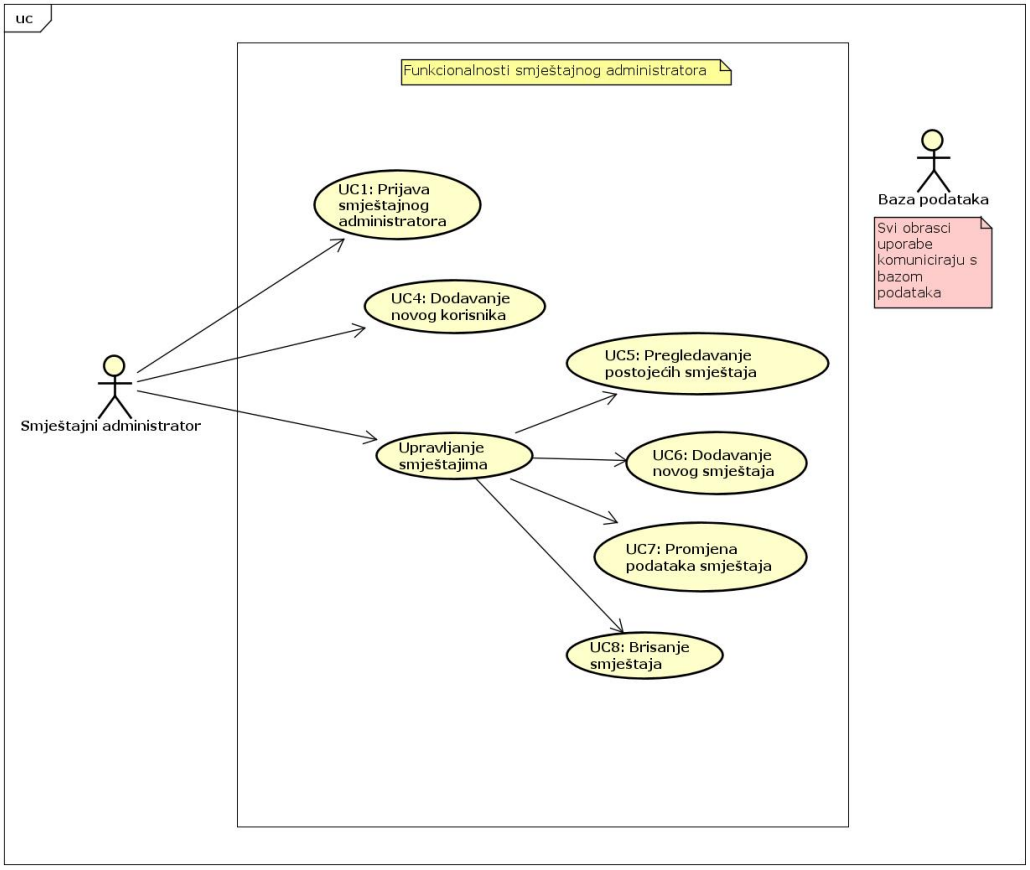
UC13 - Dodavanje novog klijenta

- **Glavni sudionik:** Korisnički administrator
- **Cilj:** Dodati novog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava korisničkog administratora
- **Opis osnovnog tijeka:**
  1. Unos potrebnih podataka za dodavanje novog korisnika
  2. Dohvat medicinskih podataka o korisniku
  3. Provjera jesu li uneseni svi podatci
  4. Dodjela smještaja klijentu
  5. Slanje poruke elektroničke pošte klijentu i prijevozniku o zaključenom planu
- **Opis mogućih odstupanja:**

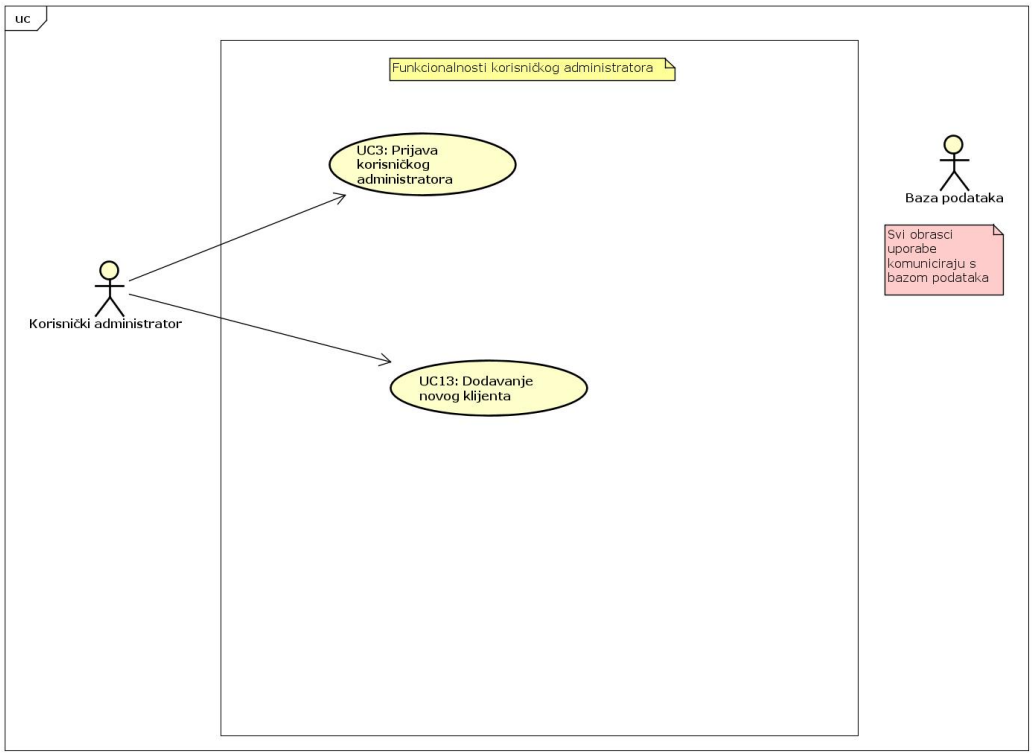
  2.b Ne mogu se dohvatiti medicinski podatci
  1. Aplikacija ponovno pokušava dohvatiti podatke
  2. Ako je podatke nemoguće dohvatiti, nemoguće je unijeti novog korisnika

  3.a Neki od podataka nije unesen
  1. Obavijestiti korisničkog administratora koji podatci su prazni
  2. Korisnički administrator mijenja potrebne podatke i pokušava opet

