

# Programsko inženjerstvo

Ak. god. 2023./2024.

## Ozdravi - olakšava život kad imate bolesnu djecu

Dokumentacija, Rev. 2

Grupa: *Tarantule*

Voditelj: *Tara Baće*

Datum predaje: *19. 1. 2024.*

Nastavnik: *Ivana Lulić*

# Sadržaj

<b>1</b>	<b>Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2</b>	<b>Opis projektnog zadatka</b>	<b>5</b>
<b>3</b>	<b>Specifikacija programske potpore</b>	<b>9</b>
3.1	Funkcionalni zahtjevi . . . . .	9
3.1.1	Obrasci uporabe . . . . .	12
3.1.2	Sekvencijski dijagrami . . . . .	33
3.2	Ostali zahtjevi . . . . .	40
<b>4</b>	<b>Arhitektura i dizajn sustava</b>	<b>41</b>
4.1	Baza podataka . . . . .	43
4.1.1	Opis tablica . . . . .	43
4.1.2	Dijagram baze podataka . . . . .	46
4.2	Dijagram razreda . . . . .	47
4.3	Dijagram stanja . . . . .	53
4.4	Dijagram aktivnosti . . . . .	54
4.5	Dijagram komponenti . . . . .	56
<b>5</b>	<b>Implementacija i korisničko sučelje</b>	<b>57</b>
5.1	Korištene tehnologije i alati . . . . .	57
5.2	Ispitivanje programskog rješenja . . . . .	59
5.2.1	Ispitivanje komponenti . . . . .	59
5.2.2	Ispitivanje sustava . . . . .	62
5.3	Dijagram razmještaja . . . . .	68
5.4	Upute za puštanje u pogon . . . . .	69
<b>6</b>	<b>Zaključak i budući rad</b>	<b>76</b>
	<b>Popis literature</b>	<b>77</b>
	<b>Indeks slika i dijagrama</b>	<b>79</b>

**Dodatak: Prikaz aktivnosti grupe**

**80**

# 1. Dnevnik promjena dokumentacije

## *Kontinuirano osvježavanje*

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak, dodani funkcionalni zahtjevi te neki obrasci uporabe.	Benedicte Gabelica	25.10.2023.
0.2	Završeni prvotno zamišljeni oblikovni obrasci.	Benedicte Gabelica	28.10.2023.
0.3	Dodan opis aplikacije	Benedicte Gabelica	4.11.2023.
0.4	Dodani sekvencijski dijagrami	Tara Baće	5.11.2023.
0.5	Dodan <i>Use Case</i> dijagram i jedan sekvencijski dijagram, funkcionalni i nefunkcionalni zahtjevi i dodatak A	Josip Pavlič	03.11.2023.
0.6.1	Dodan opis baze podataka i opis entiteta	Alan Jerbić	6.11.2023.
0.7	Popravljeni oblikovni obrasci	Benedicte Gabelica	10.11.2023.
0.8	Popravljen sekvencijski dijagram 3.6	Josip Pavlič	13.11.2023.
0.9	Uneseni ostali zahtjevi	Tara Baće	14.11.2023.
0.10	Dodan opis arhitekture	Benedicte Gabelica	16.11.2023.
0.11	Prepravljen opis baze podataka i dodan dijagram	Alan Jerbić	16.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.12	Napravljen dijagram klasa	Eugen Bošnjak	16.11.2023.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Tara Baće	16.11.2023.
1.1	Dodani dijagram stanja i dijagram aktivnosti	Benedicte Gabelica	31.12.2023.
1.2	Dodan opis korištenih alata i tehnologija	Benedicte Gabelica	8.1.2024.
1.3	Dodani dijagram komponenti te dijagram razmještaja	Benedicte Gabelica	9.1.2023.
1.4	Dodan zaključak, dodani/prepravljeni opisi u dijagramu razreda	Benedicte Gabelica	17.1.2024.
1.5	Dodan opis ispitivanja sustava	Tara Baće, Vilim Si-vec	19.1.2024.
1.6	Dodan dio dijagrama razreda	Benedicte Gabelica	19.1.2024.
1.7	Dovršen dijagram razreda	Alan Jerbić	19.1.2024.
1.8	Dodan opis ispitivanja komponenti	Eugen Bošnjak	19.1.2024.
1.9	Dodane upute za deploy	Tara Baće, Eugen Bošnjak	19.1.2024.
2.0	Konačni tekst predloška dokumentacije	Tara Baće	19.1.2024.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za oblikovanje web aplikacije *Ozdravi* koja će korisnicima omogućiti lakšu komunikaciju s liječnicima te brži način dolaska do potvrde o bolovanju zbog bolesti djeteta. Glavna ideja je omogućiti interakciju između više instanci u zdravstvu.

Ova aplikacija namijenjena je prvenstveno roditeljima koji imaju djecu podložnu čestim zdravstvenim problemima, ali mogu ju koristiti svi roditelji.

Prilikom otvaranja glavne stranice, neregistrirani korisnik može vidjeti opis aplikacije te specifičnosti koje nudi. Omogućeno mu je prijavljivanje u sustav s postojećim računom (prijava s OIB-om i lozinkom) ili kreiranje novog računa. Za kreiranje novog računa su mu također potrebni samo OIB i lozinka koju želi postaviti za svoj račun. Registrirati se mogu samo roditelji, dok su računi liječnika i pedijatra već postojeći u sustavu te se oni samo prijavljuju.

Vrste korisnika web aplikacije su:

- Roditelj
- Liječnik obiteljske medicine
- Pedijatar
- Administrator

*Roditelj* prijavom u sustav može odabrati želi li ući u svoj profil ili u profil djeteta. Pri ulasku u profil roditelj ima opciju pregleda pristiglih poruka ili opciju pregleda osobnih podataka profila. Važnu mogućnost koju ima roditelj je dodavanje maila svog poslodavca u sustav (preko svoga profila) te dodavanje maila škole ili vrtića za pojedino dijete (preko profila djeteta).

Najvažnija funkcionalnost koju pruža aplikacija i po čemu se ona razlikuje od postojećih je mogućnost automatiziranog odobravanja bolovanja. Roditelj će na svom profilu primiti obavijest ako mu je odobreno bolovanje (zbog svoje bolesti ili bolesti djeteta) te informaciju je li o tome obaviješten i poslodavac (preko maila koji je roditelj upisao na profilu). Na profilu djeteta će roditelj primiti obavijest ako

je pedijatar potvrdio bolest djeteta te informaciju o tome je li ispričnica poslana vrtiću ili školi.

Osim te funkcionalnosti, roditelj može za sebe ili za dijete učitati nalaz u sustav, pregledati nalaz iz laboratorija kojeg je poslao liječnik ili pedijatar te pregledati pregled na kojem je naručen on ili njegovo dijete (na odgovarajućem profilu). Ovdje je vidljiva i još jedna mogućnost koju nudi aplikacija - roditelju će se prikazati karta gdje će biti označen sve zdravstvene institucije u kojima je moguće obaviti vrstu pregleda za koju je roditelj ili njegovo dijete naručeno.

*Pedijatar* pri prijavi u sustav ima popis djece prijavljene kod njega te ima opciju prijavljivanja novog djeteta kod sebe. Pri odabiru određenog djeteta mu se otvori prozor gdje može vidjeti povijest poruka s roditeljem djeteta. Pedijatar ima niz mogućnosti kao što su slanje nalaza iz laboratorija ili naručivanje djeteta na specijalistički pregled.

Glavna mogućnost je unos podataka o obavljenom pregledu tj. dijagnozi djeteta pri čemu pedijatar može odabrati i opciju da se vrtiću/školi šalje ispričnica te zatražiti bolovanje roditelja. Preporuka za bolovanje roditelja će se proslijediti liječniku. Time se roditelj oslobodađa od potrebe za odlaskom i do pedijatra i do liječnika da bi mu se potencijalno odobrilo bolovanje - to se sve automatizira u sustavu.

*Liječnik obiteljske medicine* pri prijavi u sustav ima pred sobom popis prijavljenih roditelja kod njega te ima opciju prijavljivanja novog roditelja. Pri odabiru određenog pacijenta otvara mu se prozor gdje može vidjeti poruke koje je s njim izmijenio. Liječnik ima niz mogućnosti kao i pedijatar: slanje nalaza iz laboratorija roditelja te naručivanje roditelja na specijalistički pregled.

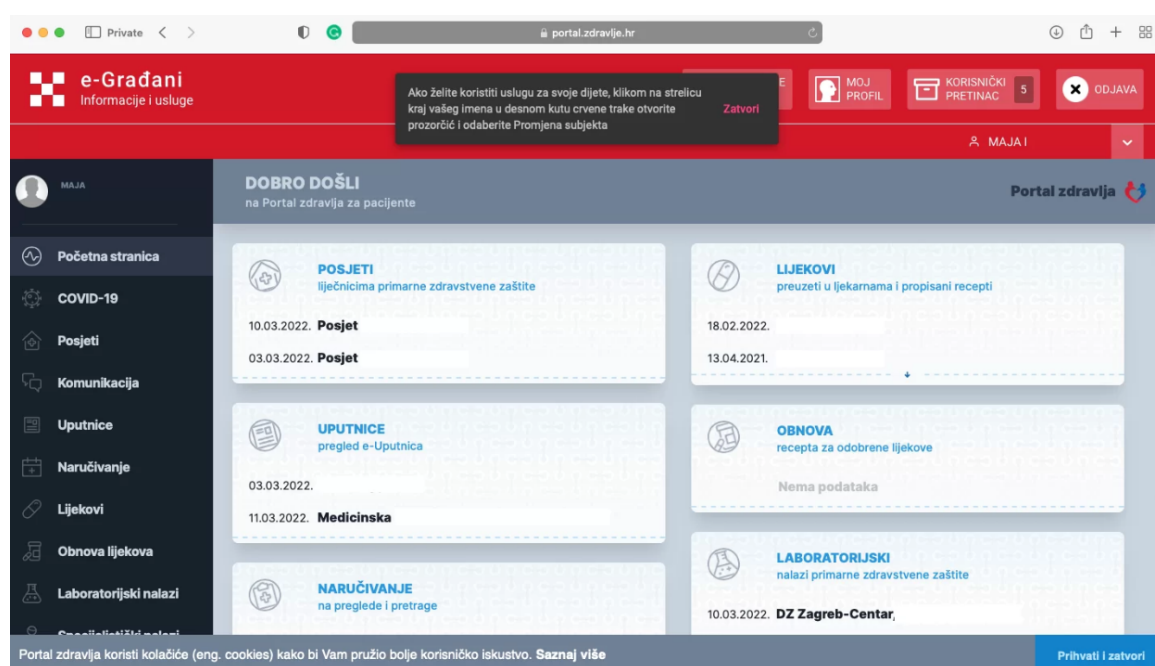
U okviru ove aplikacije najvažnija je činjenica da liječnik roditelju preko aplikacije može odobriti bolovanje. Prvi način kojim se to postiže je standardan: liječnik može upisati podatke o obavljenom pregledu i dijagnozi te odabrati opciju "Propisati bolovanje" ako je to nužno. Drugi način je da liječnik odobri preporuku za bolovanje koju je ponudio pedijatar na temelju bolesti djeteta. Potvrđivanjem te preporuke roditelju dolazi poruka o odobrenom bolovanju.

*Administrator* ima mogućnost upravljanja korisničkim računima roditelja. Administrator je taj koji u sustav upisuje postojeće osobe (njihovo ime, prezime i

OIB), a roditelj se može registrirati samo ako u aplikaciji postoji zapis o njegovom postojanju. Administrator u sustav upisuje i djecu već upisanih roditelja koju onda može povezati s njihovim roditeljem unutar sustava - time roditelj dobiva mogućnost pregleda profila tog djeteta prilikom prijave u sustav. Administrator ima mogućnost i mijenjanja osobnih podataka i roditelja i djece.

Ova web aplikacija može se usporediti s uslugom Portal zdravlja koja je građanima dostupna preko sustava E-Građani. Portal zdravlja nudi neke slične mogućnosti kao i aplikacija *Ozdravi*. Roditelj može pregledavati svoj profil ili profil djeteta. Na profilu može vidjeti nadležnog liječnika, popis prošlih dijagnoza, nalaze iz laboratorija te se može i naručiti ili otkazati pregled.

Ipak, Portal zdravlja ne nudi važnu opciju koju nudi i ova aplikacija - mogućnost izdavanja bolovanja roditelju na temelju bolesti djeteta bez da pritom roditelj mora ići i do pedijatra, i do liječnika, i do poslodavca. Sustav web aplikacije automatski zahtjev za bolovanjem šalje od pedijatra, do liječnika te naposljetku i do poslodavca, a roditelj je o tome obaviješten preko poruke.



Slika 2.1: Snimka zaslona koja prikazuje usluge Portala zdravlja

Sustav bi se u budućnosti mogao nadograditi tako da roditelju dozvoljava mogućnost naručivanja pregleda i biranja termina za sebe i svoju djecu direktno iz aplikacije.



Također, sučelje bi se moglo optimizirati time da se poruke mogu filtrirati po vrsti poruke (naručen pregled, nalaz iz laboratorija itd.). Obavljeni pregledi i dijagnoze mogli bi se smjestiti u zasebnom prozoru radi preglednosti.

## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

#### Dionici:

1. Neregistrirani korisnik
2. Roditelji
3. Zaposlenici u zdravstvenim ustanovama
  - (a) Liječnici obiteljske medicine
  - (b) Pedijatri
4. Administrator
5. Razvojni tim

#### Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
  - (a) pročitati opis stranice
  - (b) se registrirati u sustav, za što mu je potreban OIB te lozinka
  - (c) se prijaviti u sustav, za što mu je potreban OIB te lozinka
2. Roditelj (inicijator) može:
  - (a) pregledavati i mijenjati svoje osobne podatke na svom profilu (adresu, mail poslodavca)
  - (b) pregledavati i mijenjati osobne podatke svoje djece na njihovim profilima (adresu, mail škole/vrtića)
  - (c) pregledavati obavijesti o odobrenom bolovanju od strane liječnika ili poslanoj ispričnici u školu djeteta
  - (d) učitati nalaz dobiven temeljem usluge u privatnoj ustanovi te za njega zatražiti povratnu informaciju od liječnika ili pedijatra
  - (e) pregledavati obavijesti o pristiglim nalazima iz laboratorija
  - (f) pregledavati potvrde o naručivanju na određeni pregled za sebe ili svoju djecu s prikazanom lokacijom pregleda

- (g) pregledavati povijest posjeta liječniku i dijagnoze za sebe i svoju djecu
- (h) tražiti dodatna pojašnjenja od liječnika ili pedijatra u vezi bilo koje od gore navedenih stavki

3. Liječnik obiteljske medicine (inicijator) može:

- (a) pregledavati popis svih neprijavljenih roditelja i prijaviti ih kod sebe
- (b) pregledavati popis pacijenata prijavljenih kod njega (roditelja)
- (c) evidentirati pregled pacijenta te događaje na njemu te utvrditi bolest čime se šalje mail poslodavcu
- (d) odobriti preporuku za bolovanje za roditelja koju je izdao pedijatar
- (e) poslati obavijest roditelju o pristiglom nalazu iz laboratorija
- (f) pregledavati nalaze koji su u sustav učitani od strane roditelja te odgovoriti na njih
- (g) naručiti pacijenta na specijalistički pregled
- (h) pregledati pitanja roditelja i odgovoriti na ista

4. Pedijatar (inicijator) može:

- (a) pregledavati popis sve neprijavljene djece i prijaviti ih kod sebe
- (b) pregledavati popis sve djece prijavljene kod njega
- (c) izdati preporuku za bolovanje roditelju čije je dijete bolesno
- (d) evidentirati pregled djeteta te događaje na njemu te utvrditi bolest čime se šalje ispričnica u školu
- (e) poslati obavijest roditelju o pristiglom nalazu iz laboratorija
- (f) pregledavati nalaze koji su u sustav učitani od strane roditelja te odgovoriti na njih
- (g) naručiti dijete na specijalistički pregled
- (h) pregledati pitanja roditelja i odgovoriti na ista

5. Administrator (inicijator) može:

- (a) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
- (b) brisati korisnike
- (c) mijenjati osobne podatke pojedinog korisnika
- (d) unijeti registre djece i roditelja (za svaku osobu ime, prezime, OIB i adresu)
- (e) registriranom roditelju pridijeliti liječnika obiteljske medicine
- (f) djetetu pridijeliti pedijatra

(g) registriranog roditelja povezati s djetetom čiji podaci postoje u sustavu

6. Baza podataka (sudionik):

- (a) pohranjuje podatke o svim registriranim korisnicima i njihovim ulogama
- (b) pohranjuje podatke o svim postojećim porukama
- (c) pohranjuje podatke o postojećim ustanovama i pregledima koje je moguće obaviti u svakoj

### 3.1.1 Obrasci uporabe

#### UC1 - Registriraj se

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je u bazi podataka unio osnovne informacije o korisniku (ime, prezime, OIB), korisnik je otvorio početnu stranicu aplikacije
- **Opis osnovnog tijeka:**
  1. Korisnik na početnoj stranici odabire opciju "Registracija".
  2. Sustav otvara ekran registracije.
  3. Korisnik unosi svoj OIB te željenu lozinku
  4. Korisnik potvrđuje unos podataka odabirom akcije "registracija".
  5. Sustav provjerava i utvrđuje da je unos uspješan te obavještava korisnika o uspješnoj registraciji.
- **Opis mogućih odstupanja:**
  - 4.a Korisnik odabere opciju "Odustani".
    1. Sustav korisnika vraća na početnu stranicu (korak 1)
  - 5.a Korisnik je unio krivi format traženih podataka (kriva duljina ili format OIB-a ili lozinke).
    1. Sustav obavještava korisnika da je došlo do greške te ostaje na stranici registracije (ne briše unesene podatke).
  - 5.b Odabir OIB-a za koji je zabilježeno da se osoba već registrirala u sustav
    1. Sustav obavještava korisnika o neuspjeloj registraciji i prikaže mu poruku greške te ostaje na stranici registracije (ne briše unesene podatke).

#### UC2 - Prijavi se

- **Glavni sudionik:** Korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran, korisnik je otvorio početnu stranicu aplikacije
- **Opis osnovnog tijeka:**
  1. Korisnik na početnoj stranici odabire opciju "Prijava".
  2. Sustav otvara ekran prijave.

3. Korisnik unosi svoj OIB te odgovarajuću lozinku.
  4. Korisnik odabere opciju "prijava".
  5. Sustav provjerava ispravnost unesenih podataka i utvrđuje da je unos uspješan te prikazuje korisniku ekran njegovih profila.
- **Opis mogućih odstupanja:**
    - 4.a Korisnik odabere opciju "Odustani".
      1. Sustav korisnika vraća na početnu stranicu (korak 1).
    - 5.a Korisnik je unio krivi format traženih podataka (kriva duljina ili format OIB-a ili lozinke).
      1. Sustav obavještava korisnika da je došlo do greške te ostaje na stranici prijave (ne briše unesene podatke).
    - 5.b Sustav provjerava podatke i utvrđuje da je uneseni par OIB - lozinka netočan.
      1. Sustav obavještava korisnika o neuspjeloj prijavi i prikaže mu poruku greške te ostaje na stranici prijave (ne briše unesene podatke).

### UC3 - Pregledaj opis aplikacije

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati osnovne informacije o aplikaciji.
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik otvori početnu stranicu aplikacije.
  2. Na početnoj stranici se prikazuju opis i svrha aplikacije te se nude opcije registracije i prijave u sustav (ako korisnik nije prijavljen).

### UC4.1 - Pregledaj profil

- **Glavni sudionik:** Roditelj
- **Cilj:** Pregledati svoj profil i osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici odabere svoj profil označen natpisom "Moj profil".
  2. Sustav roditelju prikazuje profil s pristiglim porukama.
  3. Roditelj stisne na ikonu profila.

4. Aplikacija prikazuje osobne podatke roditelja (ime, prezime, OIB, adresa, mail poslodavca, liječnik obiteljske medicine).

#### UC4.2 - Pregledaj profil djeteta

- **Glavni sudionik:** Roditelj
- **Cilj:** Pregledati profil i osobne podatke djeteta
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici odabere profil djeteta čiji profil želi pregledati.
  2. Sustav roditelju prikazuje profil s pristiglim porukama.
  3. Roditelj stisne na ikonu profila.
  4. Aplikacija prikazuje osobne podatke djeteta (ime, prezime, OIB, adresa, mail škole/vrtića, pedijatar)
- **Opis mogućih odstupanja:**
  - 1.a Dijete još nije povezano s roditeljem u sustavu.
    1. Roditelj neće imati opciju pregleda profila tog djeteta.

#### UC5 - Promjeni osobne podatke

- **Glavni sudionik:** Roditelj
- **Cilj:** Ažurirati osobne podatke roditelja ili djeteta
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici odabere profil čije osobne podatke želi mijenjati (svoj ili djetetov).
  2. Sustav roditelju vraća pregled pristiglih poruka na odabranom profilu.
  3. Roditelj stisne na ikonu profila.
  4. Sustav roditelju vraća pregled osobnih podataka za taj profil.
  5. Na stranici s podacima roditelj bira opciju za promjenu podataka.
  6. Sustav vraća prozor u kojem roditelj može izmijeniti podatke.
  7. Roditelj izmijeni podatke (može izmijeniti samo adresu i mail poslodavca, tj. mail škole/vrtića) i odabere opciju Spremi.

8. Baza podataka se ažurira.

- **Opis mogućih odstupanja:**

7.a Roditelj promijeni osobne podatke ali odabere opciju "Odustani".

1. Sustav neće pohraniti obavljene promjene i vratit će roditelja na pregled profila (korak 4).

### **UC6 - Učitaj nalaz u sustav**

- **Glavni sudionik:** Roditelj
- **Cilj:** Učitati postojeći nalaz u sustav i poslati ga liječniku ili pedijatru na pregled
- **Sudionici:** Baza podataka, liječnik obiteljske medicine/pedijatar
- **Preduvjet:** Roditelj je prijavljen, administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici bira profil osobe čiji nalaz želi učitati.
  2. Sustav roditelju vraća stranicu koja sadrži pregled poruka na tom profilu.
  3. Roditelj na stranici bira opciju "Naruči".
  4. Sustav roditelju prikazuje prozor u koji roditelj može učitati privitak ili pisati.
  5. Roditelj učitava nalaz u sustav i opiše ga ili postavlja pitanje ako to želi.
  6. Roditelj odabere opciju "Pošalji" čime se nalaz šalje liječniku obiteljske medicine ili pedijatru (ovisno o tome s čijeg se profila šalje).
  7. Baza podataka se ažurira.
  8. Sustav vraća roditelja na prikaz poruka profila.
- **Opis mogućih odstupanja:**
  - 6.a Roditelj učitava nalaz ali odabere opciju "Zatvori".
    1. Nalaz se neće poslati i sustav će vratiti roditelja na pregled poruka na profilu.
  - 6.b Roditelj nije prijavljen ni kod jednog liječnika ili djeteta nije prijavljeno ni kod jednog pedijatra.
    1. Roditelju neće biti omogućena opcija "Pošalji".

### **UC7 - Pregledaj podatke o naručenom pregledu**

- **Glavni sudionik:** Roditelj



- **Cilj:** Pregledati podatke o naručenom pregledu za sebe ili dijete (vrsta pregleda i lokacije na kojima se može obaviti)
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, liječnik je naručio roditelja na specijalistički pregled/pedijatar je naručio dijete na specijalistički pregled (ovisno o otvorenom profilu), administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici bira profil osobe čije naručene preglede želi vidjeti.
  2. Sustav vraća pregled poruka na tom profilu.
  3. Roditelj odabere jednu od poruka s naslovom "[NARUČEN PREGLED]".
  4. Sustav roditelju vraća pregled poruke - roditelj može vidjeti koja je vrsta pregleda te može na prikazanom OpenStreetMap pregledu vidjeti u kojim najbližim zdravstvenim ustanovama (s obzirom na adresu roditelja) se pregled može obaviti.
- **Opis mogućih odstupanja:**
  - 4.a Roditelj nije unio svoju adresu u sustav.
    1. Sustav umjesto karte prikazuje poruku pogreške.

#### UC8 - Pregledaj podatke o odobrenom bolovanju

- **Glavni sudionik:** Roditelj
- **Cilj:** Pregledati podatke o odobrenom bolovanju: razlog bolovanja i trajanje bolovanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, liječnik je odobrio/preporučio bolovanje roditelju
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici bira opciju "Moj profil".
  2. Sustav roditelju vraća pregled poruka na njegovom profilu.
  3. Roditelj odabere jednu od poruka s naslovom "[BOLOVANJE]".
  4. Sustav roditelju vraća pregled poruke - roditelj može vidjeti poruku liječnika te informaciju da je odgovarajući mail poslan poslodavcu.
- **Opis mogućih odstupanja:**
  - 4.a Roditelj nije unio mail adresu svog poslodavca.
    1. Unutar obavijesti neće biti naznačeno da je mail poslan poslodavcu.

#### UC9 - Pregledaj obavijest o poslanom mailu vrtiću/školi

- **Glavni sudionik:** Roditelj
- **Cilj:** Pregledati obavijesti o poslanoj ispričnici vrtiću ili školi
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, roditelj je na profilu djeteta unio podatak o mail adresi vrtića ili škole, pedijatar je utvrdio bolest djeteta, administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na stranici bira profil djeteta za kojeg se žele pregledati poslane ispričnice.
  2. Sustav roditelju vraća pregled poruka s tog profila.
  3. Roditelj odabere jednu od poruka s naslovom "[POSLANA ISPRIČNICA]".
  4. Sustav roditelju vraća pregled poruke - roditelj može vidjeti na koji mail je poslana ispričnica.
- **Opis mogućih odstupanja:**
  - 3.a Roditelj nije unio mail adresu vrtića/škole na profilu djeteta.
    1. Ispričnica neće biti poslana (a time roditelj neće nikada ni dobiti obavijest o poslanoj ispričnici).

#### UC10 - Pregledaj obavljene preglede i dijagnoze

- **Glavni sudionik:** Roditelj
- **Cilj:** Pregledati podatke o prošlim pregledima i dijagnozama
- **Sudionici:** Baza podataka
- **Preduvjet:** Roditelj je prijavljen, administrator je povezao roditelja s djetetom
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici bira profil čije obavljene preglede želi vidjeti.
  2. Sustav roditelju vraća pregled poruka na profilu.
  3. Roditelj odabere jednu od poruka s naslovom "[OBAVLJENI PREGLED]".
  4. Sustav vraća pregled poruke roditelju - roditelj može vidjeti informacije o obavljenom pregledu te dijagnozi.
- **Opis mogućih odstupanja:**
  - 3.a Roditelj ili dijete nisu obavili nijedan pregled ili pregled čije informacije želimo vidjeti još nije zapisan u sustav od strane liječnika ili pedijatra.
    1. Roditelju se neće prikazati nijedna poruka s naslovom [OBAVLJENI PREGLED], tj. neće se prikazati poruka koja odgovara pregledu čine

informacije želi vidjeti.

### UC11 - Traži dodatna pojašnjenja liječnika ili pedijatra

- **Glavni sudionik:** Roditelj
- **Cilj:** Odgovoriti na poruku koju je primio od liječnika ili pedijatra
- **Sudionici:** Baza podataka, liječnik obiteljske medicine/pedijatar
- **Preduvjet:** Roditelj je prijavljen u sustav, roditelj je prijavljen kod liječnika/dijete je prijavljeno kod pedijatra, liječnik je poslao poruku/pedijatar je poslao poruku, administrator je povezao dijete s roditeljem
- **Opis osnovnog tijeka:**
  1. Roditelj na početnoj stranici odabere profil osobe za koju želi tražiti dodatna objašnjenja.
  2. Sustav roditelju vraća pregled poruka profila.
  3. Roditelj bira jednu od poruka koju je primio od liječnika ili pedijatra (ovisno o profilu).
  4. Sustav roditelju vraća pregled poruke.
  5. Roditelj odabere opciju "Odgovori".
  6. Sustav roditelju vraća novi prozor u kojem roditelj može utipkati svoje pitanje.
  7. Roditelj utipika svoju poruku i odabere opciju "Pošalji".
  8. Poruka se pošalje liječniku ili pedijatru i sustav roditelja vrati na pregled poruka profila.
- **Opis mogućih odstupanja:**
  - 7.a Roditelj odabere opciju "Zatvori".
    1. Odgovor neće biti poslan i sustav vraća roditelja na pregled poruka profila.

### UC12 - Pregledaj popis djece prijavljene kod pedijatra

- **Glavni sudionik:** Pedijatar
- **Cilj:** Pregledati popis djece prijavljene kod njega
- **Sudionici:** Baza podataka
- **Preduvjet:** Pedijatar je prijavio djecu kod sebe ili je administrator povezao dijete s pedijatrom, pedijatar je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Pedijatar nakon prijave na početnoj stranici može vidjeti popis djece prijavljene kod njega.

**UC13 - Prijavi novo dijete kod pedijatra**

- **Glavni sudionik:** Pedijatar
- **Cilj:** Pregledati popis neprijavljene djece i prijaviti ih kod sebe
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je unio podatke o djeci, pedijatar je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici bira opciju "Dodaj pacijenta".
  2. Sustav pedijatru vraća popis djece koja još nisu prijavljena kod nekog pedijatra.
  3. Pedijatar u popisu neprijavljene djece pronalazi dijete koje želi prijaviti kod sebe, stisne na njega te odabere opciju "Upiši".
  4. Sustav bilježi da je to dijete sada prijavljeno kod tog pedijatra i vraća pedijatra na popis djece prijavljene kod njega.

**UC14 - Upiši podatke o pregledu djeteta obavljenom kod pedijatra i dijagnozu**

- **Glavni sudionik:** Pedijatar
- **Cilj:** Upisati podatak o obavljenom pregledu djeteta
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete čiji pregled želi unijeti.
  2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar bira opciju "Nova poruka".
  4. Sustav pedijatru vraća prozor u kojem pedijatar može unijeti podatke o pregledu.
  5. Pedijatar bira kao naslov poruke "Dijagnoza", opiše pregled i dijagnozu te opcionalno može odabrati šalje li se ispričnica i preporuka za bolovanje roditelja.
  6. Pedijatar bira opciju "Pošalji".
  7. Pregled je zabilježen i sustav vraća pedijatra na pregled poruka s profilom djeteta.
- **Opis mogućih odstupanja:**
  - 6.a Pedijatar bira opciju "Zatvori".

1. Podaci neće biti upisani i sustav vraća pedijatra na pregled poruka s profilom djeteta.

#### **UC15 - Izdaj preporuku za bolovanje za roditelja bolesnog djeteta**

- **Glavni sudionik:** Pedijatar
- **Cilj:** Izdati preporuku za bolovanje roditelju bolesnog djeteta
- **Sudionici:** Baza podataka, roditelj, liječnik obiteljske medicine
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete čijem roditelju želi izdati preporuku za bolovanje.
  2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar bira opciju "Nova poruka".
  4. Sustav pedijatru vraća prozor u kojem može opisati dijagnozu djeteta.
  5. Pedijatar bira naslov "Dijagnoza", opiše razlog izdavanja preporuke te bira opciju "Bolovanje roditelja" i opciju "Ispričnica".
  6. Pedijatar bira opciju "Pošalji".
  7. Sustav pošalje poruku liječniku, a pedijatra vraća na pregled poruka s profilom djeteta.
- **Opis mogućih odstupanja:**
  - 6.a Pedijatar bira opciju "Zatvori".
    1. Preporuka neće biti upisana i sustav vraća pedijatra na pregled poruka s profilom djeteta.

#### **UC16 - Pošalji nalaz iz laboratorija djeteta**

- **Glavni sudionik:** Pedijatar
- **Cilj:** Roditelju djeteta poslati laboratorijski nalaz djeteta
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete čiji pregled želi unijeti.
  2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar bira opciju "Nova poruka".
  4. Sustav pedijatru vraća prozor u kojem pedijatar može opisati te priložiti nalaz.

5. Pedijatar bira kao naslov "Nalaz iz laboratorija", opiše nalaz, može priložiti dokument biranjem opcije "Prilog" te opcionalno može odabrati šalje li se ispričnica i preporuka za bolovanje roditelja.
  6. Pedijatar bira opciju "Pošalji".
  7. Sustav šalje poruku roditelju i pedijatra vraća na pregled poruka s profilom djeteta.
- **Opis mogućih odstupanja:**
    - 6.a Pedijatar bira opciju "Zatvori".
      1. Nalaz neće biti poslan i sustav vraća pedijatra na pregled poruka s profilom djeteta.

#### UC17 - Pregledaj učitane nalaza djeteta od strane roditelja

- **Glavni sudionik:** Pedijatar
- **Cilj:** Pregledati nalaze djeteta koje su roditelji učitali u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra, pedijatar je primio obavijest o učitanoj nalazu
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete čije nalaze želi vidjeti.
  2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar bira jednu od poruka koje je primio od roditelja.
  4. Sustav pedijatru vraća pregled odabrane poruke.
  5. Pedijatar može pregledati nalaz te dodatna pitanja ili informacije koje je roditelj priložio.

#### UC18 - Odgovori roditelju na upit o bolesti djeteta

- **Glavni sudionik:** Pedijatar
- **Cilj:** Odgovoriti na poruku roditelja u vezi djeteta
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra, pedijatar je primio obavijest o učitanoj nalazu
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete za koje postoji poruka poslana od strane roditelja koja očekuje odgovor.

2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar odabere jednu od poruka koju je poslao roditelj.
  4. Sustav pedijatru vraća pregled poruke.
  5. Pedijatar odabere opciju "Odgovori".
  6. Sustav pedijatru vraća prozor u kojem pedijatar može utipkati svoj odgovor.
  7. Pedijatar odabere opciju "Pošalji".
  8. Sustav pošalje poruku roditelju i pedijatra vraća na pregled poruka s profilom djeteta.
- **Opis mogućih odstupanja:**
    - 7.a Pedijatar odabere opciju "Zatvori".
      1. Odgovor neće biti poslan i sustav vraća pedijatra na pregled poruka s profilom djeteta.

#### **UC19 - Naruči dijete na specijalistički pregled**

- **Glavni sudionik:** Pedijatar
- **Cilj:** Naručiti dijete na specijalistički pregled
- **Sudionici:** Baza podataka
- **Preduvjet:** Pedijatar je prijavljen u sustav, dijete je prijavljeno kod pedijatra
- **Opis osnovnog tijeka:**
  1. Pedijatar na početnoj stranici iz popisa djece prijavljene kod njega bira dijete koje želi naručiti na specijalistički pregled.
  2. Sustav pedijatru vraća pregled poruka s profilom djeteta.
  3. Pedijatar odabere opciju "Nova poruka".
  4. Sustav pedijatru vraća prozor u kojem pedijatar bira vrstu pregleda te može utipkati napomenu.
  5. Pedijatar kao naslov poruke odabere "Specijalistički pregled", odabere vrstu bolesti te dodaje napomenu ako to želi.
  6. Pedijatar odabere opciju "Pošalji".
  7. Sustav šalje poruku roditelju i pedijatra vraća na pregled poruka s profilom djeteta.
- **Opis mogućih odstupanja:**
  - 6.a Pedijatar bira opciju "Zatvori".
    1. Dijete neće biti naručeno i sustav vraća pedijatra na pregled poruka s profilom djeteta.

#### **UC20 - Pregledaj popisa roditelja prijavljenih kod nekog liječnika obiteljske medicine**

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Pregledati popis roditelja prijavljenih kod njega
- **Sudionici:** Baza podataka
- **Preduvjet:** Liječnik je prijavio roditelja kod sebe ili je administrator povezoao roditelja s liječnikom
- **Opis osnovnog tijeka:**
  1. Liječnik nakon prijave na početnoj stranici može vidjeti popis roditelja (pacijenata) prijavljenih kod njega.

#### UC21 - Prijavi novog roditelja kod liječnika

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Pregledati popis neprijavljenih roditelja i prijaviti ih kod sebe
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Administrator je unio podatke o roditeljima, liječnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici bira opciju "Dodaj pacijenta".
  2. Sustav liječniku vraća popis roditelja koji još nisu prijavljeni kod nekog liječnika.
  3. Liječnik u popisu neprijavljenih roditelja pronalazi osobu koje želi prijaviti kod sebe, stisne na nju i odabere opciju "Upiši".
  4. Sustav zabilježi da je roditelj sada prijavljen kod tog liječnika i vraća liječnika na popis roditelja prijavljenih kod njega.

#### UC22 - Upiši podatke o pregledu roditelja obavljenom kod liječnika i dijagnozu

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Upisati podatak o obavljenom pregledu roditelja
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljeno kod liječnika
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja čiji pregled želi unijeti.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik bira opciju "Nova poruka".
  4. Sustav liječniku vraća prozor u kojem liječnik može upisati podatke o pregledu.



5. Liječnik kao naslov bira "Dijagnoza", opiše pregled i dijagnozu te opcionalno može odabrati izdaje li se bolovanje za roditelja.
  6. Liječnik bira opciju "Pošalji".
  7. Sustav šalje poruku roditelju i liječnika vraća na pregled roditelja prijavljenih kod njega.
- **Opis mogućih odstupanja:**
    - 6.a Liječnik bira opciju "Zatvori".
      1. Pregled neće biti upisan i sustav će vratiti liječnika na pregled roditelja upisanih kod njega.

### UC23 - Odobri preporuku za bolovanje za roditelja bolesnog djeteta

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Odobriti preporuku za bolovanje roditelju bolesnog djeteta koju je izdao pedijatar
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, pedijatar je izdao preporuku za bolovanje roditelju zbog bolesti djeteta
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja za kojeg želi odobriti preporuku za bolovanje.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik bira poruku s naslovom "[PREPORUKA BOLOVANJA]".
  4. Sustav liječniku vraća pregled poruke.
  5. Liječnik u novootvorenom prozoru bira opciju "Odobri bolovanje".
  6. Bolovanje se odobri roditelju i sustav vraća liječnika na pregled poruka s profilom roditelja.

### UC24 - Propiši bolovanje za bolesnog roditelja

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Izdati preporuku za bolovanje bolesnom roditelju
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljen kod pedijatra
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja kojem želi propisati bolovanje.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.

3. Liječnik bira opciju "Nova poruka".
  4. Sustav liječniku vraća prozor u kojem liječnik može upisati razloge propisivanja bolovanja.
  5. Liječnik kao naslov bira "Dijagnoza", opiše razlog propisivanja bolovanja te bira opciju "Bolovanje roditelja".
  6. Liječnik bira opciju "Pošalji".
  7. Sustav pošalje poruku roditelju te liječnika vraća na pregled poruka s profilom roditelja.
- **Opis mogućih odstupanja:**
    - 6.a Liječnik odabere opciju "Zatvori".
      1. Preporuka za bolovanje neće biti izdana i sustav vraća liječnika na pregled roditelja upisanih kod njega.

#### UC25 - Pošalji nalaz iz laboratorija roditelja

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Roditelju poslati njegov laboratorijski nalaz
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljen kod liječnika
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja čiji nalaz želi unijeti.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik bira opciju "Nova poruka".
  4. Sustav liječniku vraća prozor u kojem liječnik može opisati i priložiti nalaz.
  5. Liječnik bira kao naslov "Nalaz iz laboratorija", opiše nalaz, može priložiti dokument biranjem opcije "Prilog" te opcionalno može odabrati i opciju za izdavanje bolovanja.
  6. Liječnik bira opciju "Pošalji".
  7. Sustav pošalje poruku roditelji i vraća liječnika na pregled poruka s profilom roditelja.
- **Opis mogućih odstupanja:**
  - 6.a Liječnik odabere opciju "Zatvori".
    1. Nalaz neće biti poslan i sustav vraća liječnika na pregled poruka s profilom roditelja.

#### UC26 - Pregledaj učitane nalaze roditelja

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Pregledati nalaze koje je pojedini pacijent (roditelj) učitao u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljen kod liječnika, liječnik je primio obavijest o učitanoj nalazu
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira onog čije nalaze želi vidjeti.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik bira jednu od poruka koju je poslao roditelj.
  4. Sustav liječniku vraća pregled poruke.
  5. Liječnik može pregledati nalaz te dodatna pitanja ili informacije koje je roditelj priložio.

#### UC27 - Odgovori na upit roditelja

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Odgovoriti na poruku koju je poslao pacijent (roditelj)
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljen kod liječnika, liječnik je primio poruku
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja na čiju poruku želi odgovoriti.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik bira jednu od poruka koju je primio od roditelja.
  4. Sustav liječniku vraća pregled poruke.
  5. Liječnik odabere opciju "Odgovori" i sastavlja svoj odgovor.
  6. Liječnik odabere opciju "Pošalji".
  7. Sustav pošalje roditelju poruku i vraća liječnika na pregled poruka s profilom roditelja.
- **Opis mogućih odstupanja:**
  - 6.a Liječnik nije odabrao opciju "Pošalji".
    1. Odgovor neće biti poslan i sustav vraća roditelja na pregled poruka s profilom roditelja.