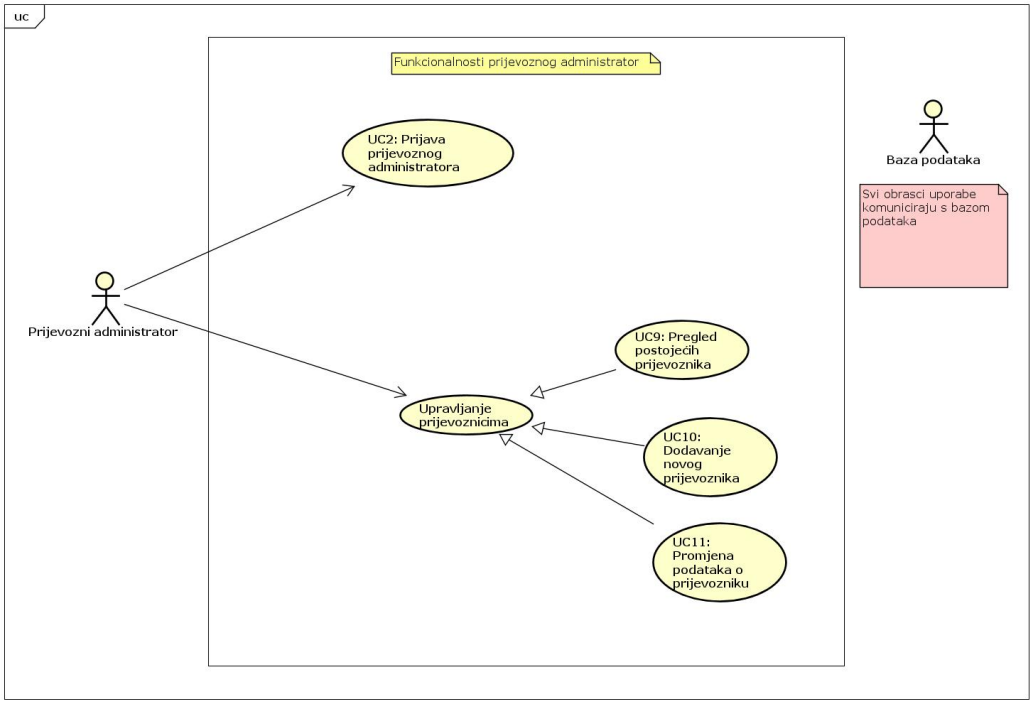
Dijagrami obrazaca uporabe



Slika 3.1: Funkcijski zahtjevi smještajnog administratora



Slika 3.2: Funkcijski zahtjevi transportnog administratora

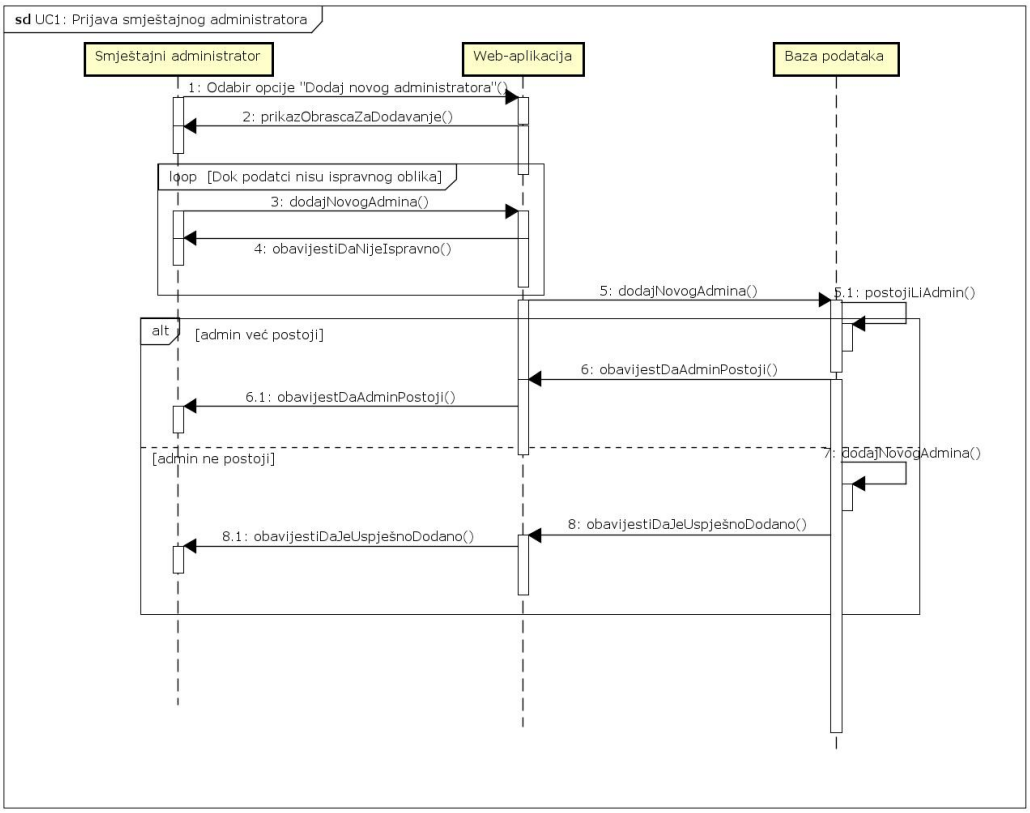


Slika 3.3: Funkcijski zahtjevi korisničkog administratora

3.1.1 Sekvencijski dijagrami

Obrazac uporabe UC4: Dodavanje novog korisnika

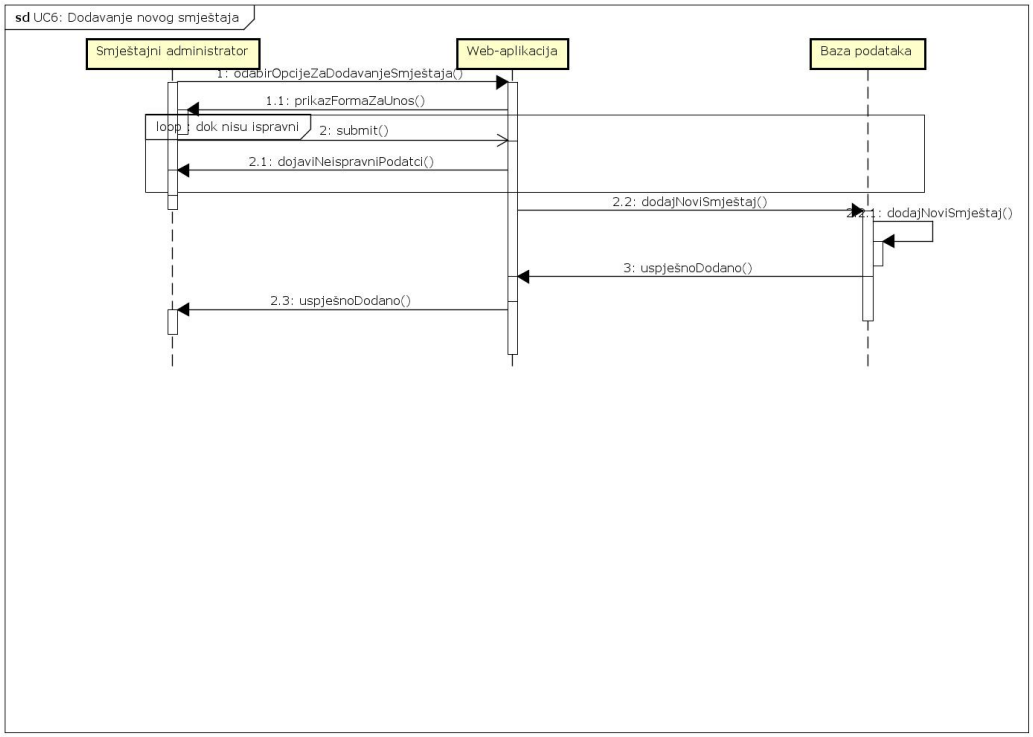
Smještajni administrator odabire opciju dodaj novog korisnika. Web-aplikacija otvara obrazac za dodavanje novog korisnika. Smještajni administrator upisuje korisničko ime, lozinku i odabire uloge koje će novi administrator imati te šalje zahtjev aplikaciji. Aplikacija radi provjeru ispravnosti tih podataka i ako su ispravni šalje ih bazi podataka. Baza podataka provjerava postoji li već administrator s tim korisničkim imenom i ako ne postoji dodaje novog administratora. Smještajnom administratoru aplikacija javlja da je unos uspješno izvršen.



Slika 3.4: Sekvencijski dijagram za UC4

Obrazac uporabe UC6: Dodavanje novog smještaja

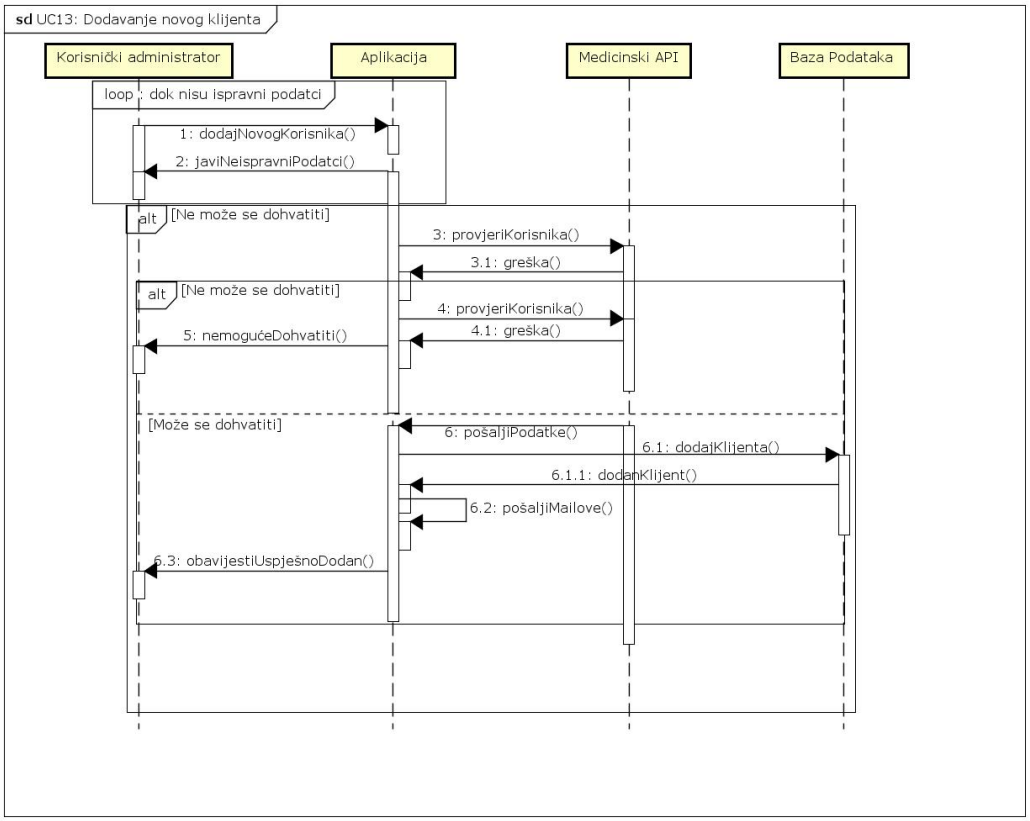
Smještajni administrator odabire opciju dodaj novi smještaj. Web-aplikacija otvara obrazac za dodavanje novog smještaja. Smještajni administrator upisuje sve potrebne informacije za dodavanje smještaja. Ako su svi potrebni podatci ispunjeni aplikacija šalje podatke bazi podataka. Baza podataka registrira novi smještaj i vraća informaciju da je smještaj ispravno unesen.



Slika 3.5: Sekvencijski dijagram za UC6

Obrazac uporabe UC13: Dodavanje novog klijenta

Korisnički administrator odabire opciju dodaj novog korisnika. Aplikacija mu pokazuje obrazac za dodavanje novog klijenta. Nakon što upiše sve potrebne podatke aplikacija pokušava dohvatiti podatke od klijenta iz API-a klinike. Ako prvi put ne uspije pokušava opet, a ako i drugi put ne uspije javlja administratoru da trenutno unos nije moguć. Ako je dohvat uspješan klijent se dodaje u bazu podataka i aplikacija šalje poruku elektroničke pošte klijentu i prijevozniku odgovornom za klijenta, a administratoru se javlja da je unos uspješan.



Slika 3.6: Sekvencijski dijagram za UC13

3.2 Ostali zahtjevi

- Sustav treba omogućiti udaljeni pristup administratorima
- Sustav treba omogućiti istovremeni rad više korisnika
- Sve operacije s bazom podataka moraju biti sigurne, a lozinke zaštićene
- Sustav bilo kakve greške treba dojaviti administratorima na pregledan način umjesto da sruši sustav
- Sustav mora poslati pravilno formatirane poruke elektroničke pošte kako one ne bi završile u spam-u
- Sustav mora biti dovoljno općenito implementiran kako bi bio lagano nadogradiv
- Sustav mora biti izgrađen koristeći principe objektno orijentiranog programiranja
- Sustav mora biti jednostavno izgrađen i svi obavezni podatci za unos moraju biti jasno naznačeni
- Greška u unosu ne smije srušiti sustav

4. Arhitektura i dizajn sustava

Za arhitekturu sustava odabrali smo klasičan klijent-server pristup.

Klijent

Strana klijenta je web stranica izgrađena u programskom jeziku JavaScript uz pomoć biblioteke React. Odabrali smo ovu tehnologiju jer je React danas najkorištenija biblioteka za razvoj web stranica i kao takva nudi najbolji ekosustav funkcionalnosti i podrške. Korišteno razvojno okruženje je VScode. Zadatak klijenta je slanje zahtjeva prema serveru koji ih zatim obrađuje. Svi zahtjevi se šalju pomoću HTTP POST metode i šalju se u JSON formatu prema serveru.

Server

Za server stranu odabrali smo programski jezik Javu i razvojno okruženje *Spring Boot*. *Spring Boot* smo odabrali jer je standardno razvojno okruženje za jezik Javu, a Javu smo odabrali kako bismo na prirodan način mogli sustav implementirati koristeći objektno orijentiranu paradigmu. Za razvoj serverskog koda korišten je alata Intellij IDEA. *Spring Boot* nam također nudi neke dodate pogodnosti kao što je proširenje *Spring Security* koje znatno olakšava proces implementacije sigurnog i točnog procesa prijave i registracije korisnika. Server prima zahtjeve od klijenta u JSON formatu i pretvara te zahtjeve u Java objekte nad kojima izvršava daljnje operacije. Kada server obradi zahtjev šalje natrag HTTP odgovor s odgovarajućim statusnim kodom kako bi klijent znao je li operacija uspjela ili nije.

4.1 Baza podataka

Kao sustav za upravljanje bazama podataka odabrali smo PostgreSQL. Implementacija naše PostgreSQL baze podataka obuhvaća nekoliko ključnih elemenata, uključujući organizaciju podataka u tablicama i uspostavljanje veza između tablica radi složenih upita. Baza podatka sastoji se od slijedećih entiteta:

- KLINIKA
- SMJEŠTAJ

- PRIJEVOZNIK
- VOZILO
- KORISNIK
- PUTOVANJE

4.1.1 Opis tablica

KLINIKA Entitet KLINIKA sadrži informacije o ID-u klinike, nazivu i adresi. Prema tome, entitet KLINIKA posjeduje sljedeće atribute: IDKlinika, naziv i adresa. Entitet KLINIKA je u vezi *One-to-Many* s entitetom SMJESTAJ preko atributa IDKlinika i u vezi *One-to-Many* s enitetom PUTOVANJE preko atributa IDKlinika. Također je u *One-to-Many* vezi s enitetom PUTOVANJE preko atributa adresa.

KLINIKA		
IDKlinika	INT	Identifikacijski ključ klinike
naziv	VARCHAR	Naziv klinike
adresa	VARCHAR	Adresa klinike

SMJESTAJ Entitet SMJESTAJ sadrži podatke o ID-u smještaja, tipu stana, kategoriji opremljenosti, adresi kao i vremenskom periodu dostupnosti za korištenje. Sukladno tome, entitet SMJESTAJ posjeduje sljedeće atribute: IDSmjestaj, tip, kategorija, adresa i dostupnost. Entiet SMJESTAJ je u vezi *Many-to-One* s entitetom KLINIKA preko atributa IDKlinika i u vezi *One-to-Many* s enitetom PUTOVANJE preko atributa IDPutovanje. Također je u *One-to-Many* vezi s enitetom PUTOVANJE preko atributa adresa.