#### UC28 - Naruči roditelja na specijalistički pregled

- **Glavni sudionik:** Liječnik obiteljske medicine
- **Cilj:** Naručiti roditelja na specijalistički pregled
- **Sudionici:** Baza podataka, roditelj
- **Preduvjet:** Liječnik je prijavljen u sustav, roditelj je prijavljen kod liječnika
- **Opis osnovnog tijeka:**
  1. Liječnik na početnoj stranici iz popisa roditelja prijavljenih kod njega bira roditelja kojeg želi naručiti na specijalistički pregled.
  2. Sustav liječniku vraća pregled poruka s profilom roditelja.
  3. Liječnik odabere opciju "Nova poruka".
  4. Sustav liječniku vraća prozor u kojem liječnik može odabrati vrstu bolesti te dodati napomenu.
  5. Liječnik kao naslov bira "Specijalistički pregled", odabere vrstu bolesti te dodaje napomenu ako to želi.
  6. Liječnik odabere opciju "Pošalji".
  7. Sustav pošalje poruku roditelju i vraća liječnika na pregled poruka s profilom roditelja.
- **Opis mogućih odstupanja:**
  - 6.a Liječnik odabere opciju "Zatvori".
    1. Roditelj neće biti naručen na pregled i sustav vraća liječnika na pregled poruka s profilom roditelja.

#### UC29 - Pregledaj sve postojeće osobe upisane u sustav

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati sve roditelje i djecu koji su već upisani u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Opis osnovnog tijeka:**
  1. Administrator nakon prijave u sustav ima pregled liste svih osoba prijavljenih u sustav (ime, prezime, OIB).
- **Opis mogućih odstupanja:**
  - 1.a Administrator još nije nijednu osobu prijavio u sustav.
    1. Lista prijavljenih je prazna.

#### UC30.1 - Prijavi novog roditelja u sustav

- **Glavni sudionik:** Administrator
- **Cilj:** Prijaviti novu osobu u sustav

- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Opis osnovnog tijeka:**
  1. Administrator na početnoj stranici bira opciju "Dodaj roditelja".
  2. Sustav administratoru vraća novi prozor u kojem se mogu upisati podaci roditelja.
  3. Administrator upisuje ime, prezime, OIB i datum rođenja roditelja.
  4. Administrator odabere opciju "Spremi".
  5. Baza podataka se ažurira.
  6. Sustav vraća administratora na popis svih prijavljenih osoba.
- **Opis mogućih odstupanja:**
  - 4.a Administrator odabere opciju "Odustani".
    1. Roditelj neće biti dodan i sustav vraća administratora na popis prijavljenih osoba.

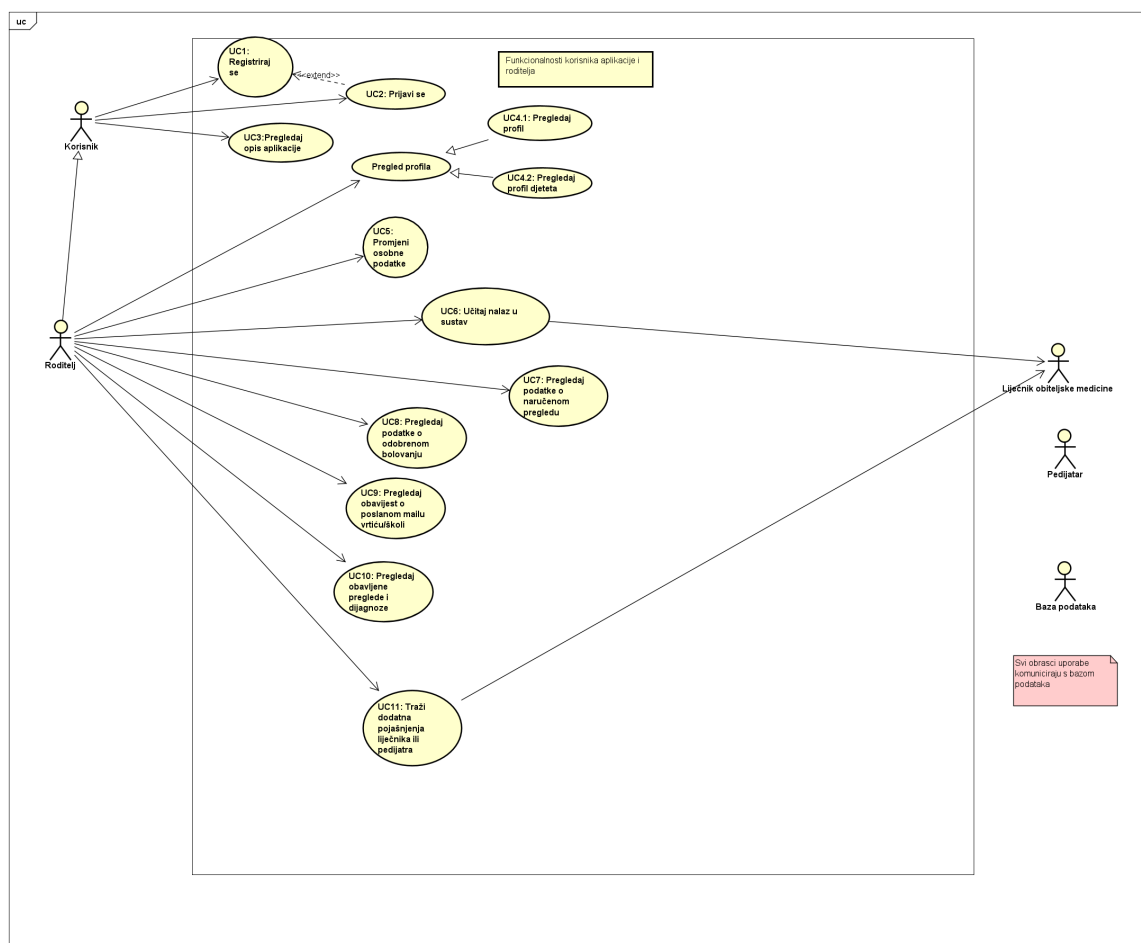
#### UC30.2 - Prijavi novo dijete u sustav

- **Glavni sudionik:** Administrator
- **Cilj:** Prijaviti novu osobu u sustav
- **Preduvjet:** Administrator je prijavljen, roditelj djeteta kojeg dodajemo već postoji u sustavu.
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
  1. Administrator na početnoj stranici bira opciju "Dodaj dijete".
  2. Sustav administratoru vraća novi prozor u kojem se mogu upisati podaci djeteta.
  3. Administrator upisuje ime, prezime, OIB djeteta, OIB roditelja (iz liste postojećih OIB-a roditelja u sustavu) i datum rođenja djeteta.
  4. Administrator odabere opciju "Spremi".
  5. Baza podataka se ažurira.
  6. Sustav vraća administratora na popis svih prijavljenih osoba.
- **Opis mogućih odstupanja:**
  - 4.a Administrator odabere opciju "Odustani".
    1. Dijete neće biti dodano i sustav vraća administratora na popis prijavljenih osoba.

#### UC31 - Izmijeni osobne podatke osobe prijavljene u sustav

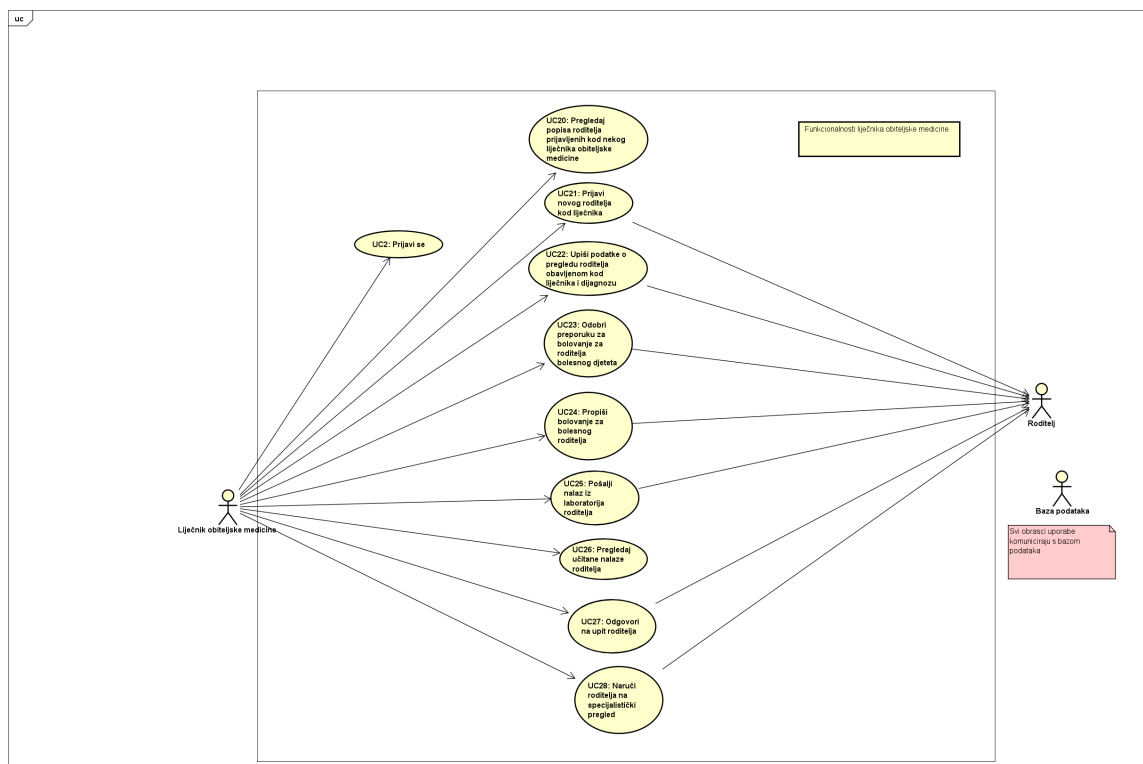
- **Glavni sudionik:** Administrator
- **Cilj:** Prijaviti novu osobu u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen, osoba je prijavljena u sustavu
- **Opis osnovnog tijeka:**
  1. Administrator na početnoj stranici bira iz liste prijavljenih osobu čije osobne podatke želi promijeniti.
  2. Sustav administratoru vraća prozor u kojem se nalaze osobni podaci osobe.
  3. Administrator u novootvorenom prozoru može mijenjati podatke osobe: ime, prezime, OIB, adresu, mail poslodavca/vrtića te može osobi pridijeliti liječnika/pedijatra.
  4. Administrator odabere opciju "Spremi".
  5. Baza podataka se ažurira.
  6. Sustav vraća administratora na popis prijavljenih osoba.
- **Opis mogućih odstupanja:**
  - 4.a Administrator odabere opciju "Odustani".
    1. Promjene neće biti pohranjene i sustav vraća administratora na popis prijavljenih osoba.

## Dijagrami obrazaca uporabe



Slika 3.1: UML dijagram koji opisuje obrasce uporabe korisnika i roditelja

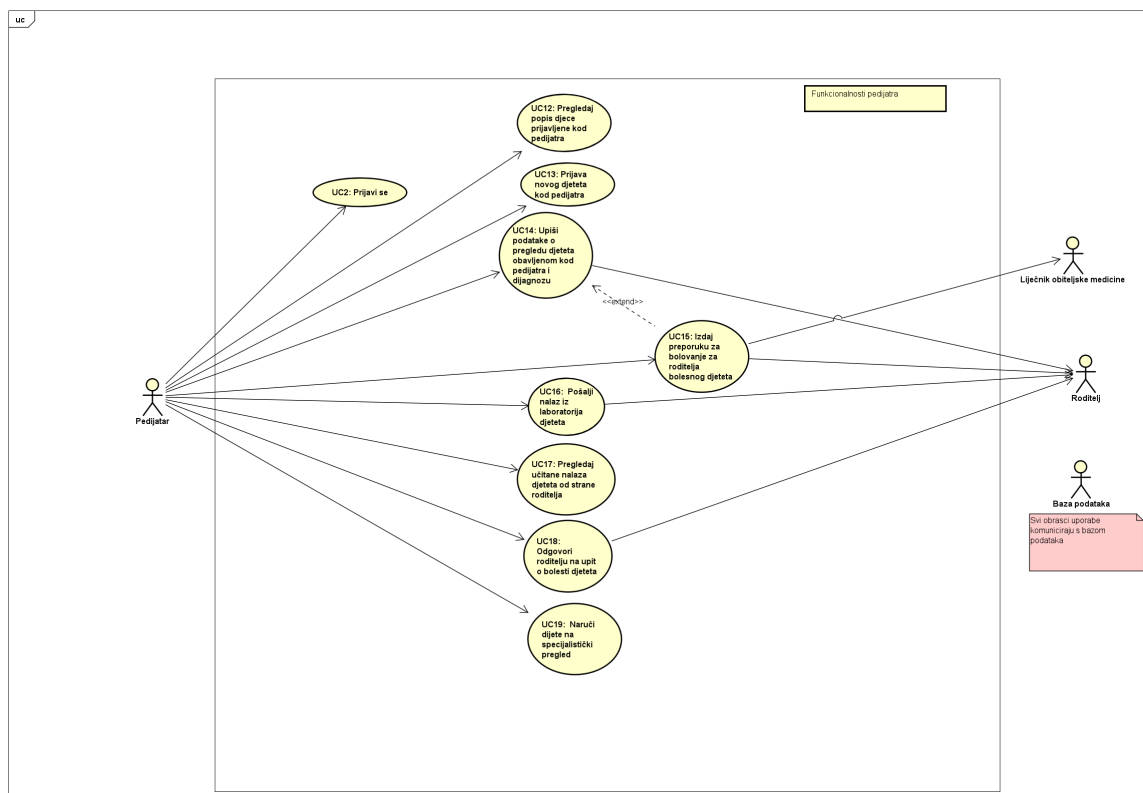
Referenciranje slike 3.1 u tekstu.



Slika 3.2: UML dijagram koji opisuje obrasce uporabe liječnika

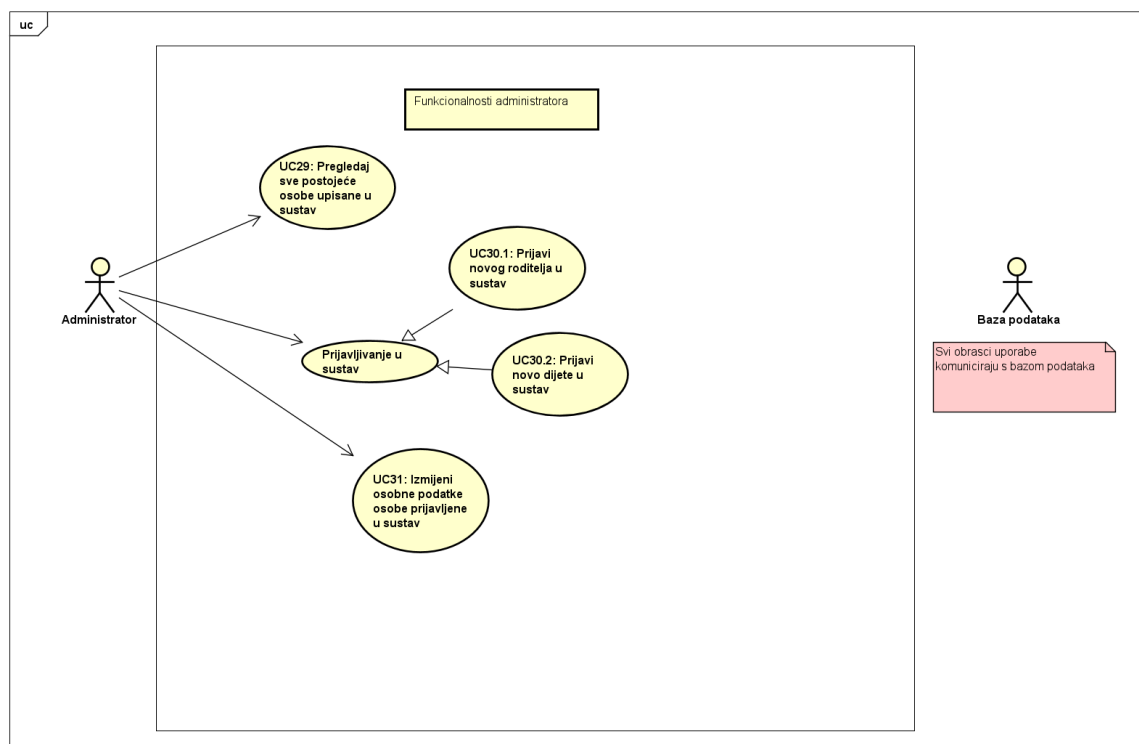
Referenciranje slike 3.2 u tekstu.





Slika 3.3: UML dijagram koji opisuje obrasce uporabe pedijatra

Referenciranje slike 3.3 u tekstu.



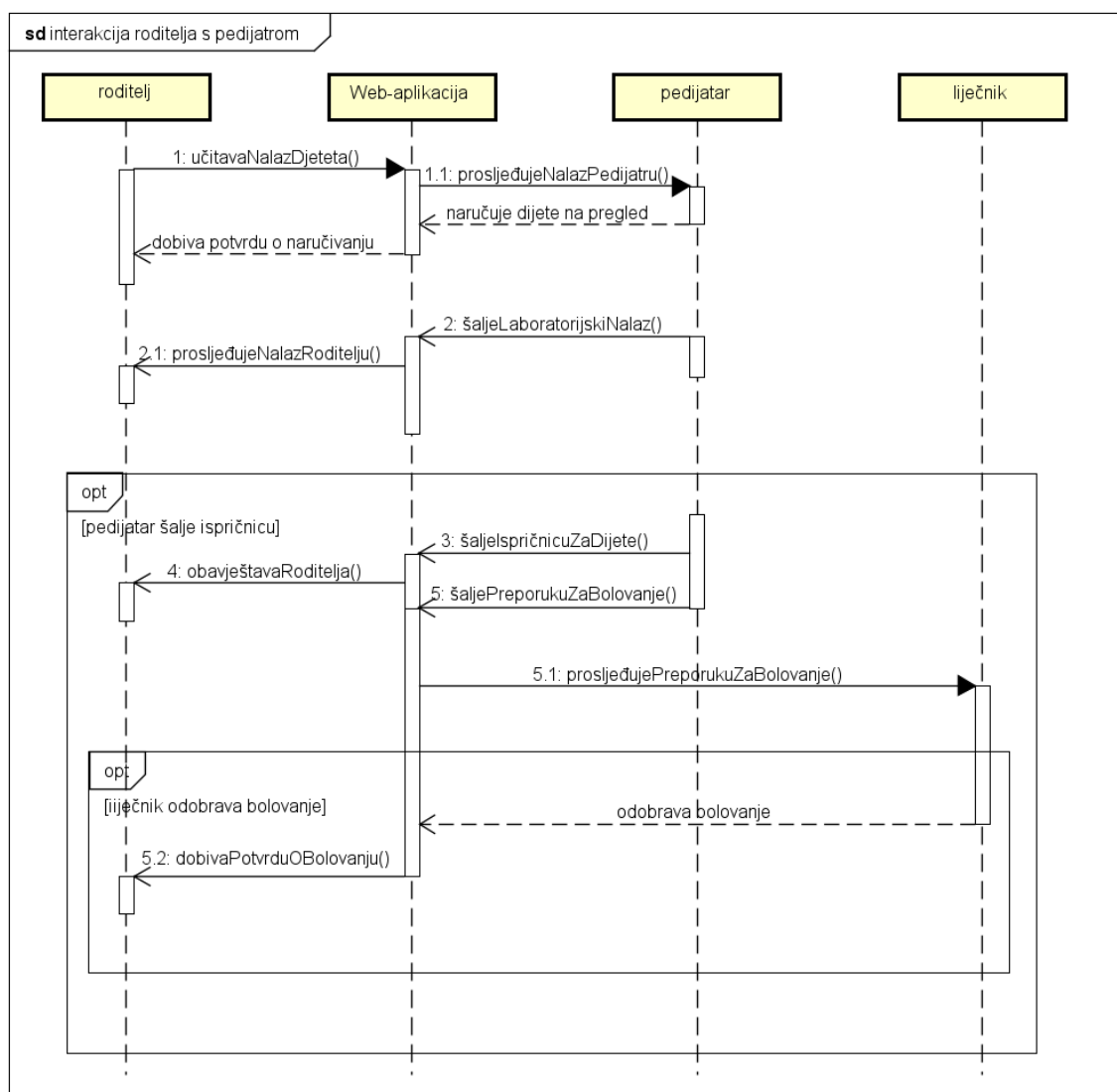
Slika 3.4: UML dijagram koji opisuje obrasce uporabe administratora

Referenciranje slike 3.4 u tekstu.

### 3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC6 - Učitaj nalaz u sustav, UC14 - Upiši podatke o pregledu djeteta obavljenom kod pedijatra i dijagnoza, UC15 - Izdaj preporuku za bolovanje za roditelja bolesnog djeteta, UC16 - Pošalji nalaz iz laboratorija djeteta, UC17 - Pregledaj učitane nalaze djeteta od strane roditelja, UC23 - Odobri preporuku za bolovanje za roditelja bolesnog djeteta

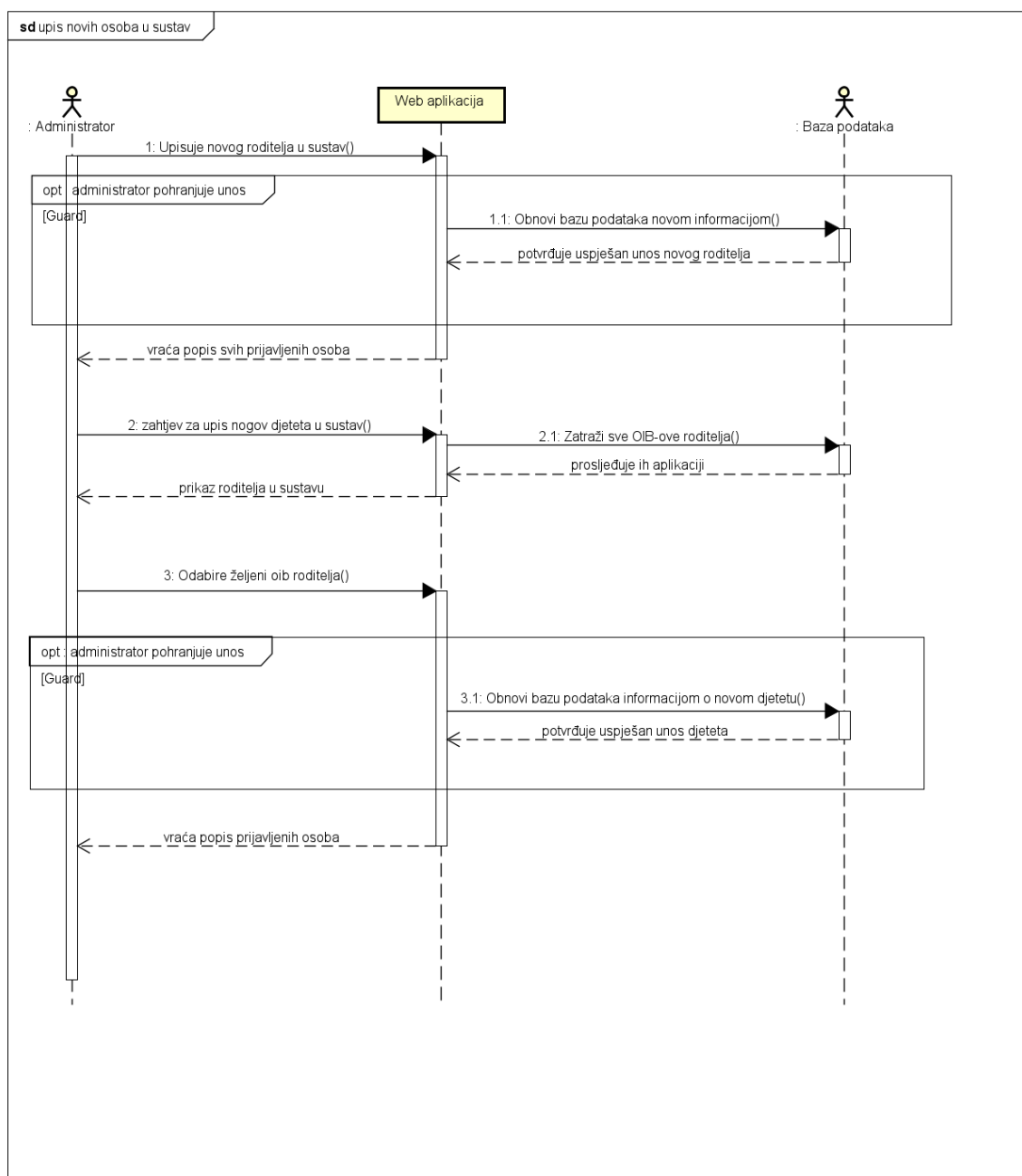
Ulogirani roditelj učitava nalaz svog bolesnog djeteta u web aplikaciju koja ga proslijeđuje pedijatru koji dijete naručuje na pregled te nakon pregleda roditelju šalje laboratorijski nalaz te po mogućnosti šalje ispričnicu za školu/vrtić roditelju te preporuku za bolovanje liječniku. Ako liječnik odobri bolovanje roditelju, preko aplikacije mu pošalje potvrdu.



Slika 3.5: Sekvencijski dijagram koji opisuje osnovnu mehaniku naručivanja djeteta i roditelja na pregled

**Obrazac uporabe UC30.1 - Prijavi novog roditelja u sustav, UC30.2 - Prijavi novo dijete u sustav**

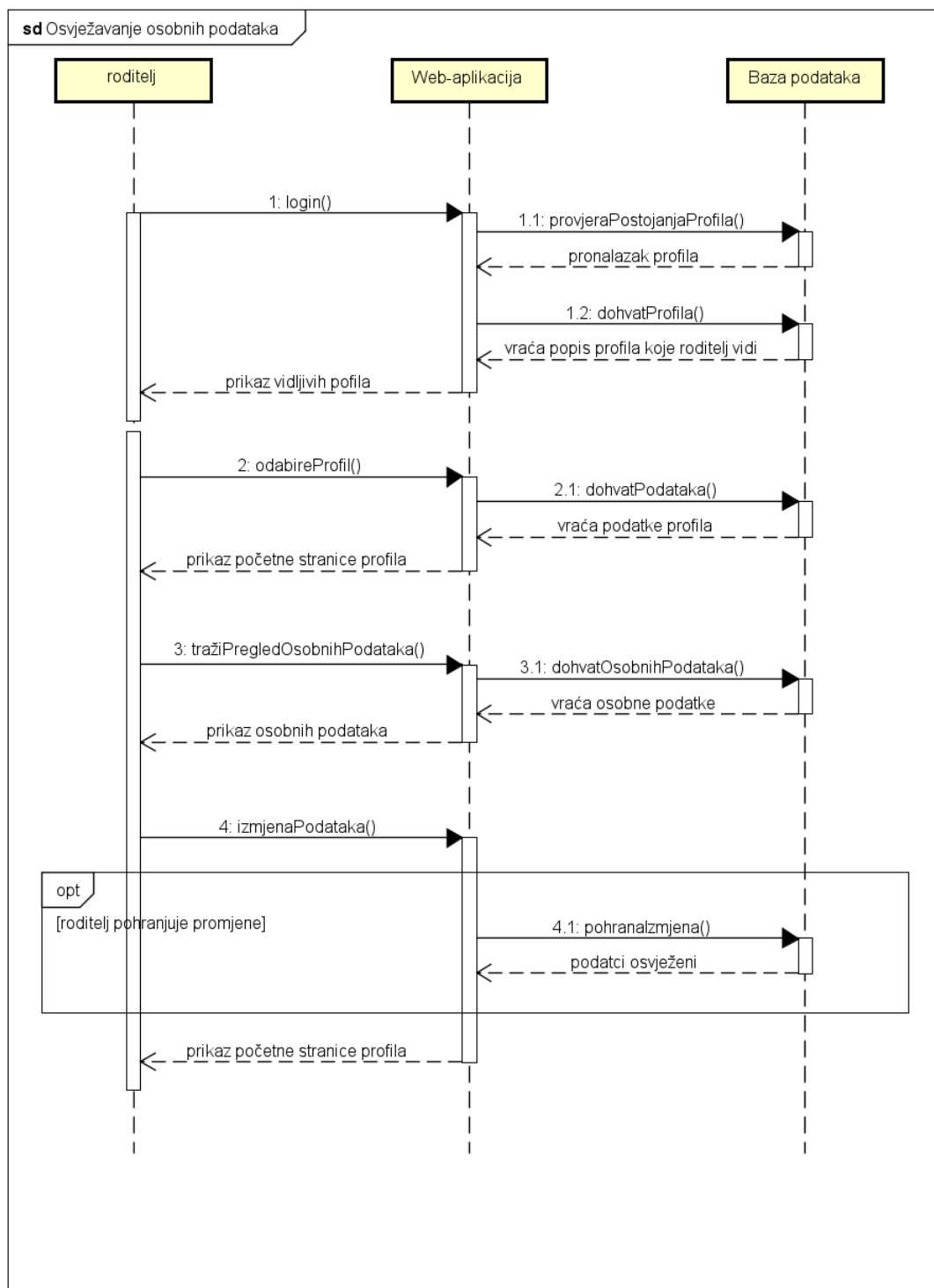
Administrator se ulogirava u sustav i na početnoj stranici bira opciju dodaj roditelja te upisuje njegovo ime, prezime, OIB i datum rođenja. Pritiskom na opciju dodaj u bazu podataka se taj roditelj dodaje. Potom Administrator pokuša upisati novo dijete upisom njegovog imena, prezimena OIB-a, datuma rođenja djeteta te iz liste koja mu se aplikaciji proslijedi iz baze podataka odabere OIB roditelja djeteta kojeg želi upisati u sustav. Potom odabere opciju Dodaj te se baza podataka obnovi.



Slika 3.6: Sekvencijski dijagram za UC30.1 i UC30.2

**Obrazac uporabe UC5 - Promjeni osobne podatke**

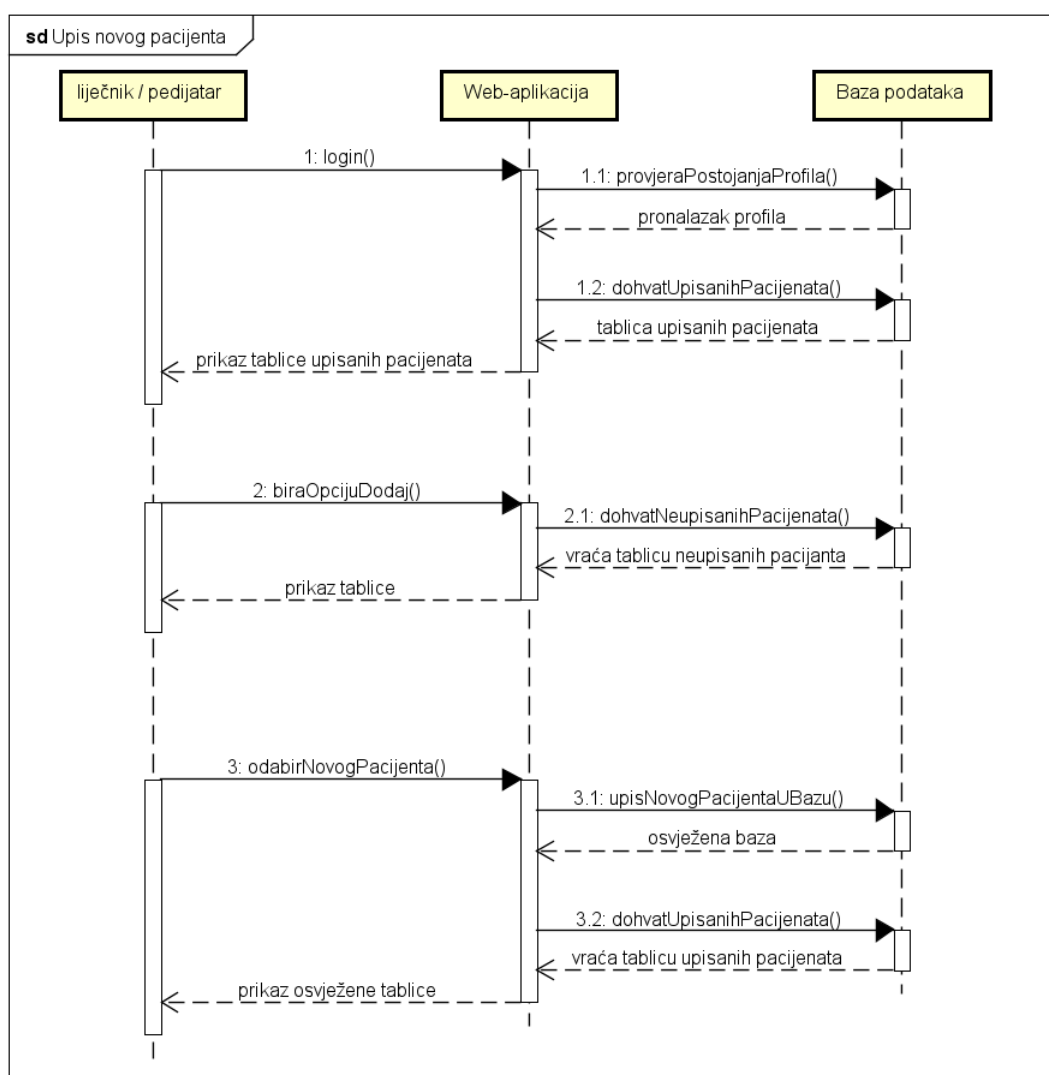
Klijent se ulogirava u sustav unosom OIB-a i korisničke lozinke. Nakon provjere postoji li korisnički profil s danim podacima u bazi, popis dostupnih profila se dohvaća iz baze i ispisuje korisniku. Korisnik odabire profil (vlastiti ili djetetov) te poslužitelj dohvaća podatke iz baze i ispisuje korisnički profil. Odabirom opcije "Pregled osobnih podataka" poslužitelj će iz baze dohvatiti osobne podatke korisnika te ih ispisati na ekran. Korisnik potom uređuje podatke te odabirom opcije "spremi" promjene pohranjuje u bazu. Preglednik će zatim dohvatiti početnu stranicu profila i prikazati ju korisniku.



Slika 3.7: Sekvencijski dijagram za UC5

### Obrazac uporabe UC13 - Prijavi novo dijete kod pedijatra, UC21 - Prijavi novog roditelja kod liječnika

Liječnik (pedijatar) se ulogirava u sustav unosom OIB-a i korisničke lozinke. Preglednik poziva bazu i provjerava postoji li korisnički profil u bazi nakon čega iz baze dohvaća upisane pacijente dotičnog korisnika te ih ispisuje na ekran. Korisnik odabire opciju "Prijavi novo(g) dijete/roditelja". Preglednik iz baze dohvaća popis neupisanih pacijenata te ih ispisuje na ekran. Odabirom pacijenta preglednik u bazu upisuje novog pacijenta liječniku/pedijatru, dohvaća osvježenu tablicu upisanih pacijenata te korisniku ispisuje početnu stranicu s upisanim pacijentima.



Slika 3.8: Sekvencijski dijagram za UC13 i UC21



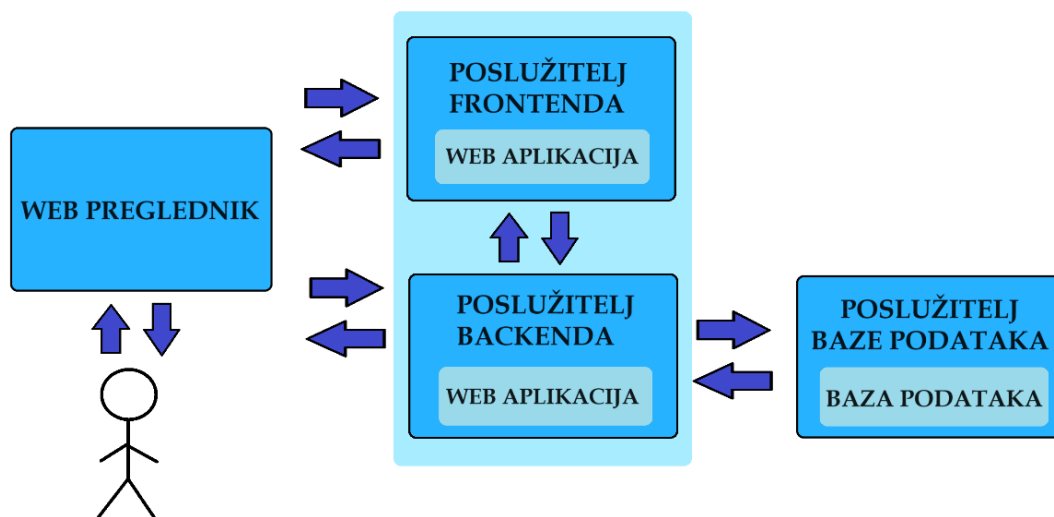
## 3.2 Ostali zahtjevi

- Responzivnost stranice (prilagođena za rad na mobilnom uređaju, tabletu, PC-ju)
- Aplikacija mora biti jednostavna za korištenje
- Sučelje je pregledno i intuitivno
- Sustav treba omogućiti istovremeni rad više korisnika
- Korisnici stranici pristupaju pomoću OIB-a i korisničke lozinke
- Korisnička lozinka mora sadržavati minimalno 5 znakova kako bi bila valjana
- Aplikacija je na hrvatskom jeziku te podržava unos i prikaz znakova hrvatske abecede
- Sustav treba biti implementiran koristeći objektno orijentirane jezike

## 4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na nekoliko podsustava:

- Poslužitelj frontenda
- Frontend (prvi dio web aplikacije)
- Poslužitelj backenda
- Backend (drugi dio web aplikacije)
- Baza podataka



Slika 4.1: Arhitektura sustava

Web preglednik je program koji omogućuje korisnicima pregled web sadržaja, a to uključuje web stranice i njihov raznolik sadržaj (slike, dijagrami...). Web preglednik je i okolina u kojoj se mogu izvršiti skripte (potrebno npr. kod client-side renderinga). Jedna od glavnih funkcionalnosti web preglednika je slanje zahtjeva prema web poslužiteljima.

Frontend poslužitelj je poslužitelj koji komunicira s web preglednikom klijenta preko HTTP (engl. *Hyper Text Transfer Protocol*) protokola. Na frontend poslužitelju se nalazi frontend dio web aplikacije čije dijelove frontend poslužitelj vraća klijentu na zahtjev (npr. pri početnom otvaranju stranice).

Backend poslužitelj je poslužitelj na kojem se nalazi backend dio aplikacije. Backend poslužitelj omogućuje slanje i primanje zahtjeva backend dijelu aplikacije. Backend poslužitelj preko HTTP protokola komunicira s frontendom ili web preglednikom klijenta. Ovaj poslužitelj komunicira i s poslužiteljem baze podataka preko TCP (eng. *Transmission Control Protocol*) protokola.

Web aplikacija dijeli se na frontend i backend. Zadaća frontend dijela je oblikovanje prikaza podataka, a potrebne podatke može zatražiti od backenda. Zadaća backend dijela je dohvaćanje podataka iz baze te njihova obrada i prosljeđivanje frontendu u određenom obliku. Komunikacija između frontenda, backenda, baze i web preglednika odvija se preko zahtjeva.

Baza podataka sadrži sve informacije o korisničkim računima, korisnicima te porukama koje korisnici šalju unutar web aplikacije.

Razvojni okvir koji smo odabrali za razvijanje backend dijela aplikacije jest Spring (programski jezik Java). Za razvoj frontend dijela aplikacije odlučili smo se za razvojni okvir React (programski jezik JavaScript). Odabrano razvojno okruženje je IntelliJ IDEA. Baza podataka napravljena je koristeći PostgreSQL.

Za razvoj backenda se koristio razvojni okvir Spring MVC (Model - Pogled - Nadglednik) koji je dio Springa. Backend je po tome organiziran u 3 sloja koja međusobno komuniciraju:

- Controller
- Service
- Repository

Controller predstavlja vanjsko sučelje web aplikacije. Ovaj sloj prihvata nadolazeće HTTP zahtjeve, poziva potrebne usluge koristeći razrede Service te vraća odgovor klijentu.

Service ostvaruje funkcionalnosti koje nudi aplikacija, predstavlja logiku programa. Ostvaruje usluge, a pritom može koristiti i druge Service razrede te može zatražiti podatke od sloja Repository.