SMJESTAJ		
IDSmjestaj	INT	Identifikacijski ključ smještaja
tip	VARCHAR	Tip stana
kategorija	VARCHAR	Kategorija opremljenosti
adresa	VARCHAR	Adresa smještaja
dostupnost	INTERVAL	Vremenski period dostupnosti za korištenje

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

SMJESTAJ		
IDKlinika	INT	Identifikacijski ključ klinike

PRIJEVOZNIK Entitet PRIJEVOZNIK sadrži informacije o ID-u prijevoznika, kontaktnim podacima i o radnom vremenu u kojem je prijevoznik raspoloživ. Prema tome, entitet PRIJEVOZNIK posjeduje sljedeće atribute: IDPrijevoznik, kontakt i radnoVrijeme. Entitet PRIJEVOZNIK je u vezi *One-to-Many* s entitetom VOZILO preko atributa IDPrijevoznik i u vezi *One-to-Many* s enitetom PUTOVANJE preko atributa IDPrijevoznik.

PRIJEVOZNIK		
IDPrijevoznik	INT	Identifikacijski ključ prijevoznika
kontakt	VARCHAR	Kontaktni podatci prijevoznika
radnoVrijeme	TIME	Radno vrijeme u kojem su prijevoznici raspoloživi

VOZILO Entitet VOZILO sadrži informacije o ID-u vozila, vrsti i kapacitetu prijevoznog sredstva. Prema tome, entitet VOZILO posjeduje sljedeće atribute: IDVozilo, vrsta i kapacitet. Entitet VOZILO je u vezi *Many-to-One* s entitetom PRIJEVOZNIK preko atributa IDPrijevoznik.

VOZILO		
IDVozilo	INT	Identifikacijski ključ vozila
vrsta	VARCHAR	Vrsta vozila
kapacitet	VARCHAR	Kapacitet vozila
IDPrijevoznik	INT	Identifikacijski ključ prijevoznika

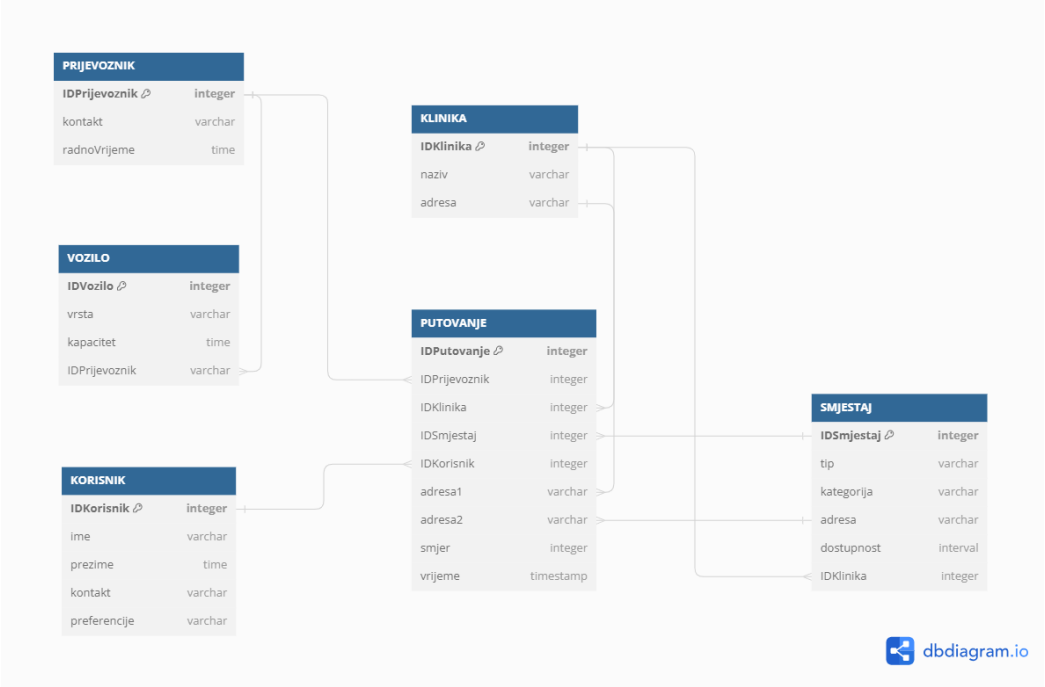
KORISNIK Entitet KORISNIK sadrži informacije o ID-u korisnika, imenu, prezimenu, kontaktnim podacima i preferencijama vezanim uz veličinu i kvalitetu smještaja. Prema tome, entitet KORISNIK posjeduje sljedeće atribute: IDKorisnik, ime, prezime, kontakt i preferencije. Entitet KORISNIK je u vezi *One-to-Many* s enitetom PUTOVANJE preko atributa IDKorisnik.

KORISNIK		
IDKorisnik	INT	Identifikacijski ključ korisnika
ime	VARCHAR	Ime korisnika
prezime	VARCHAR	Prezime korisnika
kontakt	VARCHAR	Kontakt korisnika
preferencije	VARCHAR	Preferencije vezane uz veličinu i kvalitetu smještaja

PUTOVANJE Entitet PUTOVANJE sadrži informacije o ID-u putovanja, vremenu i smjeru putovanja. Prema tome, entitet PUTOVANJE posjeduje sljedeće atribute: IDPutovanje, vrijeme i smjer. Entitet PUTOVANJE j u vezi *Many-to-One* s enitetom KLINIKA preko atributa IDKorisnik, u vezi *Many-to-One* s enitetom SMJESTAJ preko atributa IDSmjestaj, u vezi *Many-to-One* s enitetom KORISNIK preko atributa IDKorisnik, u vezi *Many-to-One* s enitetom PRIJEVOZNIK preko atributa IDPrijevoznik. Atributi adresa1 i adresa2 su atributi iz kojih saznajemo adresu polaska ili dolaska u ovisnosti o smjeru koji može biti 1 ili 0. Entitet PUTOVANJE u vezi je *Many-to-One* s enitetom KLINIKA preko atributa adresa1, u vezi *Many-to-One* s enitetom SMJESTAJ preko atributa adresa2.

PUTOVANJE		
IDPutovanje	INT	Identifikacijski ključ putovanja
vrijeme	TIME	Vrijeme putovanja
smjer	INT	Smjer u kojem se putovanje izvodi
adresa1	VARCHAR	Adresa klinike
adresa2	VARCHAR	Adresa smještaja
IDKorisnik	INT	Identifikacijski ključ korisnika
IDKlinika	INT	Identifikacijski ključ klinike
IDPrijevoznik	INT	Identifikacijski ključ prijevoznika
IDSmjestaj	INT	Identifikacijski ključ smještaja