Repository omogućuje komunikaciju s bazom podataka te izvlačenje podataka iz nje. Korišteno je sučelje JpaRepository koje omogućuje da se s bazom komunicira bez SQL upita - umjesto njih se koriste i modeli domene podataka. To su razredi koji predstavljaju strukturu tablica baze (entities).

Prilikom korištenja Spring MVC-a, klijentska strana predstavlja pogled, Controller sloj predstavlja nadglednika, a model je ostvaren slojevima Service i Repository.

Arhitektura web aplikacije temelji se na single-page application principu kojeg omogućuje razvojni okvir React. Pri početnom otvaranju web aplikacije, web preglednik korisnika uputi zahtjev frontend serveru koji mu vraća sve odgovarajuće datoteke potrebne za prikaz stranice u pregledniku (koristi se princip client-side rendering). Ovisno o prosljeđenim podacima, web preglednik prilikom obrađivanja ulaznih podataka korisnika (klik na pojedini gumb, otvaranje nove stranice) šalje zahtjeve backend serveru (ako su potrebni određeni podaci za prikaz React komponente) ili frontend serveru (za dohvat datoteka potrebnih za prikaz novog dijela stranice).

## 4.1 Baza podataka

Za našu web aplikaciju koristiti ćemo relacijsku bazu podataka koja je industrijski standard te najjednostavniji način za rješenje našeg problema. Osnovni element baze je relacija čija su obilježja njeno ime i atributi. Glavna zadaća naše baze je spajanje njenih korisnika i sustava s korisnikom, bilo kroz poruke ili kroz razne obrasce. Postoji jedna tablica veza s nazivom Pregled. Entiteti ove baze podataka su:

- Osoba
- Poruka
- Bolest
- Bolnica

### 4.1.1 Opis tablica

**Osoba** Ovaj entitet sadrži sve informacije o pojedinoj osobi spremljenoj u aplikaciji. Budući da ovaj entitet modelira i građane i zdravstvene zaposlenike ima mnogo opcionalnih atributa. Atributi entiteta su: OIB, ime, prezime, lozinka, email

ustanove, datum rođenja, adresa stanovanja, administratorska prava, uloga, OIB roditelja i OIB zadanog doktora. Moguće uloge su: 'roditelj', 'dijete', 'pedijatar', 'doktor'. Email ustanove, datum rođenja, adresa stanovanja, lozinka, OIB roditelja i OIB doktora mogu biti prazni. Ovaj entitet je u *Many-to-One* vezi s entitetom Osoba s ulogom roditelj preko atributa rodOIB, *Many-to-One* vezi s entitetom Osoba s ulogom liječnik preko dokOIB, *One-to-Many* vezi s Poruka preko OIB-a gdje jedan entitet Poruka zahtjeva dva entiteta Osoba.

Osoba		
OIB	INT	OIB osobe
ime	VARCHAR	ime osobe
prezime	VARCHAR	prezime osobe
mail	VARCHAR	email ustanove (opcionalno)
datumRod	DATETIME	datum rođenja osobe (opcionalno)
adresa	VARCHAR	adresa stanovanja (opcionalno)
adminPrava	INT	administratorska prava (0 ili 1)
lozinka	VARCHAR	šifra računa osobe (opcionalno)
uloga	VARCHAR	uloga osobe
rodOIB	INT	OIB roditelja
dokOIB	INT	OIB doktora

**Poruka** Ovaj entitet sadrži sve informacije o porukama spremljenima u aplikaciji. Njegovi atributi su: id, OIBpoš, OIBpri, naslov, tijelo, prilog, tip, dijagnoza. Prilog i dijagnoza mogu biti prazne, ovisno o tipu poruke. Ovaj entitet je u *Many-to-One* vezi s entitetom Osoba preko OIB-a te su potrebna 2 različita OIB-a, *Many-to-One* vezi s entitetom Bolest preko id-a.

Poruka		
id	INT	identifikacijski ključ poruke
priOIB	INT	OIB pošiljatelja
pošOIB	INT	OIB primatelja

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Poruka		
naslov	VARCHAR	naslov poruke
tijelo	VARCHAR	tekstualni sadržaj poruke
prilog	VARCHAR	link na poslanu sliku unutar datotečnog sustava aplikacije (opcionalno)
tip	VARCHAR	tip poslane poruke (standardna,ispričnica itd.)
dijagnozaID	INT	id dijagnosticirane bolesti (opcionalno)

**Bolest** Ovaj entitet sadrži sve informacije o bolestima spremljenima u aplikaciji. Njegovi atributi su: id i naziv. Ovaj entitet je u *Many-to-Many* vezama s entitetom Bolnica id-a bolnice,*One-to-One* vezi s entitetom Poruka preko id-a bolesti.

Bolest		
idBolest	INT	identifikacijski ključ bolesti
naziv	VARCHAR	naziv bolesti

**Bolnica** Ovaj entitet sadrži sve informacije o bolnici spremljenima u aplikaciji. Njegovi atributi su: id,naziv i adresa. Ovaj entitet je u *Many-to-Many* vezama s entitetom Bolest preko id-a bolesti.

Bolnica		
idBolnica	INT	identifikacijski ključ bolnice
naziv	VARCHAR	naziv bolnice
adresa	VARCHAR	adresa bolnice

**Pregled** Ova tablica sadrži sve veze između entiteta Bolest i Bolnica spremljene u aplikaciji koji su u *Many-to-Many* vezi. Atributi tablice su: idBolest i idBolnica.

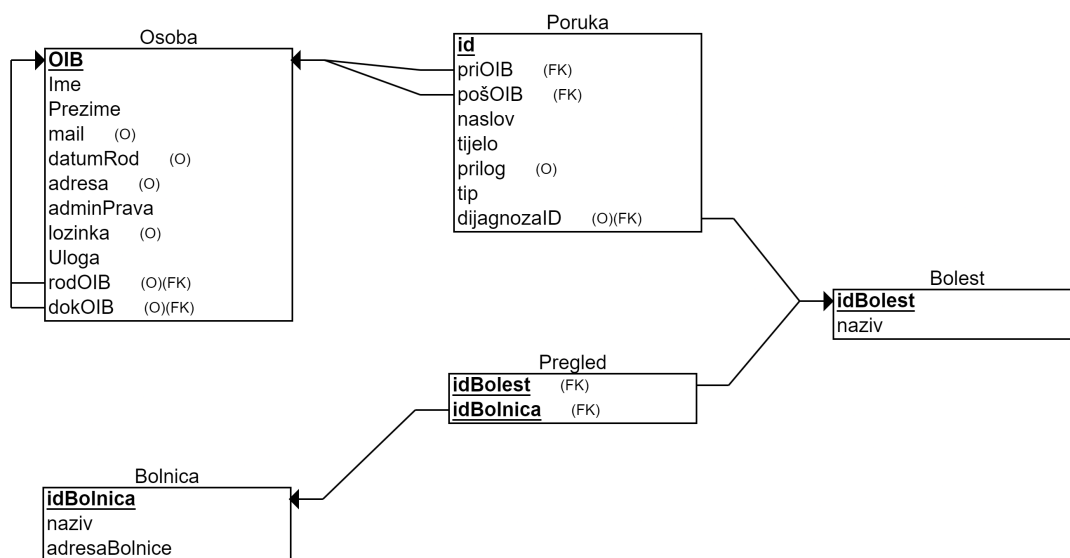
Pregled		
idBolest	INT	identifikacijski ključ bolesti

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Pregled		
idBolnica	INT	identifikacijski ključ bolnice

### 4.1.2 Dijagram baze podataka



Slika 4.2: Dijagram baze podataka

Dijagram razreda strukturni je dijagram koji prikazuje razrede, njihove atribute te veze između razreda. Na slici 4.4 prikazana je struktura DTO razreda. Na slici 4.3 prikazani su Controlleri, Service razredi te Repository razredi. Na slici 4.5 prikazana je struktura domain razreda (modela).

Slika 4.3: Dijagram Controllera, Servica i Repository-a

*Sučelje* **BolestService** opisuje operaciju kojom se pristupa svim entitetima **Bolest**.



*Razred **BolestServiceJpa*** predstavlja konkretnu implementaciju sučelja *BolestService* koja entitetu *Bolest* pristupa preko repozitorija (tipa *BolestRepository*) na kojeg pamti referencu.

*Sučelje **HospitalController*** služi za obradu zahtjeva za pregled bolnica. Pomoću GET zahtjeva se može dobiti lista svih bolnica.

*Sučelje **BolnicaRepository*** daje sučelje za pristup entitetu *Bolnica* u bazi podataka preko JPA.

*Sučelje **BolnicaService*** opisuje operaciju kojom se pristupa svim entitetima *Bolnica*.

*Razred **BolnicaServiceJpa*** predstavlja konkretnu implementaciju sučelja *BolnicaService* koja entitetu *Bolnica* pristupa preko repozitorija (tipa *BolnicaRepository*) na kojeg pamti referencu.

*Razred **PedijatarController*** služi za obradu zahtjeva za račun pedijatra. Ovime se omogućuje pregled i slanje poruka, dodavanje novih pacijenata te pregled već postojećih pacijenata. Također pomoću GET zahtjeva mogu se dobiti podaci o trenutnom pedijatru te listu djece bez dodijeljenog pedijatra.

*Razred **RegisterController*** služi za obradu zahtjeva za registraciju (register). Isto kao i razred *LoginController*, drži referencu na *OsobaService* radi pristupa bazi i na *PasswordEncoder* kako bi dobivene lozinke mogao hashirati prije spremanja u bazu. Članska funkcija *register* će na POST zahtjev s odgovarajućim podacima pokušati izvršiti registraciju novog korisnika.

*Razred **AdminController*** služi za obradu zahtjeva za račun admina. Ovo omogućuje korisnicima sa admin privilegijama da dodaju nove osobe i djecu u sustav, te dobiju listu svih osoba u sustavu. Također pomoću GET zahtjeva je moguće dobiti entitet osobe preko njenog OIB-a, te podatke trenutno ulogiranog korisnika. Pomoću POST zahtjeva mogu se ažurirati podaci o osobama u sustavu.

*Razred **LijecnikController*** služi za obradu zahtjeva za račun liječnika. Ovime se omogućuje pregled, slanje i brisanje poruka te pregled poruka poslanih pedijatrima, dodavanje novih pacijenata te pregled već postojećih pacijenata i odobravanje bolovanja. Također pomoću GET zahtjeva mogu se dobiti podaci o trenutnom liječniku te listu ljudi bez dodijeljenog liječnika.

*Razred **RoditeljController*** služi za obradu zahtjeva za račun odraslih. Ovime se omogućuje pregled i slanje poruka i pregled svoje djece. Također pomoću GET zahtjeva mogu se dobiti podaci o trenutno ulogiranom odraslom. Pomoću POST zahtjeva, osoba može ažurirati svoje i podatke svoje djece.

*Razred* **LoginController** služi za obradu zahtjeva za prijavu (login). Pamti referencu na apstraktni *OsobaService* kako bi mogao raditi operacije nad bazom *Osoba*. Članske funkcije ove klase izvršavaju se prilikom primanja HTTP zahtjeva, što vrijedi općenito za *Controller* klase (tj. njihove objekte) u Springu. Članska funkcija *login* će na POST zahtjev s odgovarajućim podacima obaviti prijavu korisnika. Iz tijela requesta izvlače se OIB i lozinka te se pomoću *Authentication Managera* provjerava njihova točnost. *Authentication Manager* koristi *Password Encoder* koji primljenu lozinku hashira te će se to usporedit s hashiranom lozinkom u bazi. Ako je sve u redu, postavlja se context i time je stvoren session. Informacije o sessionu pohranjuju se u obliku *UserDetails* objekta. Ovaj controller ima podržane i GET zahtjeve kojima se mogu vratiti OIB ili uloga trenutno ulogirane osobe.

*Razred* **LogoutController** služi za obradu zahtjeva za odjavu (logout). Primljenim GET zahtjevom na putanju `/api/logout` brišu se podaci o trenutnom session-u.

*Razred* **Security Helper** je razred koji sadrži pomoćnu funkciju kojom se iz requesta izvlače podaci o session-u. Preko njih se može pristupati odgovarajućem *UserDetails* objektu koji onda iz baze izvlači i vraća odgovarajući *Osoba* objekt (kao rezultat poziva funkcije).

*Sučelje* **OsobaRepository** daje sučelje za pristup entitetu *Osoba* u bazi podataka preko JPA.

*Sučelje* **OsobaService** opisuje operacije kojima se pristupa entitetu *Osoba*, ne oslanjajući se na konkretnu implementaciju (tj. na *Repository*), što je u skladu sa objektnom paradigmom i omogućuje primjenu dependency injectiona.

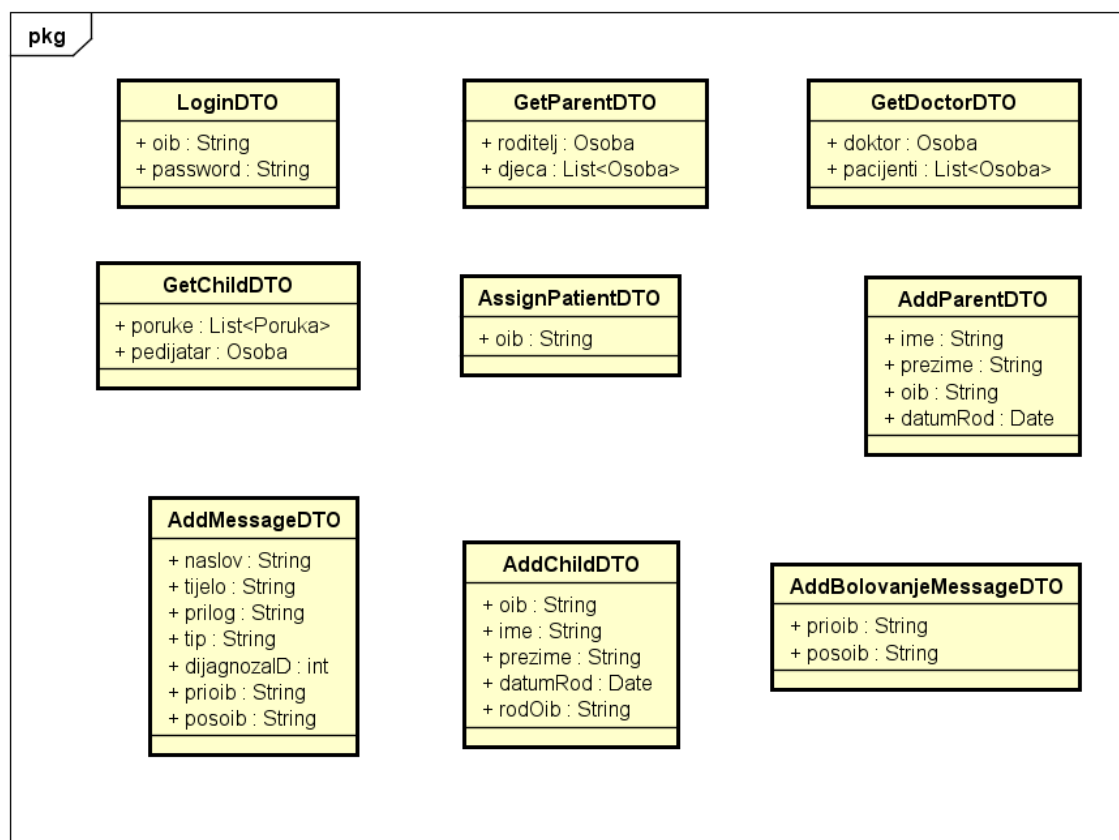
*Razred* **OsobaServiceJpa** predstavlja konkretnu implementaciju sučelja *OsobaService* koja entitetu *Osoba* pristupa preko repozitorija (tipa *OsobaRepository*) na kojeg pamti referencu.

*Sučelje* **SpecialistController** služi za obradu zahtjeva za pregled specijalista za danu bolest. Pomoću GET zahtjeva se može dobiti lista svih bolnica gdje se dana bolest može liječiti.

*Sučelje* **PregledRepository** daje sučelje za pristup entitetu *Bolnica* u bazi podataka preko JPA.

*Sučelje* **PregledService** opisuje operaciju kojom se pristupa svim entitetima *Bolnica* koji su povezani sa id-em dane bolesti.

*Razred* **PregledServiceJpa** predstavlja konkretnu implementaciju sučelja *PregledService* koja entitetu *Bolnica* pristupa preko repozitorija (tipa *PregledRepository*) na kojeg pamti referencu.



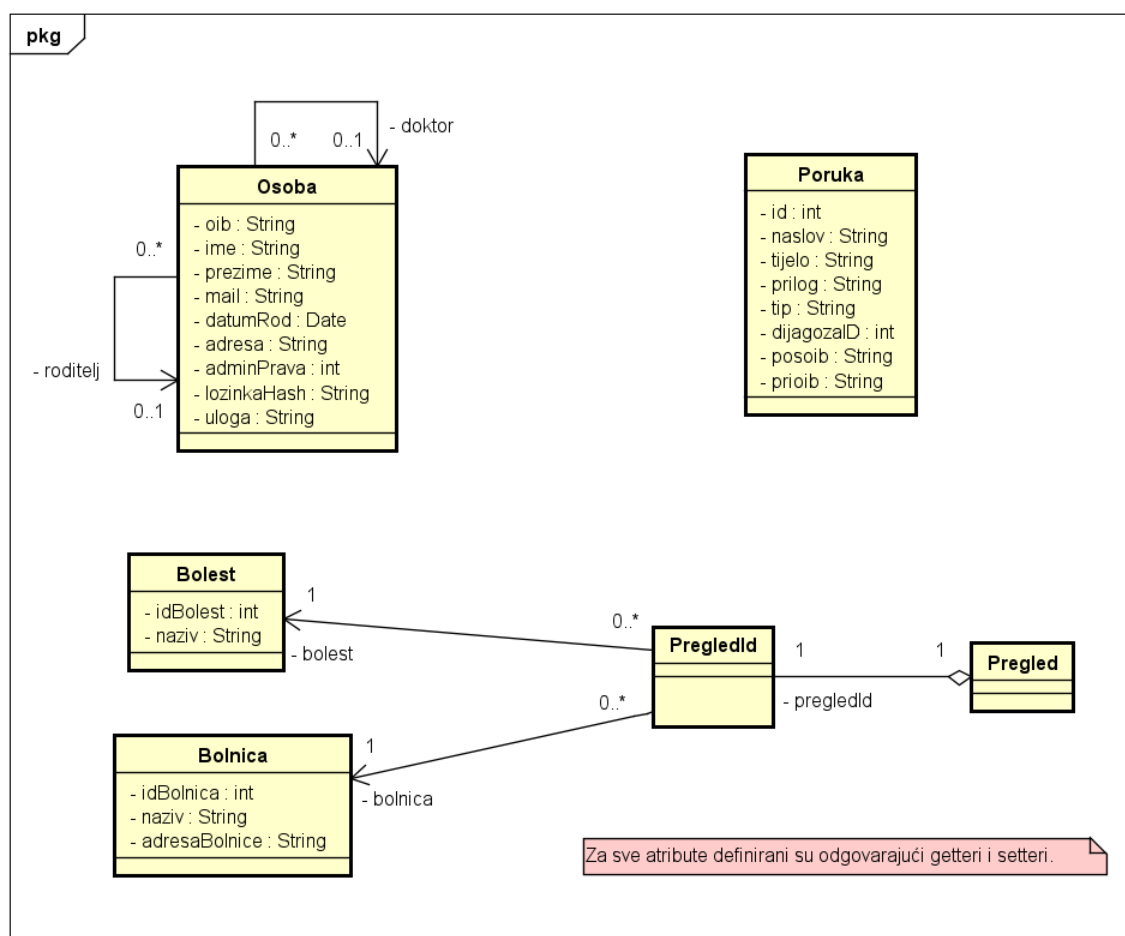
Slika 4.4: Dijagram DTO

Razredi DTO su pomoćni Data Transfer Object zapisi koji služe za pakiranje podataka dobivenih iz HTTP zahtjeva u Java objekt ili obratno. Koriste se u Controllerima.

Zapis **LoginDTO** iz zahtjeva izvlači OIB i lozinku kako bi se osoba mogla registrirati ili prijaviti. Njega prima npr. LoginController.

Zapis **GetParentDTO** se koristi za vraćanje podataka kao odgovor na zahtjev za dohvaćanje djece. Taj objekt se tada pretvara u JSON.

Razredi **GetDoctorDTO** i **GetChildDTO** koriste se također za vraćanje podataka kao odgovor na zahtjev, a razredi **AssignPatientDTO**, **AddParentDTO**, **AddMessageDTO**, **AddChildDTO** i **AddBolovanjeMessageDTO** za izvlačenje podataka iz zahtjeva.



Slika 4.5: Dijagram modela

Na dijagramu modela razredi predstavljaju entitete iz baze podataka zajedno s njihovim atributima te njihovim vezama.

Razred **Osoba** predstavlja entitet Osoba baze podataka. Osim glavnih atributa koji predstavljaju općenite podatke o osobi (oim, ime, prezime itd.) sadrži i referencu na objekt Osoba koji predstavlja roditelja osobe te na objekt Osoba koji predstavlja doktora (oboje mogu biti null).

Razred **Poruka** predstavlja poruku u bazi podataka.

Razred **Pregled** predstavlja povezanost između određene bolesti i bolnice u kojoj se ta bolest liječi. Ovaj razred koristi se pomoćnim razredom **PregledId** koji ne predstavlja entitet u bazi već je korišten zato što razred **Pregled** ima kompozitni ključ u bazi.

Razred **Bolest** predstavlja bolest.

Razred **Bolnica** predstavlja bolnicu.

Osim razreda prikazanih na dijagramima, korišteni su i drugi razredi od kojih su najbitniji:

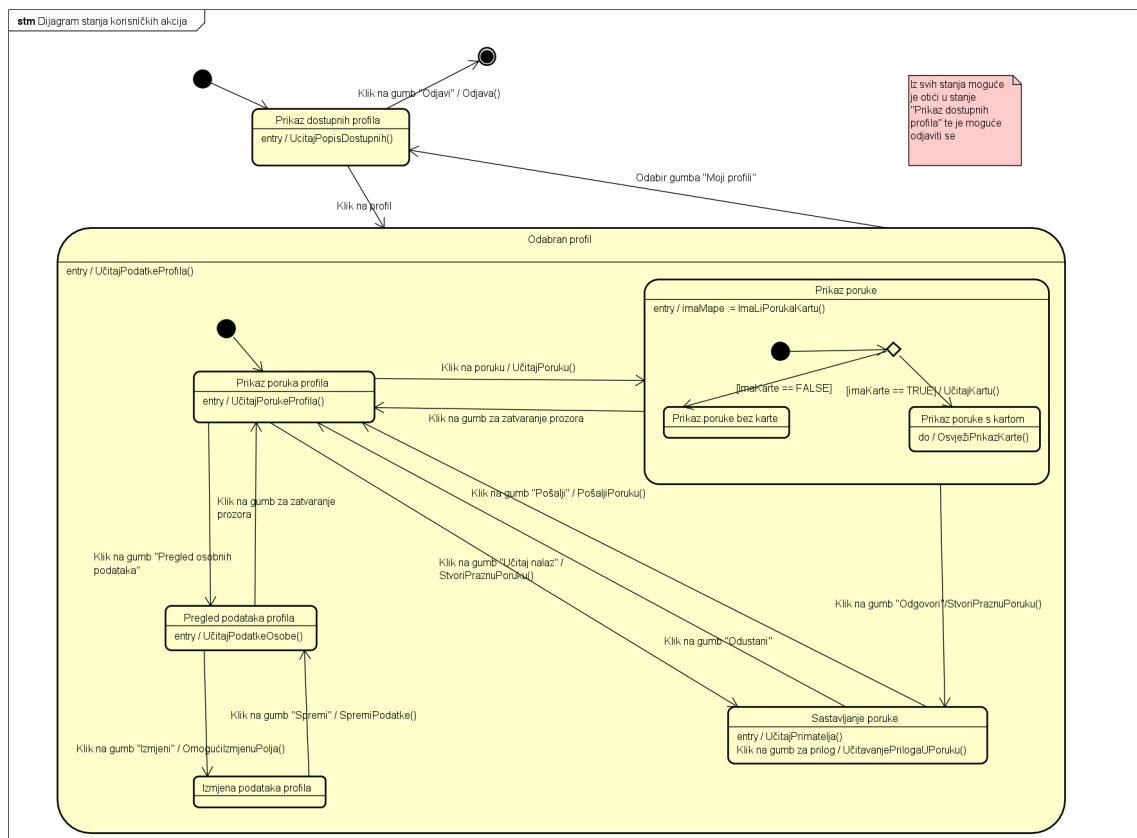
Razred **RestExceptionHandler** služi za hvatanje iznimki koje mogu biti bačene prilikom obrade HTTP zahtjeva u Controllerima i ovisno o iznimki vraća pravilan HTTP odgovor (npr. odgovor ne smije biti 500 ako se dogodi iznimka pri obradi zahtjeva jer je korisnika unio nepravilne podatke). Članska funkcija `handleInvalidPassword` će uhvatiti iznimke uzrokovane pogrešnim podacima za prijavu ili registraciju i vratiti HTTP odgovor sa smislenom porukom.

Razred **WebSecurityBasic** služi za konfiguriranje sigurnosnih postavki. U razredu je specificirano da su zahtjevi mapirani na putanjama `/api/login/*` i `/api/register/*` dozvoljeni neovisno o tome je li osoba prijavljena. Ostale putanje su dostupne samo prijavljenim osobama.

Razred **OzdraviBeApplication** je automatski generiran od strane Springa i sadrži ulaznu točku za izvršavanje programa.

### 4.3 Dijagram stanja

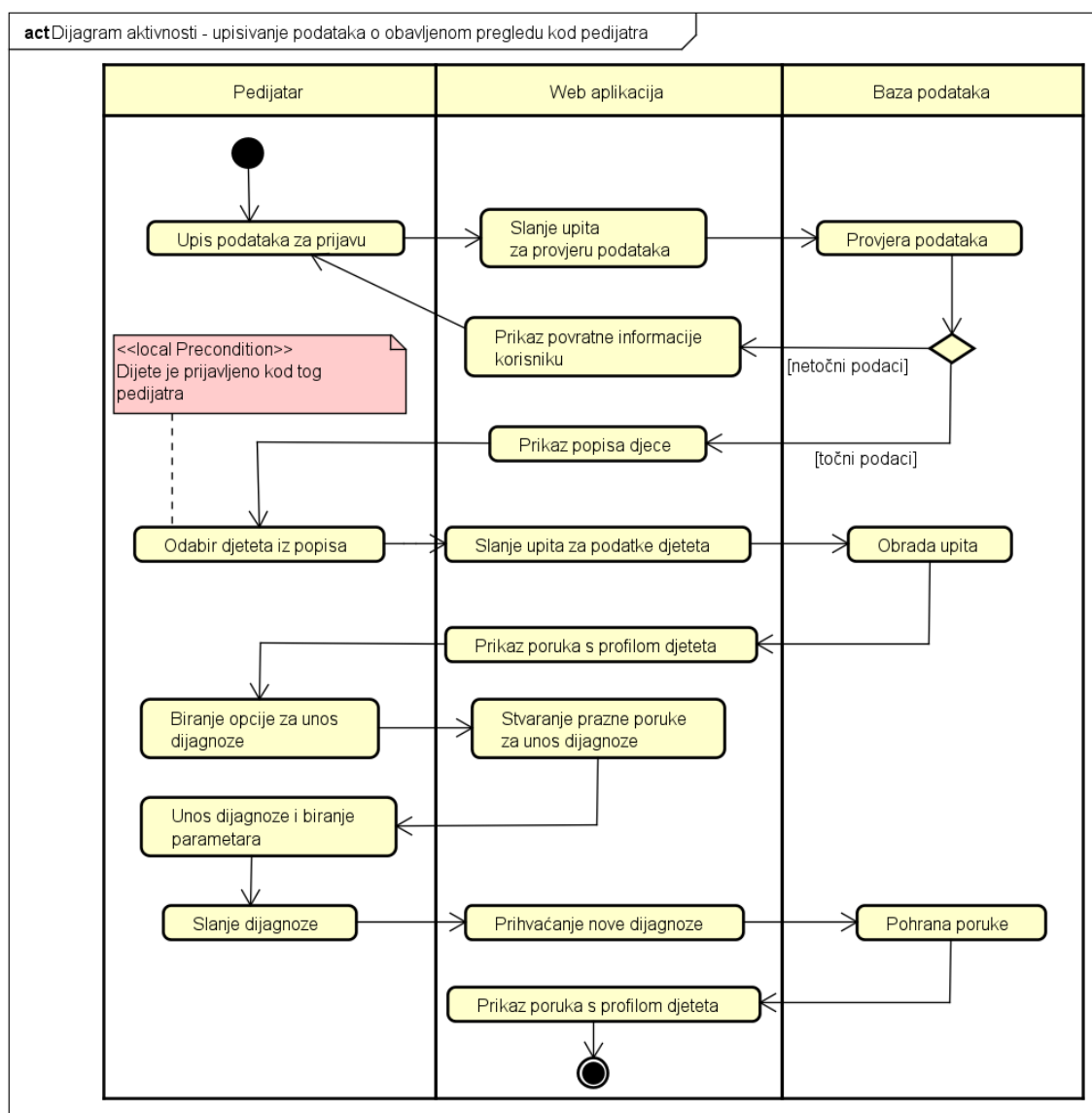
Dijagram stanja opisuje dinamičko ponašanje dijela sustava. Na slici 4.6 prikazan je dijagram stanja za prijavljenog roditelja. Roditelju se nakon prijave prikazuje popis profila kojima može pristupiti (profil roditelja i profili njegove djece). Odbirom pojedinog profila aplikacija roditelju prikaže poruke tog profila. Klikom na gumb za prikaz osobnih podataka profila, aplikacija roditelju pokaže iste te roditelj ima daljnju opciju promjene podataka. Ako roditelj na stranici s prikazanim porukama profila odabere pojedinu poruku, otvori se novi prozor u kojem se prikazuje poruka te karta ako se radi o obavijesti za naručeni pregled. Roditelj može na svaku poruku odgovoriti te mu se tada otvori novi prozor za sastavljanje poruke u kojem je automatski postavljen i primatelj poruke. Roditelju se prozor za sastavljanje poruke otvara i ako na stranici gdje su prikazane poruke tog profila klikne na gumb za učitavanje nalaza. Roditelj se u svakom trenutku može odjaviti ili vratiti na stranicu s popisom dostupnih profila.



Slika 4.6: Dijagram stanja

## 4.4 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za modeliranje upravljačkog i podatkovnog toka. Na slici 4.7 prikazan je dijagram aktivnosti za upisivanje podataka o obavljenom pregledu kod pedijatra. Nakon što pedijatar upiše podatke za prijavu u sustav, web aplikacija iz baze podataka izvlači potrebne informacije za provjeru ispravnosti podataka. Ako su podaci neispravni, prikazuje se poruka greške, a ako su ispravni web aplikacija prijavljenom pedijatru prikaže popis djece prijavljene kod njega. Pedijatar potom bira dijete za kojeg želi upisati pregled (preduvjet za to je da je dijete prijavljeno kod njega). Nakon odabira djeteta web aplikacija pedijatru prikaže poruke s profilom tog djeteta. Pedijatar sad može odabrati opciju za unos pregleda nakon čega se otvara novi prozor u kojem pedijatar može unijeti potrebne podatke. Kada je gotov s unosom podataka, pedijatar bira opciju za slanje dijagnoze. Web aplikacija novostvorenu poruku koja sadrži podatke o pregledu pohranjuje u bazu, a pedijatru se opet prikazu poruke s profilom djeteta.

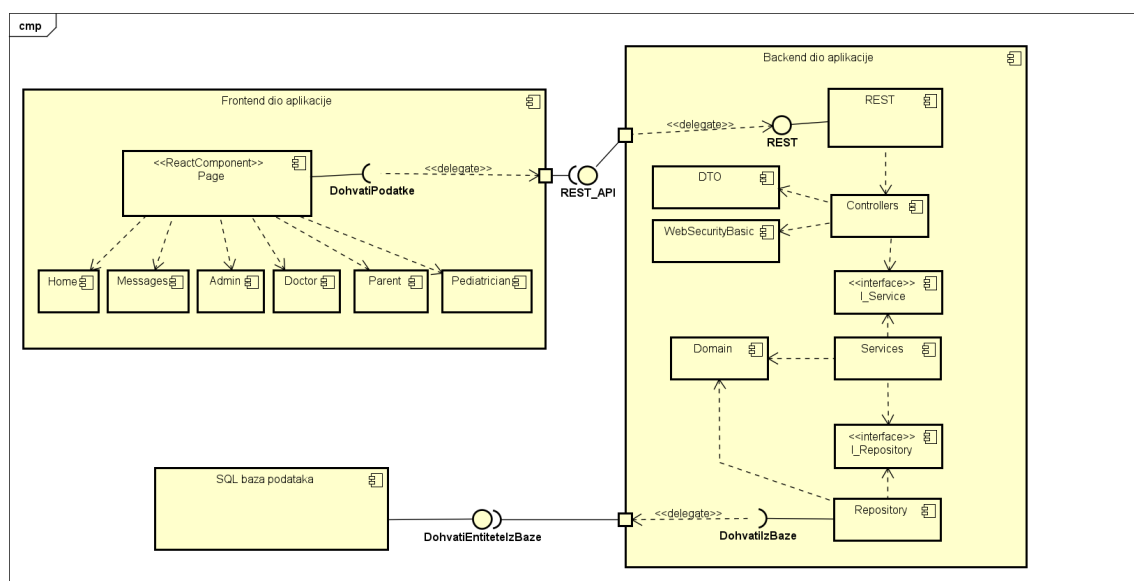


Slika 4.7: Dijagram aktivnosti



## 4.5 Dijagram komponenti

Dijagram komponenti strukturni je UML dijagram koji je dio specifikacije arhitekture programske potpore. Prikazuje organizaciju i međuovisnost između implementacijskih komponenata. Na slici 4.8 prikazan je dijagram komponenti aplikacije. React komponente frontenda preko REST zahtjeva šalju upite backend dijelu. Zahtjev prvo dolazi do Controllers razreda koji je zaštićen zahvaljujući WebSecurityBasic implementaciji. Data transfer object (DTO) omogućuje lagan pristup podacima koji su poslani u sklopu REST zahtjeva backendu. Controllers razredi preko Service implementacija mogu pozvati potrebne usluge. Service preko Repository-ja može pristupati podacima u bazi. Controllers razredi u sebi sadrže referencu na sučelje Service, a ne na konkretnu implementaciju. Tako i Service razredi u sebi sadrže referencu na sučelje Repository. To omogućuje tzv. dependency injection (Spring automatski spaja na odgovarajuću implementaciju). Radi lakšeg rada s entitetima iz baze, framework se automatski pobrine za object relational mapping. Domain klase predstavljaju strukture i attribute entiteta iz baze. SQL baza podataka nudi sučelje za dohvat podataka.



Slika 4.8: Dijagram komponenti

## 5. Implementacija i korisničko sučelje

### 5.1 Korištene tehnologije i alati

Za potrebe rada na projektu, tim je komunicirao preko aplikacija WhatsApp<sup>1</sup> i Discord<sup>2</sup>. Za izradu UML dijagrama, korišten je alat Astah<sup>3</sup>. Za upravljanje različitim verzijama datoteka korišten je sustav Git<sup>4</sup>. Udaljeni repozitorij projekta nalazi se na platformi Github<sup>5</sup>. Za izradu i testiranje baze korišten je sustav za upravljanje bazama podataka PostgreSQL<sup>6</sup>.

Kao integrirano razvojno okruženje (IDE) korišten je IntelliJ IDEA<sup>7</sup>. Ovaj IDE razvio je JetBrains. Koristi se prvenstveno za razvoj softvera napisanog u Javi, Kotlinu i drugim JVM (Java Virtual Machine) jezicima. Preko pluginova nudi podršku i za mnoge druge programske jezike kao što su Python, R, Julia itd. Za potrebe rada na projektu, tim je koristio studentske licence za navedeni IDE preko kojih je moguće koristiti podršku IntelliJ-a za razne radne okvire (npr. Spring, React).

Aplikacija *Ozdravi* napisana je koristeći radni okvir React<sup>8</sup> za *frontend* te radni okvir Spring<sup>9</sup> za *backend*. *React* je Javascript biblioteka otvorenog koda za izradu korisničkih sučelja. Održavaju ju Meta i zajednica programera i tvrtki. Ova biblioteka brza je i jednostavna za koristiti te se pomoću nje najčešće razvijaju single-page aplikacije. Velika prednost toga je što React omogućuje da se prilikom korištenja sučelja ponovno naslikaju (renderaju) samo dijelovi koji su se izmijenili. Sučelje izrađeno pomoću Reacta sastavljeno je od komponenti koje se mogu ponovno iskoristiti. *Spring* je besplatan radni okvir za razvoj aplikacija u Javi (i drugim jezicima). Vrlo je popularan zbog različitih modula koje nudi čime se znatno ubrzava stvaranje aplikacije. U sklopu izrade ove aplikacije korišteni su Spring Boot (razvoj aplikacije s minimalnom količinom konfiguracija), Spring

---

<sup>1</sup><https://www.whatsapp.com/>

<sup>2</sup><https://discord.com/>

<sup>3</sup><https://astah.net/>

<sup>4</sup><https://git-scm.com/>

<sup>5</sup><https://github.com/>

<sup>6</sup><https://www.postgresql.org/>

<sup>7</sup><https://www.jetbrains.com/idea/>

<sup>8</sup><https://react.dev/>

<sup>9</sup><https://spring.io>

Security (jednostavna konfiguracija zaštite aplikacije), Spring Data JPA (za lakšu komunikaciju s bazom podataka) i drugi. Za brzo početno podešavanje korištena je stranica Spring inicializr<sup>10</sup>.

---

<sup>10</sup><https://start.spring.io/>

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

Ispitivanje komponenti provedeno je koristeći radni okvir *JUnit 5*. Testovi se u tom radnom okviru pišu jednostavno kao zasebne metode. Ako testna metoda dođe do svog kraja, to znači da je test prošao, a ako se dogodi iznimka, test je pao. Osim toga, *JUnit 5* daje velik broj *assertion* metoda koje provjeravaju dane uvjete i bacaju iznimku (ruše test) ako isti nisu ispunjeni. Ovdje je navedeno 7 testova, od jednostavnijih prema složenijima.