4.1.2 Dijagram baze podataka

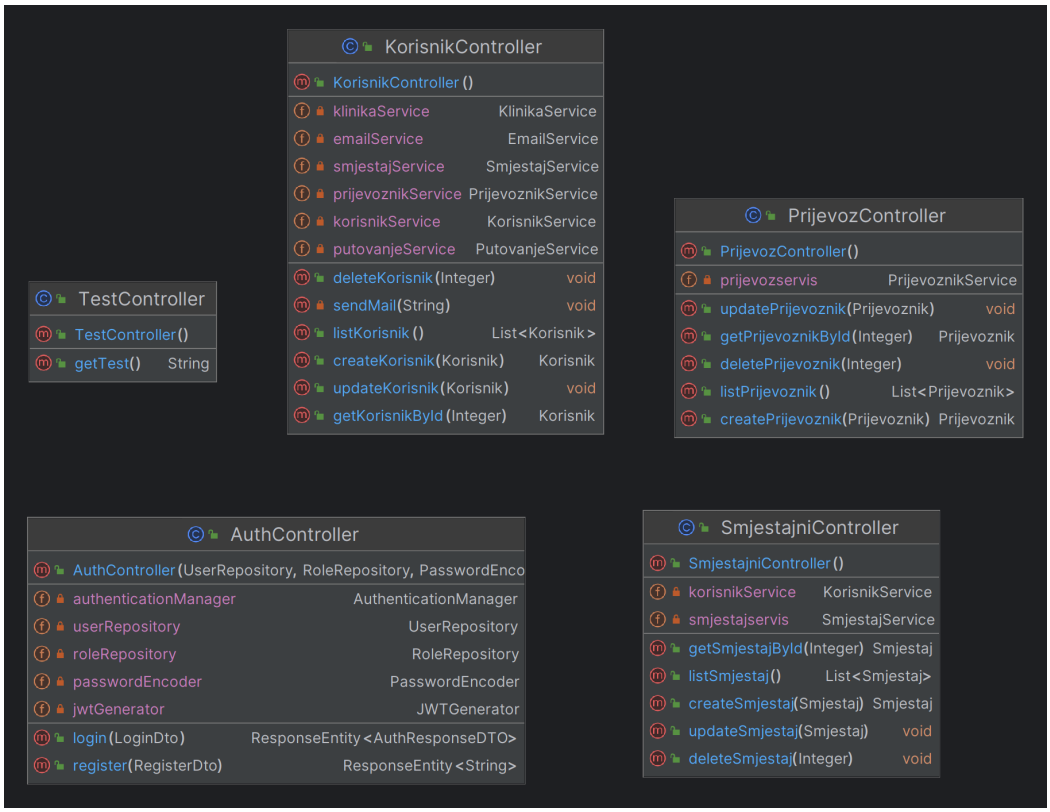


Slika 4.1: Relacijska shema baze podataka

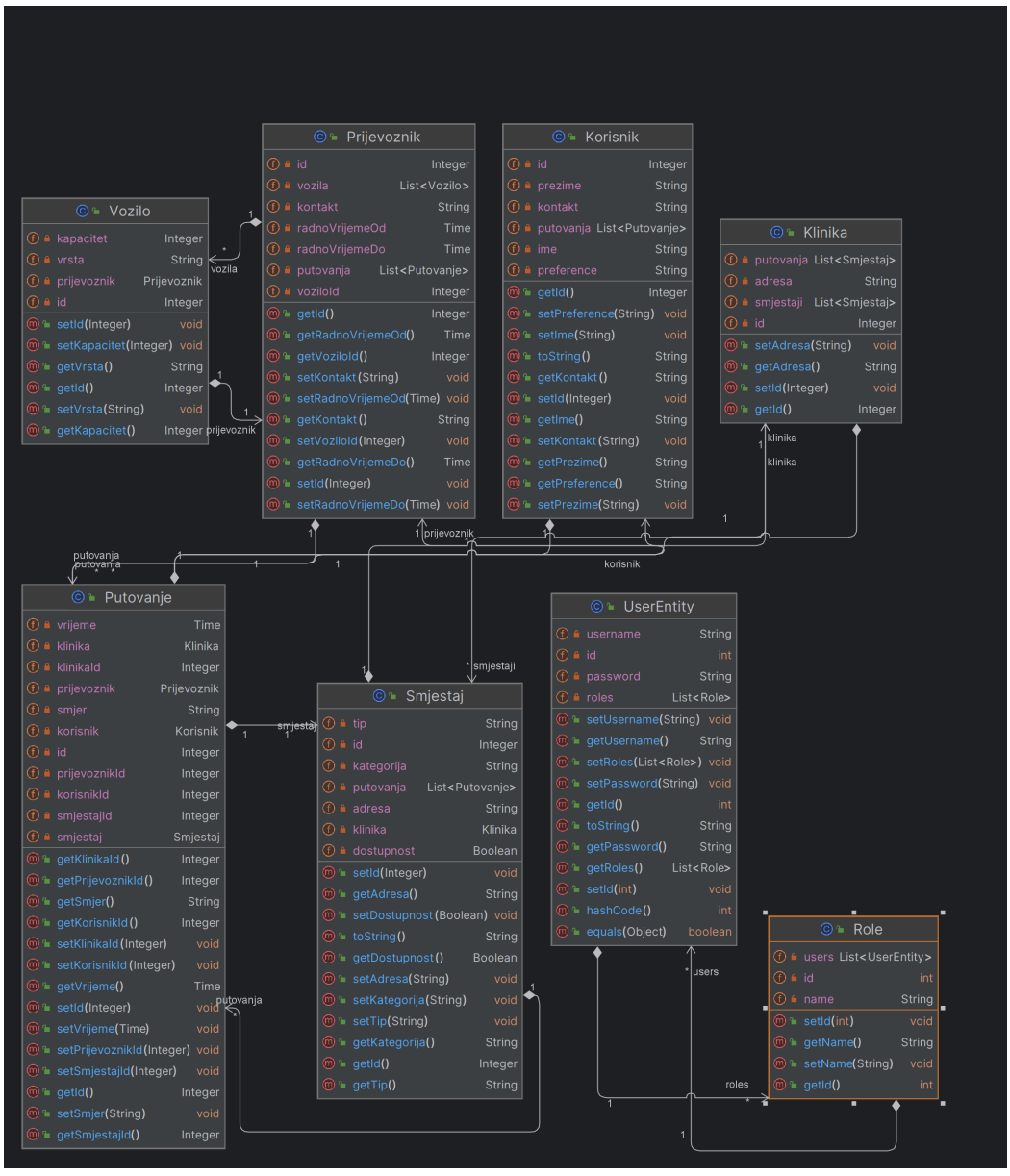
4.2 Dijagram razreda

Slika 4.2 prikazuje razred *AuthController* koji služi prihvaćanju HTTP zahtjeva od strane klijenta i to specifično za URL */auth/**. Metode *login()* i *register()* služe kao URL-ovi */auth/login* i */auth/register* na koje se šalju JSON objekti za prijavu administratora i registraciju novog administratora. Prikazan su i razredi: *PrijevozController*, *KorisnikController*, *PrijevozController* i *SmjestajniController*. Metode unutar tih razreda izvršavaju manipulacije nad modelima i vraćaju tražene informacije koje su predstavljene modelima, obično u obliku listi podataka.

Modeli su prikazani slikom 4.3. Razredi modela preslikavaju strukturu baze podataka u okviru aplikacije. Razredi odgovaraju entitetima iz baze podataka.

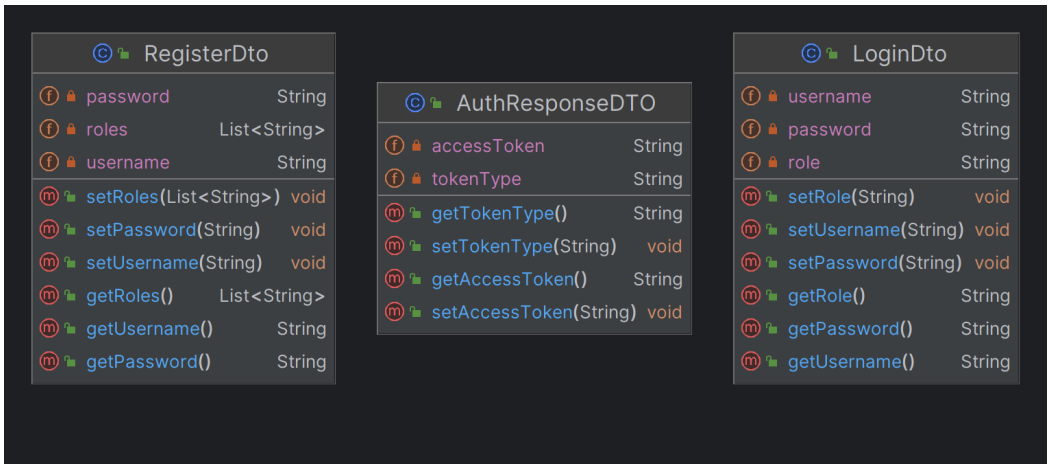


Slika 4.2: *Controllers*



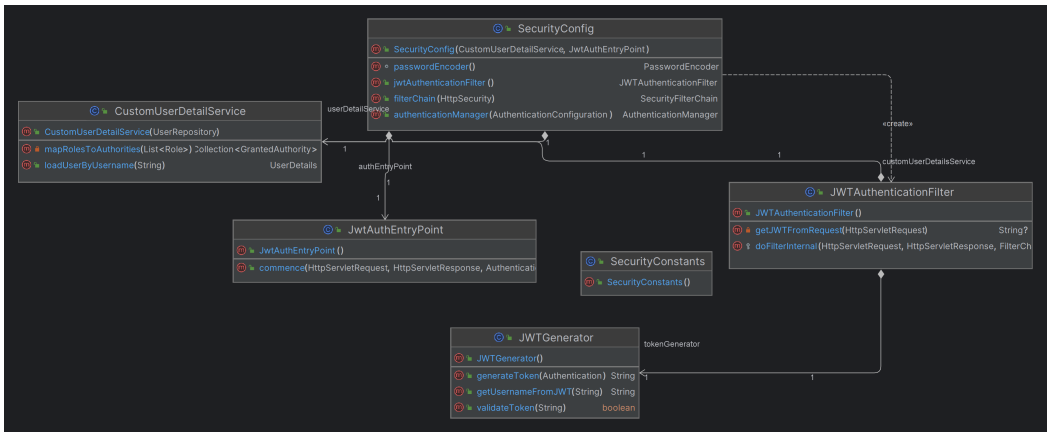
Slika 4.3: UML dijagram paketa *Models*

Slika 4.4 prikazuje paket DTO koji služi za pretvaranje JSON objekata koji stižu na određenu rutu i Java objekt i za pretvaranje Java objekata u JSON odgovore koje klijent razumije. Između razreda Controllers i DTO ne postoje veze na dijagramima.



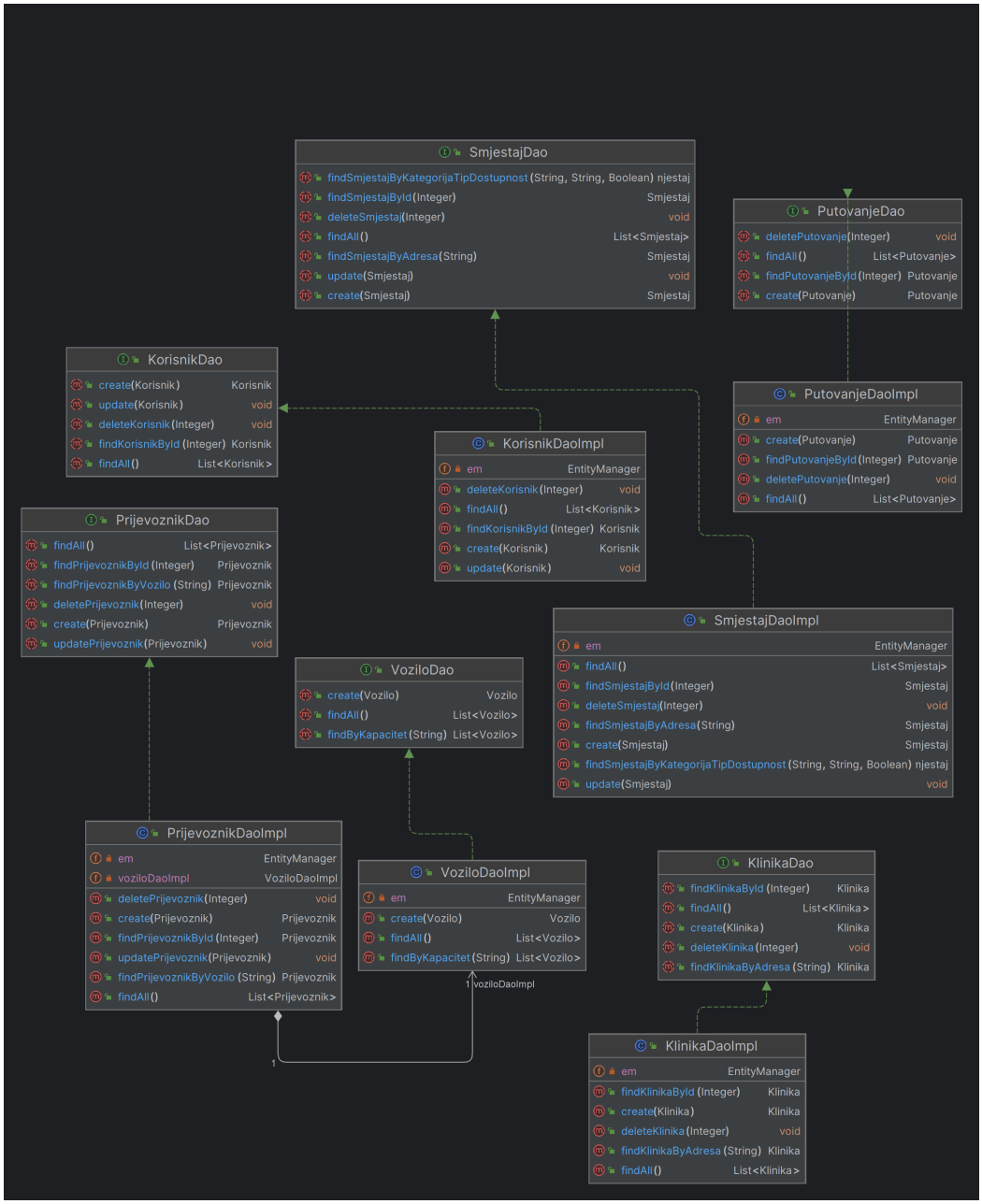
Slika 4.4: *DTO*

Slika 4.5 prikazuje paket *Security* koji je zadužen za obradu svakog zahtjeva koji stiže na server i prosuditi ima li trenutni korisnik pravo pristupa. Glavna klasa za to je klasa *SecurityConfig* koja preko metode *filterChain()* primjenjuje filtre na svaki zahtjev da odredi pravo pristupa. Također klasa *SecurityConfig* pomoću klasa *JWTGenerator, JWTAuthenticationFilter i JwtAuthEntryPoint* za svaku uspješnu prijavu generira JWT token koji se zatim u svim zahtjevima tog korisnika koristi za autentifikaciju tog korisnika.



Slika 4.5: UML dijagram paketa *Security*

Slika 4.5 prikazuje paket *DAO*. Paket *DAO* ima odgovornost za pristup podacima u sustavu i interakciju s bazom podataka. Implementacije razreda unutar ovog paketa, kao što su *PutovanjeDaoImpl*, *SmjestajDaoImpl*, *KlinikaDaoImpl*, *VoziloDaoImpl*, *PrijevoznikDaoImpl* i *KorisnikDaoImpl*, služe za implementaciju funkcija koje upravljaju bazom podataka. Ovi razredi omogućavaju izvođenje operacija čitanja, pisanja, ažuriranja i brisanja podataka u odgovarajućim entitetima sustava.