**Prvi test** [*testFetchNonexistentOsoba*] provjerava baca li se odgovarajuća iznimka pri pokušaju dohvaćanja nepostojeće osobe po OIB-u iz baze podataka.

```
@Test
public void testFetchNonexistentOsoba() {
    assertThrows(EntityNotFoundException.class, () -> osobaService.fetch("0000000000"));
}
```

Slika 5.1: Prvi ispitni slučaj

**Drugi test** [*testFetchExistingOsoba*] provjerava dohvaća li se osoba odgovarajućeg imena i prezimena pri dohvaćanju postojeće osobe po OIB-u iz baze podataka.

```
@Test
public void testFetchExistingOsoba() {
    assertEquals("Iva", osobaService.fetch("12345678900").getIme());
    assertEquals("Ivić", osobaService.fetch("12345678900").getPrezime());
}
```

Slika 5.2: Drugi ispitni slučaj

**Treći test** [*testFindOsobaByParentOib*] provjerava funkcionira li metoda za dohvaćanje djece po OIB-u njihovog roditelja na način da usporedi jesu li dobiveni točni OIB-i.

```
@Test
public void testFindOsobaByParentOib() {
    assertEquals(
        List.of("55555666666", "75289754123", "76476024110"),
        osobaService.findByParent("01020304050").stream().map(Osoba::getOib).toList()
    );
}
```

Slika 5.3: Treći ispitni slučaj

**Četvrti test** [*testCreatePoruka*] provjerava može li se dodati validna poruka i hoće li funkcija za stvaranje poruke vratiti dodani objekt.

```
@Test
public void testCreatePoruka() {
    Poruka p = new Poruka();
    p.setNaslov("test");
    p.setTip("1");
    p.setTijelo("test poruka");
    p.setPosoib("01020304050");
    p.setPriloib("12345678900");

    assertEquals(p, porukaService.createPoruka(p));
}
```

Slika 5.4: Četvrti ispitni slučaj

**Peti test** [*testDeletePoruka*] provjerava funkcionira li metoda za brisanje poruke iz baze na način da provjeri vraća li ista objekt s točnim identifikatorom te hoće li nakon toga pokušaj dohvaćanja poruke sa (sada nepostojećim) identifikatorom uzrokovati iznimku jer objekta više nema.

```
@Test
public void testDeletePoruka() {
    Poruka p = new Poruka();
    p.setNaslov("test");
    p.setTip("1");
    p.setTijelo("test poruka");
    p.setPosoib("01020304050");
    p.setPrioib("12345678900");

    int id = porukaService.createPoruka(p).getId();
    assertEquals(p.getId(), porukaService.deletePoruka(id).getId());
    assertThrows(EntityNotFoundException.class, () -> porukaService.fetch(id));
}
```

Slika 5.5: Peti ispitni slučaj

**Šesti test** [*testPorukaBetweenPeople*] provjerava je li moguće ispravno dohvatiti poruke u razgovoru između dvoje ljudi tako da stvori novu poruku između dvoje ljudi i zatim provjeri je li ona među porukama koje vraća testirana metoda.

```
@Test
public void testPorukaBetweenPeople() {
    Poruka p = new Poruka();
    p.setNaslov("test");
    p.setTip("1");
    p.setTijelo("test poruka");
    p.setPosoib("01020304050");
    p.setPrioib("12345678900");

    int id = porukaService.createPoruka(p).getId();

    assertTrue(porukaService.findBetween("01020304050", "12345678900").stream().map(Poruka::getId).toList().contains(id));
}
```

Slika 5.6: Šesti ispitni slučaj

**Sedmi test** [*testFindUnassignedOsoba*] provjerava metodu koja služi za pronalaženje svih osoba bez doktora/pedijatra. U bazu se doda osoba bez doktora, provjeri se da testirana metoda vraća tu osobu, zatim se osobi doda doktor i konačno se opet poziva testirana metoda i provjeri se da ona ovaj puta ne vraća osobu (jer ona sada ima doktora).

```
@Test
public void testFindUnassignedOsoba() {
    Osoba o = new Osoba();
    o.setIme("ime");
    o.setPrezime("prezime");
    o.setOib("87631723612");
    o.setUloga("roditelj");

    if (osobaService.findByOib(o.getOib()).isPresent()) {
        osobaService.deleteOsoba(o.getOib());
    }

    osobaService.createOsoba(o);

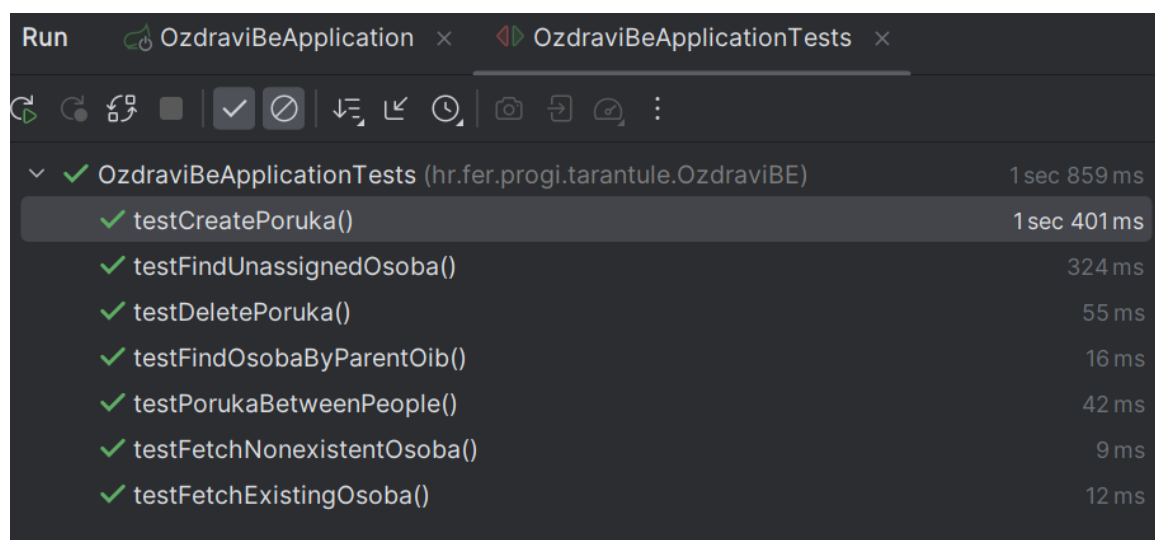
    assertTrue(osobaService.findUnassigned("roditelj").stream().map(Osoba::getOib).toList().contains(o.getOib()));

    o.setDoktor(osobaService.fetch("12345678900"));
    osobaService.updateOsoba(o);

    assertFalse(osobaService.findUnassigned("roditelj").stream().map(Osoba::getOib).toList().contains(o.getOib()));
}
```

Slika 5.7: Sedmi ispitni slučaj

Konačno, testovi se pokreću pomoću razvojnog okruženja i rezultati su ubrzo vidljivi:



Slika 5.8: Rezultati testova

### 5.2.2 Ispitivanje sustava

Testovi su izvršeni pomoću Selenium IDE ekstenzije za Google Chrome. Selenium IDE omogućava automatsko testiranje funkcionalnosti web stranica. Ovaj pristup

olakšava identifikaciju grešaka i poboljšava efikasnost procesa testiranja, pružajući preciznost i brzinu u otkrivanju potencijalnih problema.

Sam proces testiranja sastoji se od snimanja korisnikovih akcija koje se zatim spremaju kao test. Svaki test moguće je višetruko pokretati te mijenjati unesene parametre. U dokumentaciji je prikazano i opisano testiranje četiri osnovna obrasca uporabe (UC2, UC7, UC24, UC5).

**Prvi test** provjerava proces prijave u sustav već registriranog korisnika (UC2 - Prijava se). Tijek testiranja:

Ulazi:

1. Otvaranje web stranice u pregledniku.
2. Pritisak na gumb „Prijava“.
3. Pritisak polja „oib“.
4. Unos OIB-a postojećeg korisnika.
5. Pritisak polja „password“
6. Unos lozinke postojećeg korisnika.
7. Pritisak na gumb „prijava“.

Očekivani rezultat: prikaz stranice korisničkog profila.

Rezultat: očekivani rezultat je zadovoljen i aplikacija prolazi test.

Running 'testLogin2'	19:12:18
1. open on / OK	19:12:18
2. setWindowSize on 1133x1040 OK	19:12:19
3. click on linkText=prijava OK	19:12:19
4. click on name=oib OK	19:12:21
5. type on name=oib with value 01020304050 OK	19:12:21
6. click on name=password OK	19:12:21
7. type on name=password with value password333 OK	19:12:21
'testLogin2' completed successfully	19:12:21

Slika 5.9: Rezultati prvog ispitnog slučaja



**Drugi test** provjerava pregled podataka o naručenom specijalističkom pregledu (UC7).

Ulazi:

1. Odabir profila.
2. Odabir poruke s naslovom "Naručen pregled".
3. Pritisak kursorom na kartu, pomicanje kotačića na mišu.
4. Pritisak kursorom na jednu od označenih bolnica.

Očekivani rezultati:

Prikazuje se karta sa svim bolnicama iz baze podataka koje su u blizini stanovanja pacijenta. Na karti je moguće mijenjati visinu i poziciju pogleda te otvoriti ikonu s nazivom bolnice.

Rezultati: očekivani rezultat je zadovoljen i aplikacija prolazi test.

Running 'testMapUC7'

1. open on http://localhost:3000/parentInfo OK
2. setWindowSize on 1055x527 OK
3. click on linkText=Roko Luk OK
4. Trying to find css=li:nth-child(4) > #emailSender... OK
5. mouseDownAt on css=.leaflet-container with value 218.90625,12.48687744140625 OK
6. mouseMoveAt on css=.leaflet-container with value 218.90625,12.48687744140625 OK
7. mouseUpAt on css=.leaflet-container with value 218.90625,12.48687744140625 OK
8. click on css=.leaflet-container OK
9. click on css=.leaflet-control-zoom-in > span OK
10. mouseOver on css=.leaflet-control-zoom-in > span OK
11. mouseOut on css=.leaflet-control-zoom-in > span OK
12. click on css=.leaflet-control-zoom-out > span OK
13. mouseOver on css=.leaflet-control-zoom-out > span OK
14. mouseOut on css=.leaflet-control-zoom-out > span OK
15. mouseOver on linkText=- OK
16. mouseDownAt on css=.leaflet-container with value 167.90625,18.48687744140625 OK
17. mouseMoveAt on css=.leaflet-container with value 167.90625,18.48687744140625 OK
18. mouseUpAt on css=.leaflet-container with value 167.90625,18.48687744140625 OK
19. click on css=.leaflet-container OK
20. mouseDownAt on css=.leaflet-container with value 139.90625,131.48687744140625 OK

Slika 5.10: Rezultati drugog ispitnog slučaja

```
20. mouseDownAt on css=.leaflet-container with value 139.90625,131.48687744140625 OK
21. mouseMoveAt on css=.leaflet-container with value 139.90625,131.48687744140625 OK
22. mouseUpAt on css=.leaflet-container with value 139.90625,131.48687744140625 OK
23. click on css=.leaflet-container OK
'testMapUC7' completed successfully
```

Slika 5.11: Rezultati drugog ispitnog slučaja

**Treći test** provjerava akciju propisivanja bolovanja roditelju (UC24).

Ulazi:

1. Odabir profila roditelja.
2. Pritisak na „Nova poruka“.
3. Unos teksta poruke.
4. Označavanje opcije „Bolovanje roditelja“.
5. Pritisak gumba „Pošalji“.

Očekivani rezultati:

Prikaz profila roditelja s novom porukom naziva „dijagnoza“. Odabirom te poruke prikazuje se prikaz za čitanje poruke u kojoj vidimo sadržaj i na dnu tekst „Odobreno bolovanje“.

Rezultat: Očekivani rezultat je zadovoljen. Aplikacija je prošla test.

```
Running 'testBolovanjeUC24' 19:17:20
1. open on http://localhost:3000/doctor OK 19:17:21
2. setWindowSize on 1055x527 OK 19:17:21
3. click on linkText=01020304050 Ivan Ivanić OK 19:17:21
4. click on css=.Message OK 19:17:22
5. click on id=messageBody OK 19:17:22
6. type on id=messageBody with value Imate bolovanje. OK 19:17:23
7. click on css=.check OK 19:17:23
8. click on id=sendMessage OK 19:17:23
'testBolovanjeUC24' completed successfully 19:17:23
```

Slika 5.12: Rezultati trećeg ispitnog slučaja

**Četvrti test** provjerava izmjenu osobnih podataka na stranici roditelja/djeteta (UC 5).

Ulazi:

1. Odabir profila roditelja/djeteta.
2. Pritisak ikone za izmjenu podataka.
3. Odabir polja za izmjenu.
4. Unos novih podataka.
5. Pritisak na gumb "spremi.

Očekivani rezultati:

Prikazuje se stranica korisničkog profila. Po otvaranju prozora s osobnim podacima prikazuju se ažurirani podatci.

Rezultati: Očekivani rezultat je zadovoljen. Aplikacija je prošla test.

Running 'testPromjenaPodataka'

1. open on http://localhost:3000/parentInfo OK
2. setWindowSize on 1055x527 OK
3. click on linkText=Ivan Ivanić (my profile) OK
4. click on id=profileIcon OK
5. click on name=adresa OK
6. type on name=adresa with value Ilica 20, Zagreb OK
7. click on name=mail OK
8. type on name=mail with value peroperic@gmail.com OK
9. click on css=.addChild:nth-child(2) OK

'testPromjenaPodataka' completed successfully

Slika 5.13: Rezultati četvrtog ispitnog slučaja

Posljednjim testom prikazujemo kako sustav reagira ukoliko test ne prolazi. Test je proveden ponovnim pokretanjem prvog testa, no ovaj put s neispravnom lozinkom. Očekivani izlaz je neuspješna prijava korisnika te prikaz poruke o pogrešci.

Project: Tarantule\*

Executing ▾

X testLogin3\*

http://localhost:3000/

	Command	Target	Value
4	✓ click	name=oib	
5	✓ type	name=oib	01020304050
6	✓ click	name=password	
7	✓ type	name=password	kriviPassword
8	✓ click	css=.loginbutton:nth-child(2)	
9	X click	linkText=Roko Luk	

Command  //

Target

Value

Description

Runs: 1 Failures: 1

Log Reference

Running 'testLogin3'

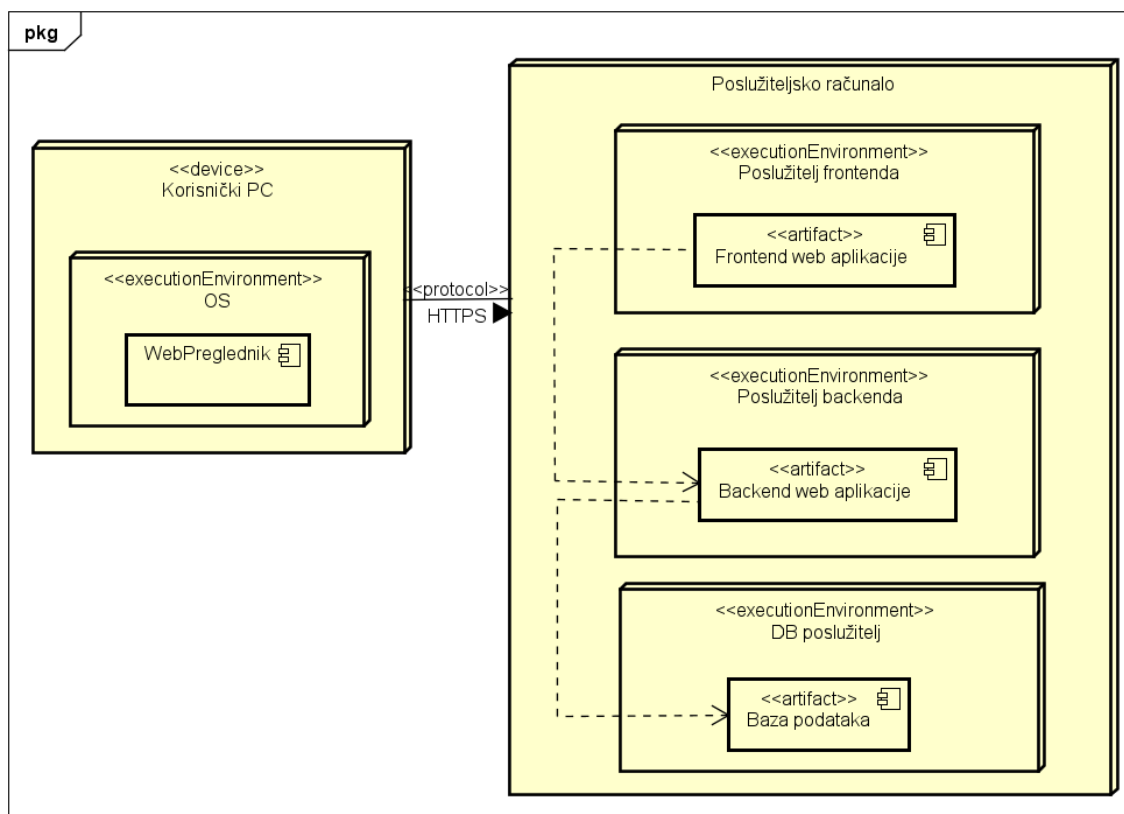
1. open on / OK
2. setWindowSize on 1133x1040 OK
3. click on linkText=prijava OK
4. click on name=oib OK
5. type on name=oib with value 01020304050 OK
6. click on name=password OK
7. type on name=password with value kriviPassword OK
8. click on css=.loginbutton:nth-child(2) OK
9. Trying to find linkText=Roko Luk... Failed:  
Implicit Wait timed out after 30000ms

'testLogin3' ended with 1 error(s)

Slika 5.14: Rezultati i parametri petog ispitnog slučaja

## 5.3 Dijagram razmještaja

Dijagram razmještaja je strukturni UML dijagram koji opisuje topologiju sustava i prikazuje odnos sklopovskih i programskih dijelova. Na slici 5.15 prikazan je specifikacijski dijagram razmještaja aplikacije. Na čvoru koji predstavlja korisnički uređaj nalazi se web preglednik (artefakt) korisnika. On preko HTTPS protoka komunicira s poslužiteljem. Poslužitelj pruža uslugu poslužitelja frontenda, backenda te baze podataka. Sve te usluge mogu međusobno komunicirati slanjem zahtjeva.



Slika 5.15: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

Za deploy aplikacije se koristi servis Render<sup>11</sup> koji nudi opciju korištenja besplatnih instanci na cloudu. Naravno, one nude malo procesorske snage i također imaju neka druga ograničenja ali je to sasvim dovoljno za ovaj projekt. Potrebno je napraviti tri instance na Renderu: na jednoj će biti upogonjen backend, druga će posluživati frontend, a treća PostgreSQL bazu podataka.

### Backend

Napraviti instancu tipa “web service”. Povezati ju s git repozitorijem u kojemu se nalazi projekt. Ime odabrati proizvoljno (ovdje *ozdraviapp-be*), regiju postaviti na geografski najbližu (Frankfurt), za granu izabrati master, a za korijenski direktorij onaj u kojemu se nalazi Java projekt za backend. Konačno, runtime će biti Docker, a zbog toga je potrebno napraviti još nekoliko koraka prije nego se aplikacija može deployati.

Kako bi se Docker mogao koristiti pri deployu, potrebno je u korijenski direktorij backend projekta dodati novi direktorij *docker* i u njemu Dockerfile. Njegov sadržaj vidljiv je na slici 5.16.