Slika 4.6: UML dijagram paketa *DAO*

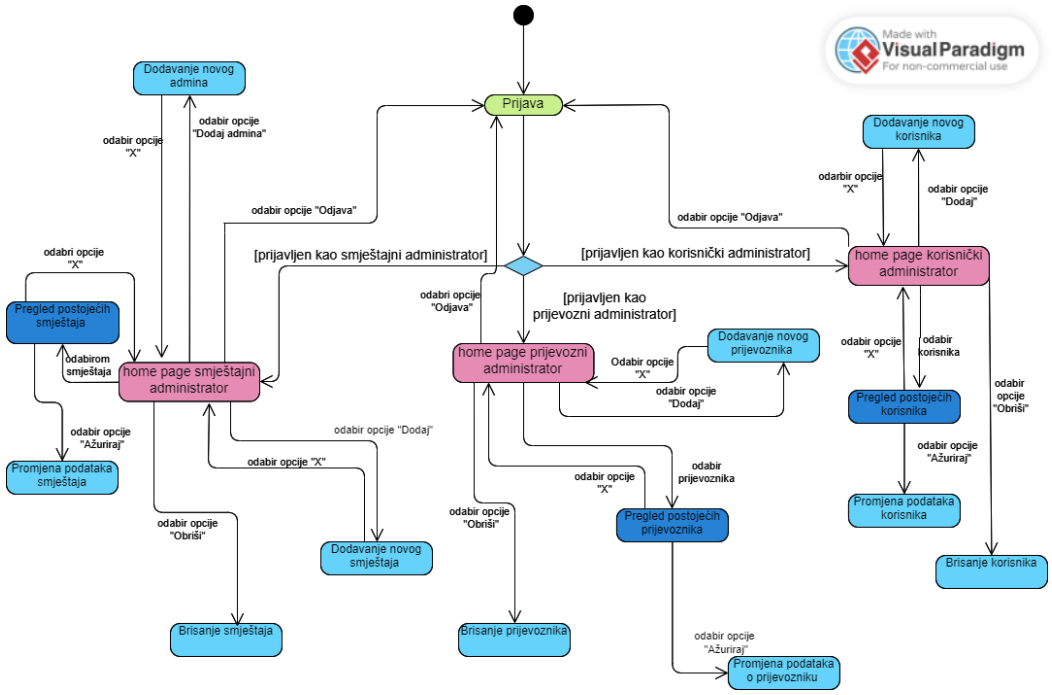
4.3 Dijagram stanja

Na slici 4.7 prikazan je dijagram stanja. Dijagram stanja prikazuje u kojem se stanju korisnik aplikacije može nalaziti. Početno stanje je prijava. Korisnik se može prijaviti kao jedan od administratora: Korisnički administrator, Smještajni administrator ili Prijevozni administrator. Ovisno o tome koji administrator se prijavi, odlazi na odgovarajuću početnu stranicu (homepage) gdje se nalaze opcije za korištenje ovisno o ulozi. Svi administratori mogu se vratiti na prijavu tako što se prethodno odjave, klikom ”Odjava”.

Ako je prijavljeni administrator smještajni administrator, ima opciju dodavanja novih administratora odabirom opcije ”Dodaj admina”. Također, ima mogućnost pregleda smještaja i ažuriranja istog pomoću opcije ”Ažuriraj”. Već postojeći smještaj administrator može obrisati opcijom ”Obriši”. Novi smještaj dodaje opcijom ”Dodaj”.

Ako je prijavljeni administrator prijevozni administrator, ima opciju pregleda već postojećih prijevoznika te ažuriranja istih opcijom ”Ažuriraj”. Također, prijevozni administrator može dodavati nove prijevoznike opcijom ”Dodaj” i brisati već postojeće opcijom ”Obriši”.

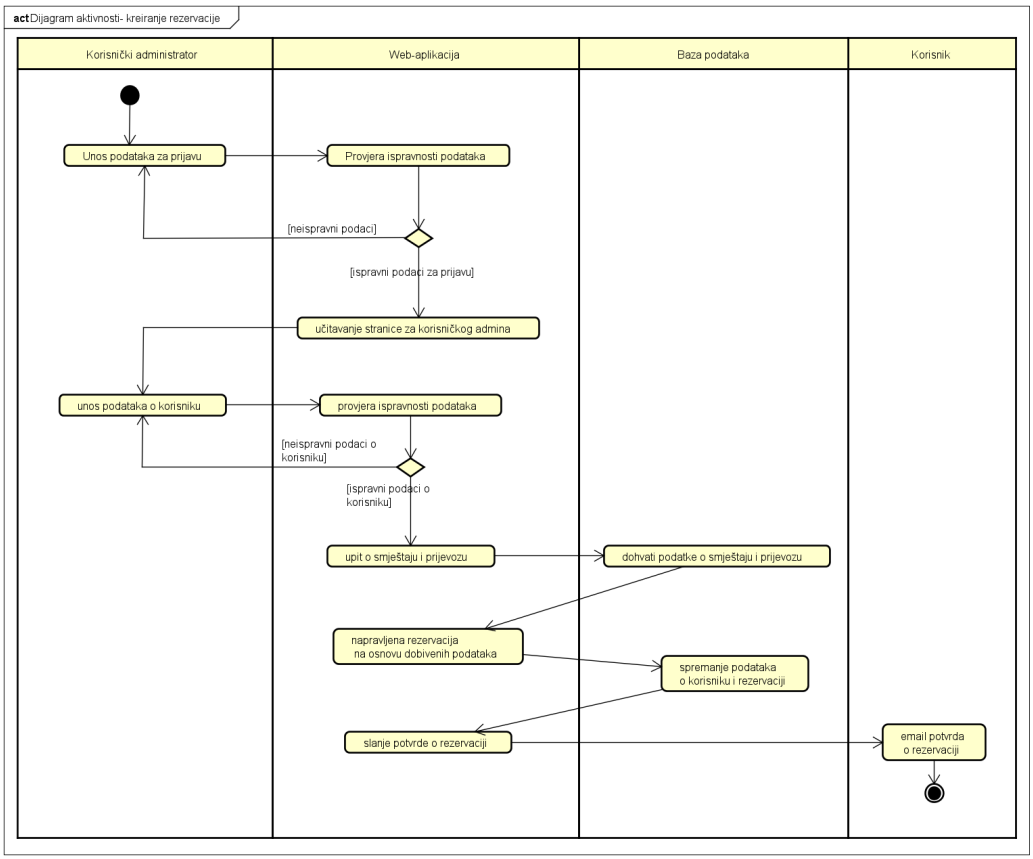
Ako je prijavljeni administrator korisnički administrator, ima opciju dodavanja novog korisnika (klijenta) opcijom ”Dodaj”. Također, odabirom korisnika mogu se pregledavati njegovi podaci te odabirom opcije ”Ažuriraj” ažurirati. Korisnike se može brisati opcijom ”Obriši”.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

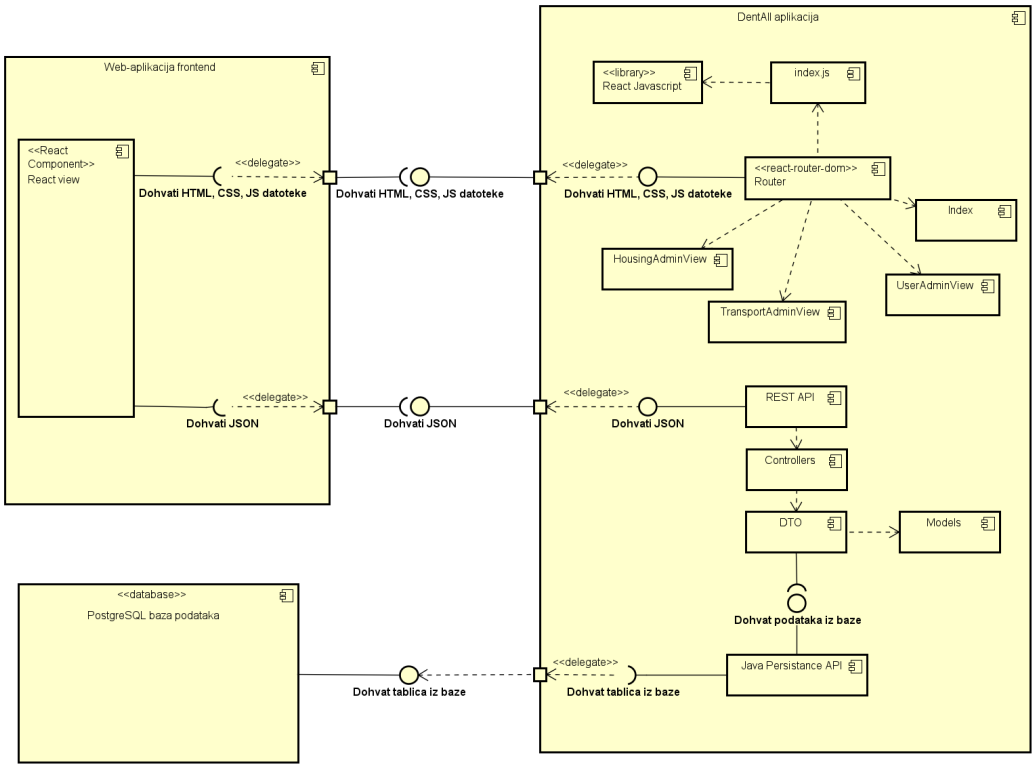
Na slici 4.8 prikazan je dijagram aktivnosti. Dijagram aktivnosti ilustrira detalje procesa kreiranja rezervacije za korisnika putem web-aplikacije. Korisnički administrator započinje postupak unosom ispravnih podataka za prijavu u web-aplikaciju. Ukoliko se uneseni podaci pokažu neispravnima, administrator će ponovno unijeti potrebne informacije kako bi uspješno pristupio aplikaciji. Nakon uspješne prijave administratoru se učitava stranica, a zatim korisnički administrator unosi podatke o korisniku za kojeg želi napraviti rezervaciju. Web-aplikacija provodi provjeru ispravnosti unesenih podataka, osiguravajući da su svi potrebni detalji upisani. Kada su svi podaci uneseni ispravno, aplikacija nastavlja s provjerom dostupnosti smještaja i prijevoza za odabranog korisnika. Na temelju rezultata provjere dostupnosti, web-aplikacija generira rezervaciju za korisnika. Baza podataka se ažurira označavajući odabrani smještaj i prijevoz kao zauzet u određenom razdoblju rezerviranom za korisnika. Nakon spremanja podataka, Web-aplikacija šalje e-mail potvrdu korisniku o potvrdi rezervacije.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Na slici 4.9 prikazan je dijagram komponenti. Za dohvaćanje HTML, CSS i JS datoteka na klijentsku stranu (*f*rontend) koristi se prvo od dva sučelja. Ovisno o potrebnom prikazu (HousingAdminView, TransportAdminView, UserAdminView, Index), Router komponenta određuje koje se HTML, CSS i JS datoteke poslužuju na prvom sučelju. Sučelju za primanje JSON podataka pristupa se putem REST API komponenti. REST API pruža podatke koji pripadaju serverskoj strani aplikacije (*b*ackend). Java Persistence API omogućava dohvaćanje podataka iz baze generirajući SQL upite te upisivanje u bazu. Na serverskoj strani implementiramo kontrolere (Controllers) kako bismo prenijeli modele, pretvorene u DTO (Data Transfer Object), prema klijentskoj strani dijela aplikacije.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija unutar tima ostvarena je korištenjem aplikacije Whatsapp[1], dok je za direktnu interakciju s asistentom grupe odabran Microsoft Teams[2]. Za upravljanje verzijama koda i suradnju u razvoju, korišteni su Git[3] i GitHub[4] na kojemu se nalazi udaljeni repozitorij. U izradi UML dijagrama korišteni su alati Astah UML[5] i Visual Paradigm Online[6]. Za oblikovanje dijagrama baze podataka korišten je alat DBDiagram[7].

Stranica klijenta je razvijena korištenjem programskog jezika JavaScript[8] i React[9] biblioteke, a korišteno je razvojno okruženje Visual Studio Code[10]. Za server stranu korišteni su programski jezik Java[11] i radni okvir Spring Boot[12], a korišteno je razvojno okruženje Intellij IDEA[13]. Za puštanje aplikacije u pogon korištena je platforma Vercel[14] i Render[15]. Ispitivanje sustava provedeno je koristeći radni okvir Selenium[16], a ispitivanje komponenti provedeno je pomoću radni okvir JUnit[17].

[1] https://www.whatsapp.com/
[2] https://www.microsoft.com/hr-hr/microsoft-teams/group-chat-software
[3] https://git-scm.com/
[4] https://github.com/
[5] https://astah.net/products/astah-uml/
[6] https://online.visual-paradigm.com/
[7] https://dbdiagram.io/home
[8] https://www.javascript.com/
[9] https://react.dev/
[10] https://code.visualstudio.com/
[11] https://www.java.com/en/
[12] https://spring.io/projects/spring-boot/
[13] https://www.jetbrains.com/idea/
[14] https://vercel.com
[15] https://render.com/
[16] https://www.selenium.dev/
[17] https://junit.org/junit5/

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Za ispitivanje komponenti korišten je radni okvir JUnit5. Ispitane su komponente *AuthController* i *KorisnikController* jer su te dvije komponente najkritičnije komponente sustava. Komponenta *AuthController* zadužena je za funkcionalnost prijave i registracije novih administratora. U nastavku slijedi kod svih JUnit testova *AuthController* komponente. Opis funkcionalnosti svakog testa je u komentarima. Nažalost paket za prikaz koda ne podržava dijakritičke znakove pa su na mjestima zamijenjeni svojim ne dijakritičnim oblikom.

```
1       package com.dental.controllers;
2
3       import com.dental.DentAllApplication;
4       import com.dental.dto.LoginDto;
5       import com.dental.dto.RegisterDto;
6       import jakarta.transaction.Transactional;
7       import org.junit.jupiter.api.Test;
8       import org.springframework.beans.factory.annotation.Autowired;
9       import org.springframework.boot.test.autoconfigure.orm.jpa.
    DataJpaTest;
10      import org.springframework.boot.test.context.SpringBootTest;
11      import org.springframework.http.HttpStatus;
12      import org.springframework.http.ResponseEntity;
13      import org.springframework.test.context.ActiveProfiles;
14      import org.springframework.test.context.junit4.SpringRunner;
15      import org.springframework.transaction.annotation.Propagation;
16
17      import java.util.ArrayList;
18      import java.util.List;
19
20      import static org.assertj.core.api.Assertions.*;
21
22      @SpringBootTest
23      @ActiveProfiles("test")
24      public class AuthControllerTests {
25
26        // Omogucava SpringBoot-u da pristupi AuthController objektu
27        @Autowired
28        AuthController controller;
29
```

```
30          // Test koji provjerava je li kontroler pravilno ucitan u
    memoriju. Sluzi kao svojevrstan "sanity-check"
31          @Test
32          public void contextLoads() {
33            assertThat(controller).isNotNull();
34          }
35
36          // Najjednostavniji test koji provjerava je li sve u redu ako
    u controller posaljemo tocne i dobro formatirane podatke.
37          @Test
38          public void registerAllValuesProvided() {
39            List<String> roles = new ArrayList<>();
40            roles.add("sleep_admin");
41
42            assertThat(controller.register(new RegisterDto("new_user", "
    user_pass", roles))).isEqualTo(new ResponseEntity<>("User
    successfully added to db", HttpStatus.OK));
43          }
44
45          // Test koji provjerava reagira li sustav dobro na slucaj ako
    korisnicko ime vec postoji u sustavu tj. vraca li tocnu gresku
46          @Test
47          public void registerWithUsernameExists() {
48            List<String> roles = new ArrayList<>();
49            roles.add("sleep_admin");
50
51            controller.register(new RegisterDto("username", "password",
    roles));
52
53            assertThat(controller.register(new RegisterDto("username", "
    password", roles))).isEqualTo(new ResponseEntity<>("Error: Username
    already exists", HttpStatus.BAD_REQUEST));
54          }
55
56          // Test koji provjerava da nije moguce registrirati korisnika
    bez dodijeljenih rola
57          @Test
58          public void registerWithNoRoles() {
59            assertThat(controller.register(new RegisterDto("
    user_no_roles", "user_pass", new ArrayList<>()))).isEqualTo(new
    ResponseEntity<>("Error: No role selected", HttpStatus.BAD_REQUEST));
60          }
61
```

```
62          // Najjednostavniji test prijave u kojoj se ruti za prijavu
    salju tocni podatci koji postoje u bazi podataka
63          @Test
64          public void loginWithProperCreds() {
65            List<String> roles = new ArrayList<>();
66            roles.add("sleep_admin");
67
68            controller.register(new RegisterDto("username", "password",
    roles));
69
70            assertThat(controller.login(new LoginDto("username", "
    password", "sleep_admin")).getStatusCode()).isEqualTo(HttpStatus.OK);
71          }
72
73          // Test koji provjerava reagira li sustav dobro na pogresnu
    lozinku
74          @Test
75          public void loginWithWrongPassword() {
76            List<String> roles = new ArrayList<>();
77            roles.add("sleep_admin");
78
79            controller.register(new RegisterDto("username", "password",
    roles));
80
81            assertThat(controller.login(new LoginDto("username", "pass",
     "sleep_admin")).getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
82          }
83
84          // Test koji provjerava reagira li sustav dobro na pogresno
    korisnicko ime
85          @Test
86          public void loginWithWrongUsername() {
87            List<String> roles = new ArrayList<>();
88            roles.add("sleep_admin");
89
90            controller.register(new RegisterDto("username", "password",
    roles));
91
92            assertThat(controller.login(new LoginDto("user", "password",
     "sleep_admin")).getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
93          }
94
```

```
95          // Test koji provjerava reagira li sustav dobro na tocno ime i
    lozinku, ali pogresnu rolu
96          @Test
97          public void loginWithWrongRole() {
98            List<String> roles = new ArrayList<>();
99            roles.add("sleep_admin");
100
101           controller.register(new RegisterDto("username", "password",
    roles));
102
103           assertThat(controller.login(new LoginDto("username", "
    password", "user_admin")).getStatusCode()).isEqualTo(HttpStatus.
    BAD_REQUEST);
104         }
105
106         // Test koji provjerava reagira li sustav dobro na rolu koju
    nije moguce imati
107         @Test
108         public void loginWithImpossibleRole() {
109           List<String> roles = new ArrayList<>();
110           roles.add("sleep_admin");
111
112           controller.register(new RegisterDto("username", "password",
    roles));
113
114           assertThat(controller.login(new LoginDto("username", "
    password", "impossible_admin")).getStatusCode()).isEqualTo(HttpStatus
    .BAD_REQUEST);
115         }
116       }
117
```

Listing 5.1: JUnit testovi za AuthController

Komponenta *KorisnikController* zadužena je za funkcionalnost stvaranja, mijenjanja i brisanja podataka o korisnicima. Najvažniji dio funkcionalnosti ove komponente je ruta zadužena za stvaranje korisnika jer se u njoj odvija logika za stvaranje putovanje i za slanje poruka e-pošte korisnicima koje se unese u sustav. U nastavku slijede JUnit testovi za *KorisnikController*.

```
1       package com.dental.controllers;
2
3       import com.dental.dao.PutovanjeDaoImpl;
4       import com.dental.dao.SmjestajDaoImpl;
```

```
5         import com.dental.models.Korisnik;
6         import com.dental.models.Prijevoznik;
7         import com.dental.models.Putovanje;
8         import com.dental.models.Smjestaj;
9         import org.checkerframework.checker.units.qual.A;
10        import org.junit.jupiter.api.Test;
11        import org.springframework.beans.factory.annotation.Autowired;
12        import org.springframework.boot.test.context.SpringBootTest;
13        import org.springframework.http.HttpStatus;
14        import org.springframework.security.core.parameters.P;
15        import org.springframework.test.context.ActiveProfiles;
16
17        import java.sql.Time;
18
19        import static org.assertj.core.api.Assertions.*;
20
21        @SpringBootTest
22        @ActiveProfiles("test")
23        public class KorisnikControllerTest {
24
25          @Autowired
26          KorisnikController controller;
27
28          @Autowired
29          PrijevozController prijevozController;
30
31          @Autowired
32          SmjestajniController smjestajniController;
33
34          @Autowired
35          PutovanjeDaoImpl putovanjeDao;
36
37          @Autowired
38          SmjestajDaoImpl smjestajDao;
39
40          @Test
41          public void contextLoaded() {
42            assertThat(controller).isNotNull();
43          }
44
45          // Test koji provjerava dodaje li se korisnik s tocnim
    pocetnim preferencama ako nisu specificirane neke nove
46        @Test
```

```
47          public void createKorisnikDefaultPreferencesStatus() {
48            prijevozController.createPrijevoznik(new Prijevoznik(null, "
    kontak@prijevoz.com", new Time(8, 0, 0), new Time(15, 0, 0), "auto",
    4, "Opel Ad Astra"));
49            smjestajniController.createSmjestaj(new Smjestaj("stan", "1"
    , "Baker Street 221B", true));
50
51            HttpStatus status = controller.createKorisnik(new Korisnik("
    Brad", "Pitt", "", "pitt@brad.com"));
52            assertThat(status).isEqualTo(HttpStatus.OK);
53          }
54
55          // Test koji provjerava stvara li se dobro putavnje tj. salje
    li se tocan mail korisniku koji nije specificirao nikakve nove
    preference
56          @Test
57          public void createKorisnikDefaultPreferencesPutovanje() {
58            prijevozController.createPrijevoznik(new Prijevoznik(null, "
    kontak@prijevoz.com", new Time(8, 0, 0), new Time(15, 0, 0), "auto",
    4, "Opel Ad Astra"));
59            smjestajniController.createSmjestaj(new Smjestaj("kuca", "1"
    , "Baker Street 221B", true));
60            smjestajniController.createSmjestaj(new Smjestaj("stan", "1"
    , "Baker Street 221B", true));
61
62            HttpStatus status = controller.createKorisnik(new Korisnik("
    Brad", "Pitt", "", "pitt@brad.com"));
63            Putovanje p = putovanjeDao.findPutovanjeById(1);
64            assertThat(status).isEqualTo(HttpStatus.OK);
65            assertThat(p.getPrijevoznikId()).isEqualTo(1);
66            assertThat(p.getSmjestajId()).isEqualTo(2);
67          }
68
69          // Test koji provjerava stvaranje korisnika s preferencama
70          @Test
71          public void createKorisnikWithAPreference() {
72            prijevozController.createPrijevoznik(new Prijevoznik(null, "
    kontak@prijevoz.com", new Time(8, 0, 0), new Time(15, 0, 0), "auto",
    4, "Opel Ad Astra"));
73            smjestajniController.createSmjestaj(new Smjestaj("kuca", "1"
    , "Baker Street 221B", true));
74            smjestajniController.createSmjestaj(new Smjestaj("stan", "1"
    , "Baker Street 221B", true));
```

```
75
76          HttpStatus status = controller.createKorisnik(new Korisnik("
    Brad", "Pitt", "tip:kuca", "pitt@brad.com"));
77          Putovanje p = putovanjeDao.findPutovanjeById(3);
78
79          assertThat(status).isEqualTo(HttpStatus.OK);
80          assertThat(p.getPrijevoznikId()).isEqualTo(1);
81          assertThat(p.getSmjestajId()).isEqualTo(1);
82        }
83
84        // Test koji provjerava moze li se dohvatiti korisnik putem id
    -a
85        @Test
86        public void getKorisnikById() {
87          HttpStatus status = controller.getKorisnikById(1);
88          assertThat(status).isEqualTo(HttpStatus.OK);
89        }
90
91        // Test koji provjerava odgovara li sustav dobro na id koji ne
     postoji u bazi
92        @Test
93        public void getKorisnikByInvalidId() {
94          HttpStatus status = controller.getKorisnikById(-1);
95          assertThat(status).isEqualTo(HttpStatus.BAD_REQUEST);
96        }
97      }
98
```