---

<sup>11</sup><https://render.com/>

```
FROM openjdk:21-jdk AS builder

COPY ../.mvn .mvn
COPY ../mvnw .
COPY ../pom.xml .
COPY ../src src
RUN chmod +x mvnw

RUN ./mvnw clean package -DskipTests

FROM openjdk:21-jdk

## package installation goes here

COPY --from=builder target/*.jar /app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Slika 5.16: Dockerfile

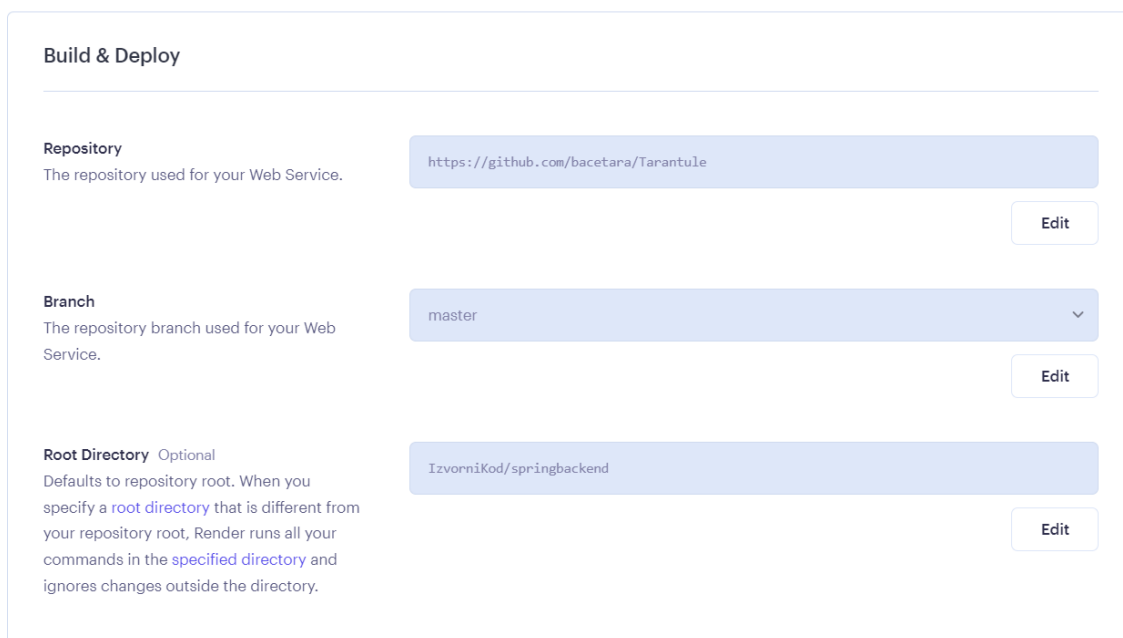
Ukratko, Dockerfile će napraviti okruženje u koje će kopirati izvorni kod backenda i sve potrebne datoteke te kompajlirati taj kod, a zatim u drugo okruženje kopirati dobiveni *.jar*. Na kraju, definira se da se pri pokretanju kontejnera pokreće

upravo taj `.jar` pomoću JVM. Koriste se Docker slike `openjdk:21-jdk`, zbog toga što je potreban JDK kako bi se kompajlirao Java projekt.

Sada se na Renderu može izabrati besplatna verzija instance te stvoriti web servis. Potrebno je namjestiti varijable okruženja, što se može pronaći u kartici “Environment”.

Spring za spajanje na bazu podataka koristi varijable `SPRING_DATASOURCE_URL`, `SPRING_DATASOURCE_NAME`, `SPRING_DATASOURCE_PASSWORD`. Sve one moraju biti postavljene na odgovarajuće vrijednosti kako bi aplikacija funkcionirala. Ove varijable se mogu definirati tek kad se stvori Render instance za posluživanje baze podataka.

Na kraju, servis se pokreće klikom na opciju “manual deploy - deploy latest commit”, no ovo se treba napraviti tek nakon namještanja baze podataka.



**Build & Deploy**

**Repository**  
The repository used for your Web Service.

`https://github.com/bacetara/Tarantule` Edit

**Branch**  
The repository branch used for your Web Service.

`master` Edit

**Root Directory** Optional  
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/springbackend` Edit

Slika 5.17: Postavke za backend instancu, prvi dio



The screenshot shows a configuration interface for a Docker service. It has three main sections, each with a text input field and an 'Edit' button.

- Dockerfile Path:** The text input contains 'IzvorniKod/springbackend/' followed by a sub-field containing 'docker/Dockerfile'. The description states: 'Path to your Dockerfile relative to the repository root. This is *not* relative to your Docker build context. For example, `./subdir/Dockerfile`.' The 'Edit' button is to the right.
- Docker Build Context Directory:** The text input contains 'IzvorniKod/springbackend/' followed by a sub-field containing '.'. The description states: 'Docker build context directory. This is relative to your repository root. Defaults to the root.' The 'Edit' button is to the right.
- Docker Command:** The text input is empty. The description states: '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`' The 'Edit' button is to the right.

Slika 5.18: Postavke za backend instancu, drugi dio

The screenshot shows the 'Environment Variables' configuration page. It includes a header, a description, a table of variables, and action buttons.

**Environment Variables**

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

Key	Value	
SPRING_DATASOURCE_PASSWORD	.....	
SPRING_DATASOURCE_URL	.....	
SPRING_DATASOURCE_USERNAME	.....	

[Create Environment Group](#) [+ Add Environment Variable](#) [Save Changes](#)

Slika 5.19: Varijable okruženja za backend

## Baza podataka

Napraviti instancu tipa “PostgreSQL”. Za regiju odabrati opet geografski najbližu, a imena instance, baze i korisnika izabrati proizvoljno. Uzeti besplatnu opciju i kreirati instancu.

U postavkama instance se sada može pronaći URL baze podataka na poslužitelju, kao i lozinka za kreiranog korisnika. Ove podatke valja zapisati u odgovarajuće varijable okruženja u instanci za backend. Osim toga, prikazana je i naredba koja se može iskoristiti za povezivanje na bazu podataka. Sada je dovoljno iskoristiti tu naredbu za spajanje na bazu podataka i dalje je sve isto kao i kad se radi lokalno. Sa “\i imeSqlSkripte” se može pokrenuti skripta za stvaranje tablica i sl.

Pri radu s bazom podataka treba obratiti pozornost na kodne stranice. Zato je važno (vrijedi za Windows okruženje) prije spajanja na bazu podataka unijeti naredbu “*chcp 65001*”, a nakon spajanja na bazu unijeti SQL naredbu “*set client\_encoding = 'UTF8'*”, kako bi se hrvatska slova ispravno prikazivala.

#### Connections

Hostname	dpg-clackou2eqrc7398q2lg-a
Port	5432
Database	ozdravidb_akk2
Username	ozdraviapp
Password	<input type="password"/>
Internal Database URL	<input type="password"/>
External Database URL	<input type="password"/>
PSQL Command	<input type="password"/>

Slika 5.20: Postavke za instancu baze podataka

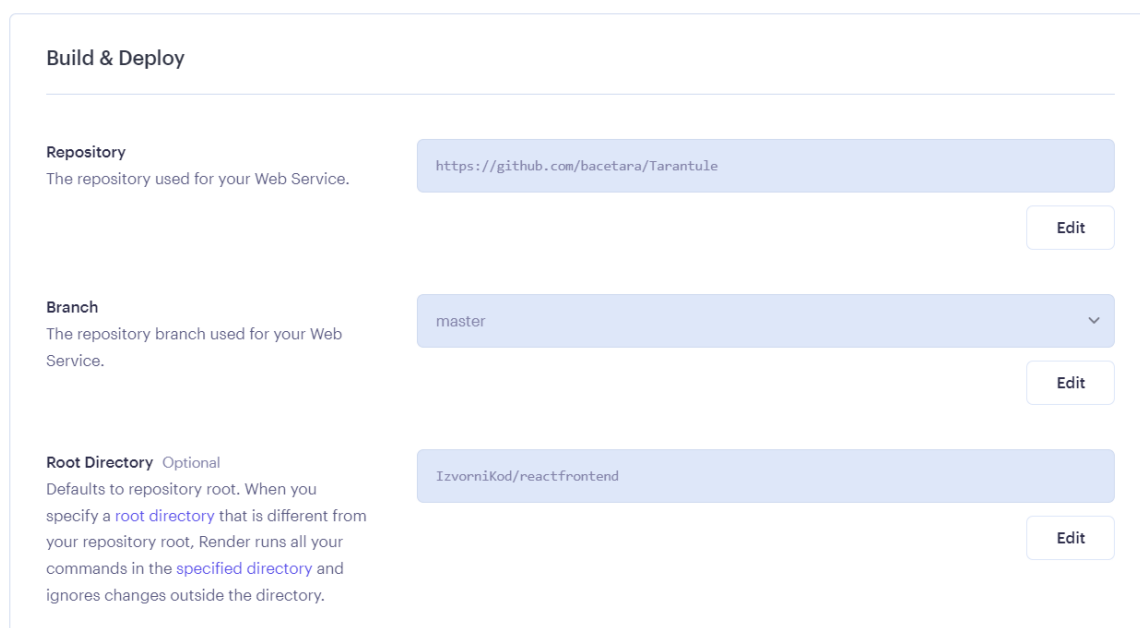
### Frontend

Opet, stvoriti Render instancu, ovog puta tipa “web servis”, kao i za backend. Povezati s git repozitorijem. Postavke izabrati na analogan način kao i za backend. Ime proizvoljno (ovdje *ozdraviapp-fe*), regija geografski najbliža, grana master, korigirani direktorij frontend projekta i besplatna verzija. No, ovdje runtime neće biti Docker već Node.js. U polje *Build command* upisati “*yarn build*”, a u polje *Start command* upisati “*npm start*”.

Kako bi frontend funkcionirao, treba mu dati do znanja gdje se nalazi backend. To treba postaviti u datoteci *setupProxy.js*. Za lokalno testiranje, postavljeno je da svi pozivi prema putanjama */api/...* budu preusmjereni na *localhost:8080* (port na kojemu je backend). Analogno tome, za deploy ovi pozivi trebaju ići na URL na kojemu se nalazi backend (npr. *ozdraviapp-be.onrender.com*), a koji se može pronaći u postavkama instance na kojoj je backend. Taj se URL može ili kodirati u samu datoteku *setupProxy.js* ili postaviti kao varijabla okruženja na render

instanci za frontend (što je bolja opcija) i onda se u kodu dohvatiti pomoću “process.env.IME\_VARIJABLE”. Sada se frontend može pokrenuti na isti način kao i backend, opcijom “manual deploy - deploy latest commit”.

Aplikaciji se sada može pristupiti koristeći URL frontenda (što se tiče ove aplikacije ovdje). Valja napomenuti da se nakon perioda nekorisćenja render instance same gase, pa je zato prvo pokretanje nakon nekoliko sati obično vrlo sporo.



**Build & Deploy**

---

**Repository**  
The repository used for your Web Service.

`https://github.com/bacetara/Tarantule`

Edit

**Branch**  
The repository branch used for your Web Service.

master

Edit

**Root Directory** Optional  
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/reactfrontend

Edit

Slika 5.21: Postavke za frontend instancu, prvi dio

**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/reactfrontend/ \$ yarn build

Edit

**Pre-Deploy Command** Optional  
This command runs before starting your service. It is typically used for tasks like running a database migration or uploading assets to a CDN.

IzvorniKod/reactfrontend/ \$

Edit

**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/reactfrontend/ \$ npm start

Edit

Slika 5.22: Postavke za frontend instancu, drugi dio

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

Key

Value

REACT\_APP\_PROXY\_HOSThttps://ozdraviapp-be.onrender.com/

Create Environment Group+ Add Environment VariableSave Changes

Slika 5.23: Varijable okruženja za frontend

## 6. Zaključak i budući rad

Cilj rada na ovom projektu bio je napraviti aplikaciju koja omogućava roditeljima s često bolesnom djecom i liječnicima te pedijatrima lakše dogovaranje oko pregleda, slanje nalaza, slanje ispričnica te odobravanje bolovanja. Tijekom rada na projektu najveći je izazov bio što nijedan član tima nije bio upoznat s načinom izrade aplikacija koristeći Spring Boot i React. Članovi su puno vremena uložili u samo upoznavanje s tehnologijama te bi ovaj projekt bio puno lakši da su ta znanja već bila stečena.

Rad na projektu trajao je 11 tjedana. U prvih 5 tjedana članovi su stekli znanja o tome kako organizirati funkcionalne i ostale zahtjeve sustava te iz njih napisati obrasce uporabe. Naučili su pomoću njih izraditi sekvencijski dijagram te dijagram obrazaca uporabe. Također su se upoznali s arhitekturom mobilne aplikacije te izradom baze podataka.

U idućih 6 tjedana članovi su uglavnom radili na implementaciji projekta. Upoznali su se sa strukturom koda u Springu te načinu funkcioniranja React stranica. Veliki je dio izazova bio i sinkronizacija backenda i frontenda jer se prilikom njihovog prvotnog spajanja uočilo puno greški. Osim toga, članovi su se upoznali i s ostatkom UML dijagrama: dijagram stanja, aktivnosti, komponenti te razmještaja.

Sudjelovanje na projektu bilo je vrlo korisno iskustvo. Članovi su osim stečenih znanja o navedenih tehnologija imali priliku iskusiti timski rad u kojem su organizacija i međusobno pomaganje ključni za uspjeh. Ovo je također mnogim članovima bio prvi doticaj s ozbiljnijim projektom u kojem postoje strogi vremenski rokovi. Svi su članovi zadovoljni sa stečenim znanjem i iskustvom.

# Popis literature

## *Kontinuirano osvježavanje*

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

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

# Indeks slika i dijagrama

2.1	Snimka zaslona koja prikazuje usluge Portala zdravlja . . . . .	7
3.1	UML dijagram koji opisuje obrasce uporabe korisnika i roditelja . . .	30
3.2	UML dijagram koji opisuje obrasce uporabe liječnika . . . . .	31
3.3	UML dijagram koji opisuje obrasce uporabe pedijatra . . . . .	32
3.4	UML dijagram koji opisuje obrasce uporabe administratora . . . . .	33
3.5	Sekvencijski dijagram koji opisuje osnovnu mehaniku naručivanja djeteta i roditelja na pregled . . . . .	34
3.6	Sekvencijski dijagram za UC30.1 i UC30.2 . . . . .	36
3.7	Sekvencijski dijagram za UC5 . . . . .	38
3.8	Sekvencijski dijagram za UC13 i UC21 . . . . .	39
4.1	Arhitektura sustava . . . . .	41
4.2	Dijagram baze podataka . . . . .	46
4.3	Dijagram Controllera, Servica i Repository-a . . . . .	47
4.4	Dijagram DTO . . . . .	50
4.5	Dijagram modela . . . . .	51
4.6	Dijagram stanja . . . . .	53
4.7	Dijagram aktivnosti . . . . .	55
4.8	Dijagram komponenti . . . . .	56
5.1	Prvi ispitni slučaj . . . . .	59
5.2	Drugi ispitni slučaj . . . . .	59
5.3	Treći ispitni slučaj . . . . .	60
5.4	Četvrti ispitni slučaj . . . . .	60
5.5	Peti ispitni slučaj . . . . .	61
5.6	Šesti ispitni slučaj . . . . .	61
5.7	Sedmi ispitni slučaj . . . . .	62
5.8	Rezultati testova . . . . .	62
5.9	Rezultati prvog ispitnog slučaja . . . . .	63
5.10	Rezultati drugog ispitnog slučaja . . . . .	64

5.11 Rezultati drugog ispitnog slučaja . . . . .	65
5.12 Rezultati trećeg ispitnog slučaja . . . . .	65
5.13 Rezultati četvrtog ispitnog slučaja . . . . .	66
5.14 Rezultati i parametri petog ispitnog slučaja . . . . .	67
5.15 Dijagram razmještaja . . . . .	68
5.16 Dockerfile . . . . .	70
5.17 Postavke za backend instancu, prvi dio . . . . .	71
5.18 Postavke za backend instancu, drugi dio . . . . .	72
5.19 Varijable okruženja za backend . . . . .	72
5.20 Postavke za instancu baze podataka . . . . .	73
5.21 Postavke za frontend instancu, prvi dio . . . . .	74
5.22 Postavke za frontend instancu, drugi dio . . . . .	75
5.23 Varijable okruženja za frontend . . . . .	75
6.1 Dijagram pregleda promjena 1 . . . . .	85
6.2 Dijagram pregleda promjena 2 . . . . .	86



# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 24. listopad 2023.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:
  - sastanak s asistenticom i demosom
  - razrada i analiza zahtjeva projekta
  - definiranje osnovnih funkcionalnosti
  - podjela zaduženja na projektu

### 2. sastanak

- Datum: 27. listopad 2023.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:
  - predstavljanje nacrtu aplikacije
  - skica baze podataka
  - komentiranje oblikovnih obrazaca

### 3. sastanak

- Datum: 7. studenoga 2023.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:
  - sastanak s asistenticom i demosom
  - komentiranje i ispravak baze podataka
  - komentiranje dijagrama

### 4. sastanak

- Datum: 15. studenoga 2023.

- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, J. Pavlić
- Teme sastanka:
  - dovršavanje front end-a prve verzije aplikacije
  - komentiranje statusa projekta
  - deploy aplikacije

#### 5. sastanak

- Datum: 5. prosinca 2023.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:
  - sastanak s asistenticom i demosom
  - Analiza prve predaje

#### 6. sastanak

- Datum: 12. prosinca 2023.
- Prisustvovali: T. Baće, M. Crnolatac, V. Sivec
- Teme sastanka:
  - dodjela poslova na frontendu
  - dogovaranje poslova na backendu

#### 7. sastanak

- Datum: 13. prosinca 2023.
- Prisustvovali: T. Baće, M. Crnolatac, J. Pavlić, V. Sivec
- Teme sastanka:
  - nastavak dodjele poslova na frontendu
  - definiranje zahtjeva za backend

#### 8. sastanak

- Datum: 15. siječnja 2024.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:
  - spajanje frontenda i backenda

#### 9. sastanak

- Datum: 16. siječnja 2024.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac, B. Gabelica, A. Jerbić, J. Pavlić, V. Sivec
- Teme sastanka:

- pregled napravljenog s asistenticom i demosom

#### 10. sastanak

- Datum: 19. siječnja 2024.
- Prisustvovali: T. Baće, E. Bošnjak, M. Crnolatac
- Teme sastanka:
  - deploy aplikacije

## Tablica aktivnosti

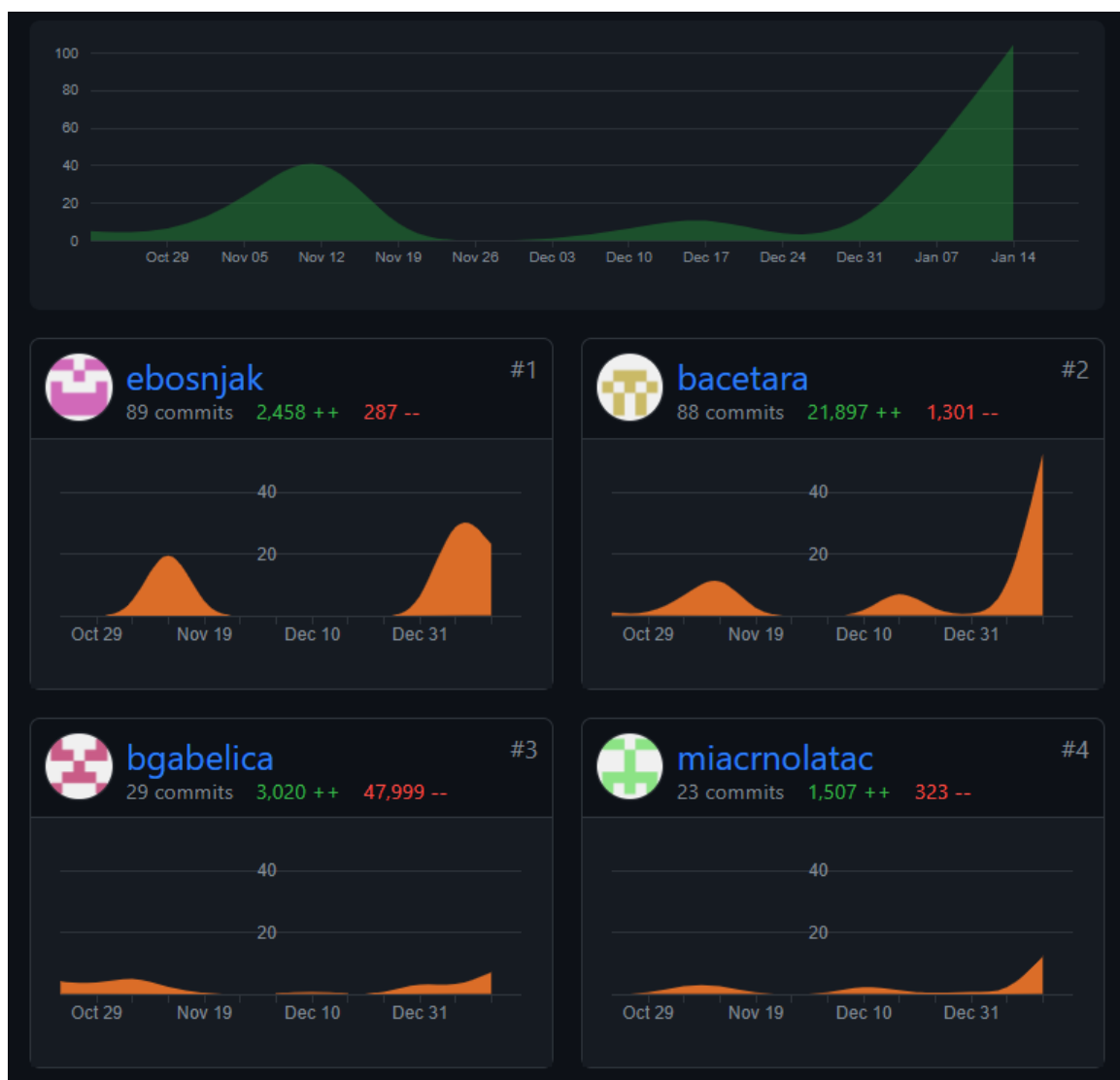
	Tara Baće	Eugen Bošnjak	Mia Crnolatac	Benedicte Gabelica	Alan Jerbić	Josip Pavlić	Vilim Sivec
Upravljanje projektom	12						
Opis projektnog zadatka				1			
Funkcionalni zahtjevi				1			
Opis pojedinih obrazaca				13			
Dijagram obrazaca						4	
Sekvencijski dijagrami	5					5	
Opis ostalih zahtjeva	1						
Arhitektura i dizajn sustava				3			
Baza podataka					7		
Dijagram razreda	1			3	3		
Dijagram stanja				2			
Dijagram aktivnosti				2			
Dijagram komponenti				1			
Korištene tehnologije i alati				1			
Ispitivanje programskog rješenja	10	5	5				5
Dijagram razmještaja				1			
Upute za puštanje u pogon	2	2	2				
Dnevnik sastajanja	2						
Zaključak i budući rad				1			

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Tara Baće	Eugen Bošnjak	Mia Crnolatac	Benedicte Gabelica	Alan Jerbić	Josip Pavlić	Vilim Sivec
Popis literature				1			
<i>skica aplikacije</i>	3		4	2			
<i>front end</i>	30		30			12	20
<i>izrada baze podataka</i>					2		
<i>spajanje s bazom podataka</i>		4					
<i>back end</i>	1	30	1	11	20		
<i>deploy</i>	5	5	5	2		2	

## Dijagrami pregleda promjena



Slika 6.1: Dijagram pregleda promjena 1



Slika 6.2: Dijagram pregleda promjena 2