Ako navedene testove pokrenemo u našem razvojom okruženju koristeći naredbu *mvn test* vidjet ćemo da sustav pravilno reagira i prolazi sve testove.

```
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.179 s - in com.dental.controllers.KorisnikC
ontrollerTest
[INFO] Running com.dental.controllers.AuthControllerTests
2024-01-19T00:32:38.005+01:00  INFO 244868 --- [           main] t.c.s.AnnotationConfigContextLoaderUtils : Could no
t detect default configuration classes for test class [com.dental.controllers.AuthControllerTests]: AuthControllerTe
sts does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
2024-01-19T00:32:38.007+01:00  INFO 244868 --- [           main] .b.t.c.SpringBootTestContextBootstrapper : Found @S
pringBootConfiguration com.dental.DentAllApplication for test class com.dental.controllers.AuthControllerTests
2024-01-19T00:32:38.353+01:00 DEBUG 244868 --- [           main] o.s.s.a.dao.DaoAuthenticationProvider    : Failed t
o authenticate since password does not match stored value
2024-01-19T00:32:38.454+01:00 DEBUG 244868 --- [           main] o.s.s.a.dao.DaoAuthenticationProvider    : Authenti
cated user
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.579 s - in com.dental.controllers.AuthContr
ollerTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 16, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.734 s
[INFO] Finished at: 2024-01-19T00:32:38+01:00
[INFO] ------------------------------------------------------------------------
```

Slika 5.1: Uspješno izvršeni svi JUnit testovi

5.2.2 Ispitivanje sustava

Za provođenje ispitivanja sustava koristili smo radni okvir Selenium u programskom jeziku Python3. Za pokretač Selenium-a korišten je Geckodriver[18] tj. web preglednik Mozilla Firefox[19]. Svih pet testova koje smo proveli se provode odjednom kako bi sustav bio pod maksimalnim opterećenjem i kako bi se osiguralo da sustav dobro prati stanje između izvršavanih akcija.

Test 1 - prijava administratora u sustavi (UC1/UC2/UC3)

- **Ulazni podatci:** korisničko ime, lozinka i uloga
- **Očekivani izlaz:** Administratora se navigira na stranicu veznu za ulogu koju je odabrao ako su podatci točni i od servera prima token koji mu omogućuje pristup sustavu
- **Koraci ispitivanja:**
  1. Otvaranje početne stranice
  2. Upisivanje svih potrebnih podataka i odabir uloge
  3. Pritisak na gumb za prijavu u sustav
- **Rezultati ispitivanja:** Administrator je ispravno navigiran u pripadajuću

[18] https://github.com/mozilla/geckodriver/releases
[19] https://www.mozilla.org/en-US/firefox/new/

stranicu i dodijeljen mu je token ako upiše točne podatke, a ako upiše krive sustav mu javlja poruku da neki od ulaznih podataka nisu točni.

Test 2 - Dodavanje novog prijevoznika u sustav (UC10)

- **Ulazni podatci:** email, radno vrijeme od, radno vrijeme do, vrsta, kapacitet, model
- **Očekivani izlaz:** U listu prijevoznika je dodan novi prijevoznik s točnim podacima
- **Koraci ispitivanja:**
  1. Navigacija na rutu /transport
  2. Pritisak na gumb dodaj
  3. Upisivanje svim potrebnih podataka u formu
  4. Pritisak na gumb za dodavanje prijevoznika
- **Rezultati ispitivanja:** Forma se uspješno pojavila i prijevoznik je uspješno dodan u sustav.

Test 3 - Dodavanje novog smještaja u sustav (UC6)

- **Ulazni podatci:** adresa, tip, kategorija, dostupnost
- **Očekivani izlaz:** U listu smještaja je dodan novi smještaj s točnim podacima i prikazan je na karti
- **Koraci ispitivanja:**
  1. Navigacija na rutu /housing
  2. Pritisak na gumb dodaj
  3. Upisivanje svim potrebnih podataka u formu
  4. Pritisak na gumb za dodavanje smještaja
- **Rezultati ispitivanja:** Forma se uspješno pojavila i smještaj je dodan s točnim podacima i uspješno se prikazuje na karti

Test 4 - Dodavanje novog klijenta u sustav (UC13)

- **Ulazni podatci:** ime, prezime, preferencije, email, vrijeme dolaska i vrijeme odlaska
- **Očekivani izlaz:** U listu korisnika dodan je novi korisnik i poslana je poruke e-pošte klijentu i prijevozniku
- **Koraci ispitivanja:**
  1. Navigacija na rutu /users
  2. Pritisak na gumb dodaj

3. Upisivanje svim potrebnih podataka u formu
4. Pritisak na gumb za dodavanje korisnika

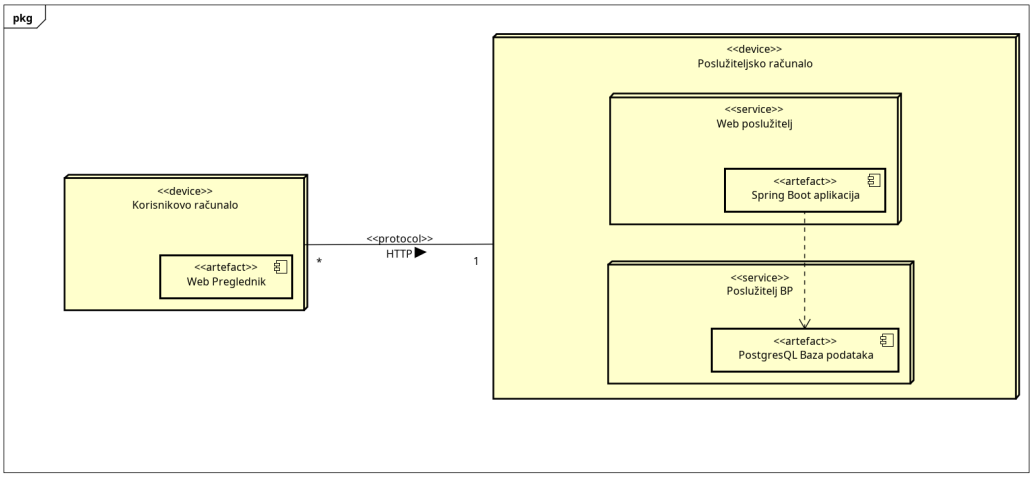
- **Rezultati ispitivanja:** Forma se uspješno pojavila i korisnik je dodan s točnim podacima i uspješno su poslane poruke e-pošte klijentu i prijevozniku.

Test 5 - Dodavanje novog administratora u sustav i prijava kao isti (UC4)

- **Ulazni podatci:** korisničko ime, lozinka i uloge novog administratora
- **Očekivani izlaz:** Moguća je odjava iz smještajnog administratora i prijava kao novo dodani administrator
- **Koraci ispitivanja:**
  1. Navigacija na rutu /housing
  2. Pritisak na gumb dodaj admina
  3. Upisivanje svim potrebnih podataka u formu
  4. Pritisak na gumb za dodavanje
  5. Pritisak na gumb odjava
  6. Upisivanje informacija za prijavu novog administratora
  7. Prijava u sustav novog administratora
- **Rezultati ispitivanja:** Administrator je uspješno dodan u sustav, odjava radi točno (navigacija na početnu stranicu i brisanje tokena) i moguće se odmah prijaviti kao novi administrator

5.3 Dijagram razmještaja

Na poslužiteljskom računalu pokreće se web aplikacija (Spring Boot) u kontekstu JRE-a i PostgresQL poslužitelj baze podataka. Klijenti web aplikaciji mogu pristupiti pomoću web preglednika. Sustav koristi klijent-server arhitekturu gdje klijent šalje HTTP zahtjeve na koje poslužitelj odgovara HTTP odgovorima koji podatke prenose u JSON formatu.



Slika 5.2: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Za puštanje aplikacije u pogon korištene su platforme Vercel[20] i Render[21]. Vercel je korišten za puštanje u pogon web stranice kojoj pristupaju administratori. Korišten je jer omogućuje besplatno puštanje u pogon za male aplikacije, a ima i odličnu integraciju s bibliotekom React[22] i sustavom Vite[23] koji je korišten za izradu React projekta. Render je korišten za puštanje u pogon web poslužitelja i baze podataka jer kao i Vercel nude besplatan plan za male aplikacije. Render također podržava PostgresQL baze podataka direktno bez dodatne konfiguracije, a web poslužitelj koji je pisan u razvojnom okviru Spring Boot bilo je potrebno upakirati u Docker[24] kontejner kako bi ga Render mogao pustiti u pogon.

Priprema Docker kontejnera sa Spring Boot aplikacijom

Tehnologija Docker omogućuje stvaranje takozvanih kontejnera koji unutar sebe imaju sve potrebno za pokretanje aplikacije, a mogu se pokrenuti iz jedinstvenog Docker sučelja. Docker postiže ovu funkcionalnosti virtualizacijom operacijskog sustava Linux uz pomoć posebne funkcionalnosti jezgre operacijskog sustava Linux koju su preuzeli i Windows i Mac OS. Docker kontejnere opisujemo pomoću posebne datoteke koja se zove Dockerfile. U Dockerfile-u našeg sustava opisujemo Docker-u kako da izgradi i pokrene našu aplikaciju.

[20] https://vercel.com
[21] https://render.com/
[22] https://react.dev/
[23] https://vitejs.dev/
[24] https://www.docker.com/

```
1
2     # Za poceni kontejner uzimamo Ubuntu kontejner koji dolazi s JDK
  17
3     FROM eclipse-temurin:17-jdk-jammy AS build
4     # Maven je sustav za upravljanje paketima i  stvaranje Spring Boot
   projekta
5     RUN apt-get update && apt-get install -y maven
6     # Prebacivanje svih datoteka nase Spring Boot aplikacije u
  datotecni sustav kontejnera
7     COPY . .
8     # Maven naredba za prevodenje koda u Java Bytecode koji se moze
  izvesti na JVM-u
9     RUN mvn clean package -Pprod -DskipTests
10
11
12
13    # Pokretanje nase aplikacije
14    FROM openjdk:17-jdk-slim
15    COPY --from=build /target/DentAll-0.0.1-SNAPSHOT.jar DentAll.jar
16    # Tomcat (web posluzitelj koji Spring Boot koristi) se pokrece na
  vratima 8080
17    EXPOSE 8080
18    ENTRYPOINT ["java","-jar","DentAll.jar"]
19
```

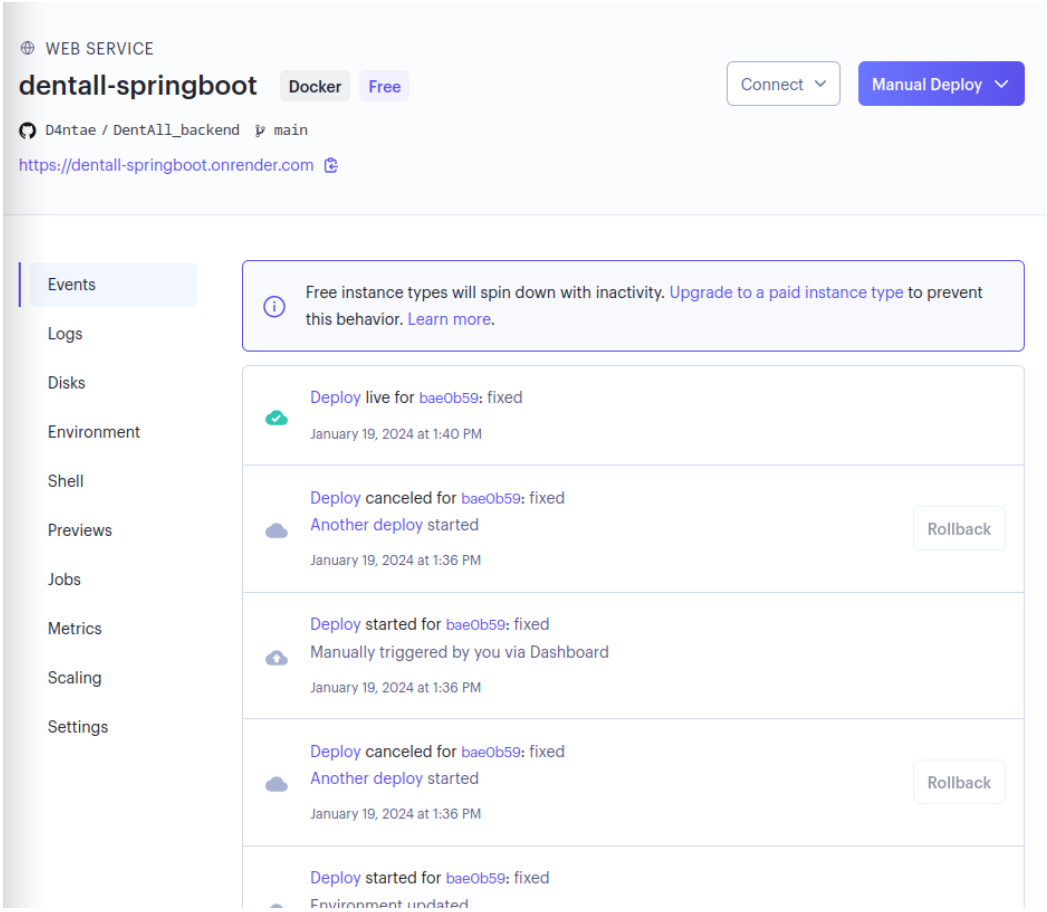
Baza podataka

Baza podataka je najlakša od svih triju komponenata sustava jer sve što je potrebno je na platformi Render odabrati instancu Postgres baze, odabrati korisnika i ime baze podataka unutra Postgres poslužitelja i Render će se pobrinuti za sve ostalo. Poslužitelj baze podataka može se pronaći na URL-u: `dpg-cml5snacn0vc739nhukg-a.frankfurt-postgres.render.com`.

Web poslužitelj

Kako bismo mogli pustiti web poslužitelj u pogon na platformi Render potrebno je odabrati mogućnost *Web service*. Pošto platforma Render podržava Docker kontejnere kao način puštanja u pogon našu smo aplikaciju po prijašnjim uputama umotali u Docker kontejner. Također smo u datoteci *application.properties* koju Spring Boot koristi kao konfiguracijsku datoteku promijenili podatke o bazi podataka tako da usmjerimo Spring Boot na našu novo nastalu Postgres bazu na Render-u. Nakon što imamo imamo Dockerfile napisan za našu aplikaciju samo je potrebno us-

mjeriti Render na git repozitorij s Dockerfile-om i Spring Boot aplikacijom. Za git poslužitelj na kojem se nalazim naš repozitorij odabrali smo GitHub. Nakon što je Render obratio Dockerfile potrebno je u kartici *Environment* definirati vrata na kojima se pokreće naša aplikacija pošto Render prima zahtjeve samo na vratima 443 potrebno je preusmjeriti te zahtjeve na vrata 8080 na kojima se pokreće Spring Boot. Naš web poslužitelj pokrenut je na `https://dentall-springboot.onrender.com`



Slika 5.3: Prikaz web poslužitelja puštenog u pogon

Klijent

S obzirom na to da smo za izradu React projekta koristili sustav Vite platforma Vercel može pustiti u pogon našu web aplikaciju tako da joj samo damo link na git repozitorij koji u sebi sadrži Vite projekt. Ponovno smo za poslužitelja git repozitorija odabrali GitHub. Kako bi naša web aplikacija znala gdje je web poslužitelj na jednom mjestu u kodu potrebno je izmijeniti link na gore navedeni URL za Spring Boot. Web aplikacija može se pronaći na `https://dent-all-frontend.vercel.app`.

6. Zaključak i budući rad

Zadatak naše grupe bio je izraditi aplikaciju koja će dentalnim klinikama olakšati dovođenje stranaca kao klijenata tako što će im na jednom mjestu prikazivati sve smještaje u najmu klinike, prijevoznike s kojima klinika surađuje i sve klijente aplikacije. Također aplikacija automatski pri dodavanju klijenta tom klijentu dodjeljuje smještaj i prijevoznika. Taj zadatak odradili smo u dvije faze.

Prva faza bila je fokusirana na izradu projektne dokumentacije, osnovnu funkcionalnost aplikacije i podjelu zadataka pojedinim članovima tima. Pošto nitko u timu osim voditelja se prije nije puno susreo s korištenim tehnologija (React i Spring Boot) ova faza također je iskorištena za upoznavanje s tim tehnologijama. Ovu fazu smo uspješno izvršili.

Druga faza bila je fokusirana na izradu same funkcionalnosti aplikacije. Ovdje je došla do izražaja ne upoznatost članova tima s odabranim tehnologijama što je dovelo do toga da se funkcionalnost implementira sporo i često na ne optimalan način. Više iskustva s tehnologija i projektima općenito bi definitivno pomoglo u ovom slučaju. Također došlo je do izražaja da na projektu radi puno ljudi što je dovelo do raznih stilova u kodu i prema kraju projekta kod je postao prilično teži za navigaciju i snalaženje. Rješenje za ovo je ubuduće uvesti definirane standarde pisanje koda odmah na početku projekta i koristiti automatizirane alate za provođenje tih standardi.

Komunikacija u timu je bila dobra, ali ubuduće bismo odabrali platformu koja je bolje prilagođena timskoj komunikacija od WhatsApp-a jer bi to vjerojatno dovelo do toga da smo svi informiraniji i lakše bi izvršavali zadatke.

Svi smo puno naučili iz sudjelovanja na ovakvom projektu, a najviše kako koristiti odabrane tehnologije za razvoj jednog kompletnog sustava i kako surađivati kao tim tj. kako jedni drugima olakšati posao, a ne smetati.

U aplikaciji nije potpuno implementiran algoritam za automatsku dodjelu prijevoznika klijentu. Trenutni algoritam dodaje samo jedan termin i ne podržava više termina. Također algoritam, ako nije suprotno navedeno ručno, dodjeljuje termin klijentu u 8 ujutro što bi vjerojatno trebalo automatizirani. Funkcionalnost slanja e-pošte klijentima je implementirana, ali platforma Render i bilo koja druga

besplatna platforma koju smo našli onemogućuju slanje e-pošte tako da funkcionalnost radi i testirana je, ali trenutna platforma nam ju ne dopušta.

Popis literature

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

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
   - Datum: 16.10.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Štrulić-Tupek, Stela Dermit, Danko Delimar
   - Teme sastanka:
     - upoznavanje članova tima
     - upoznavanje s temom
     - okvirno definiranje baze podataka
2. sastanak
   - Datum: 23.10.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Štrulić-Tupek, Stela Dermit, Danko Delimar
   - Teme sastanka:
     - završena baza podataka
     - podjela uloga na poslužiteljsku i klijentsku stranu
3. sastanak
   - Datum: 30.10.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Šturlić-Tupek, Stela Dermit, Danko Delimar
   - Teme sastanka:
     - podjela zadataka unutar klijentske i poslužiteljske strane
     - početak izrade generičke funkcionalnosti klijentske i poslužiteljske strane
4. sastanak
   - Datum: 6.11.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Šturlić-Tupek, Stela Dermit, Danko Delimar

- Teme sastanka:
  - završena klijentska strana
  - završena autentifikacija
  - završeno modeliranje baze podataka unutar spring boota

5. sastanak
   - Datum: 18.12.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Šturlić-Tupek, Stela Dermit, Danko Delimar
   - Teme sastanka:
     - Razgovor o tome kako implementirati dijelove klijentske i poslužiteljske strane
6. sastanak
   - Datum: 8.1.2023
   - Prisustvovali: Vedran Lončar, Jan Kozina, Filip Stilinović, Lorena Jakić, Viktorija Šturlić-Tupek, Stela Dermit, Danko Delimar
   - Teme sastanka:
     - Početak intenzivnije implementacije funkcionalnosti i razgovor o tome tko će što

Tablica aktivnosti

	Danko Delimar	Stela Dermit	Lorena Jakić	Jan Kozina	Vedran Lončar	Filip Stilinović	Viktorija Šturlić-Tupek
Upravljanje projektom	60	4	2				
Opis projektnog zadatka	2	1					
Funkcionalni zahtjevi	3						
Opis pojedinih obrazaca	4						
Dijagram obrazaca	1						
Sekvencijski dijagrami	1						
Opis ostalih zahtjeva	1						
Arhitektura i dizajn sustava	2						
Baza podataka			16				
Dijagram razreda	1						
Dijagram stanja			2				
Dijagram aktivnosti			2				
Dijagram komponenti			2				
Korištene tehnologije i alati			1	4	8	8	
Ispitivanje programskog rješenja	4						
Dijagram razmještaja	1						
Upute za puštanje u pogon	2						
Dnevnik sastajanja	1						

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Danko Delimar	Stela Dermit	Lorena Jakić	Jan Kozina	Vedran Lončar	Filip Stilinović	Viktorija Šturlić-Tupek
Zaključak i budući rad	1						
Popis literature	1						
Dodatne stavke kako ste podijelili izradu aplikacije							
npr. izrada početne stranice					8	8	
izrada baze podataka			16				
spajanje s bazom podataka		8					
back end		12		2			

Dijagrami pregleda promjena

Slika 6.1: Dijagram pregled promjena