

Programsko inženjerstvo

Ak. god. 2023./2024.

# Medicinska rehabilitacija

Dokumentacija, Rev. 1.0

Grupa: MedBay

Voditelj: Karlo Vrančić

Datum predaje: 17. studenog 2023.

Nastavnik: Miljenko Krhen

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
<b>3 Specifikacija programske potpore</b>	<b>14</b>
3.1 Funkcionalni zahtjevi . . . . .	14
3.1.1 Obrasci uporabe . . . . .	16
3.1.2 Sekvencijski dijagrami . . . . .	27
3.2 Ostali zahtjevi . . . . .	32
<b>4 Arhitektura i dizajn sustava</b>	<b>36</b>
4.1 Baza podataka . . . . .	42
4.1.1 Opis tablica . . . . .	43
4.1.2 Dijagram baze podataka . . . . .	47
4.2 Dijagram razreda . . . . .	48
4.3 Dijagram stanja . . . . .	51
4.4 Dijagram aktivnosti . . . . .	53
4.5 Dijagram komponenti . . . . .	54
<b>5 Implementacija i korisničko sučelje</b>	<b>55</b>
5.1 Korištene tehnologije i alati . . . . .	55
5.2 Ispitivanje programskog rješenja . . . . .	58
5.2.1 Ispitivanje sustava . . . . .	61
5.3 Dijagram razmještaja . . . . .	64
5.4 Upute za puštanje u pogon . . . . .	65
<b>6 Zaključak i budući rad</b>	<b>68</b>
<b>Popis literature</b>	<b>70</b>
<b>Indeks slika i dijagrama</b>	<b>71</b>

**Dodatak: Prikaz aktivnosti grupe**

72

# 1. Dnevnik promjena dokumentacije

*Kontinuirano osvježavanje*

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljena inicijalna skica toka i obrazaca upotrebe.	svi	23.10.2023.
0.2	Dodani funkcionalni zahtjevi. Dodani obrasci uporabe.	Tea, Nikola, Ivan	05.11.2023.
0.2.1	Revizija funkcionalnih zahtjeva i obrazaca uporabe.	Karlo	07.11.2023.
0.3	Dodan <i>Use Case</i> dijagram i četiri sekvencijska dijagrama. Dodatna revizija obrazaca uporabe.	Niko, Ivan	09.11.2023.
0.4	Dodan opis baze podataka.	Tea	12.11.2023.
0.5	Dodan opis arhitekture.	Tea, Karlo	14.11.2023.
0.6	Napisan <i>Opis projektnog zadatka, Dodatak, Ne-funkcionalni zahtjevi i Dnevnik promjena</i>	Karlo	14.11.2023.
0.9	Opisi obrazaca uporabe	*	07.09.2013.
0.10	Preveden uvod	*	08.09.2013.
0.11	Sekvencijski dijagrami	*	09.09.2013.
0.12.1	Započeo dijagrame razreda	*	10.09.2013.
0.12.2	Nastavak dijagrama razreda	*	11.09.2013.
<b>1.0</b>	Verzija samo s bitnim dijelovima za 1. ciklus	svi	17.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodatka</b>	<b>Autori</b>	<b>Datum</b>
1.1	Uređivanje teksta – funkcionalni i nefunkcionalni zahtjevi	*	14.09.2013.
1.2	Manje izmjene:Timer - Brojilo vremena	*	15.09.2013.
1.3	Popravljeni dijagrami obrazaca uporabe	*	15.09.2013.
1.4	Izmjenjen dijagram aktivnosti i dodano ispitivanje programskog rješenja	Nikola	19.1.2024.
1.5	Generalna revizija strukture dokumenta	*	19.09.2013.
1.5.1	Manja revizija (dijagram razmještaja)	*	20.09.2013.
<b>2.0</b>	Konačni tekst predloška dokumentacije	*	28.09.2013.

*Moraju postojati glavne revizije dokumenata 1.0 i 2.0 na kraju prvog i drugog ciklusa. Između tih revizija mogu postojati manje revizije već prema tome kako se dokument bude nadopunjavao. Očekuje se da nakon svake značajnije promjene (dodatak, izmjene, uklanjanja dijelova teksta i popratnih grafičkih sadržaja) dokumenta se to zabilježi kao revizija. Npr., revizije unutar prvog ciklusa će imati oznake 0.1, 0.2, ..., 0.9, 0.10, 0.11.. sve do konačne revizije prvog ciklusa 1.0. U drugom ciklusu se nastavlja s revizijama 1.1, 1.2, itd.*

## 2. Opis projektnog zadatka

*dio 1. revizije*

### Uvod

Jedan od ključnih izazova s kojima se suočavamo u *suvremenom zdravstvu* jest **neefikasnost sustava dodjele termina** i upravljanja medicinskim procesima. Područje u kojem je to zbog svoje bliskosti s "običnim" čovjekom posebno evidentno jesu **sustavi rehabilitacije**. Njih karakterizira kompleksnost potreba pacijenata, raznolikost terapijskih pristupa i ograničeni resursi, posebno izraženi u kontekstu hrvatskog zdravstva. Dosadašnji **manualni procesi** evidencije i upravljanja terminima često dovode do neoptimalnog iskorištavanja već oskudnih resursa.

### Potreba za inovacijom

Postoji očita potreba za razvojem *novog sustava* koji će omogućiti bolje upravljanje rehabilitacijskim procesima, povećati efikasnost i znatno unaprijediti iskustvo i zadovoljstvo pacijenata.

### Transformacija procesa

Projekt je započeo s vizijom transformacije postojećih manualnih procesa u **automatizirani, digitalizirani sustav**. Cilj nam je bio kreiranje platforme koja optimizira raspodjelu termina, **povećava efikasnost** i pruža transparentnost u praćenju i upravljanju procesima rehabilitacije.

### Inkluzivnost i dostupnost

Fokus projekta bio je na razvoju *inkluzivnog* servisa dostupnog za široku populaciju. Zbog toga je potrebno omogućiti jednostavno korištenje platforme za sve članove društva, uključujući pacijente i djelatnike.

## Razvoj web aplikacije

Ključni dio projekta uključuje razvoj web aplikacije koja bolesnicima omogućava prijavu na rehabilitaciju, odabir terapije i termine te praćenje njihovog napretka. Djelatnicima zdravstvene ustanove pruža se niz alata za efikasno upravljanje terminima i bilježenje napretka pacijenata, dok se administratorima omogućava upravljanje korisničkim računima i resursima.

## Funkcionalnost aplikacije

Aplikacija je zamišljena tako da omogućuje bolesnicima prijavu na rehabilitaciju i praćenje napretka te sljedećih termina u stvarnom vremenu. S druge strane, djelatnici imaju interaktivan raspored kojem mogu pristupiti u bilo kojem trenutku, dok administratori nadziru sve termine i rasporede kako bi bili sigurni da je čitav proces optimiziran.

U našoj aplikaciji, interakcija između glavnih dionika - bolesnika, djelatnika zdravstvene ustanove i administratora sustava - ključna je za njezin uspješan rad. Svaki od ovih dionika ima specifične uloge i funkcionalne zahtjeve.

### Administratori sustava

- Upravljanje korisnicima:** Administratori imaju ključnu ulogu u upravljanju korisničkim računima, kako za djelatnike, tako i za bolesnike. Oni mogu prihvati ili odbiti registracije te uređivati ili brisati postojeće račune.
- Nadzor termina:** Odgovorni su za pregled i potvrdu prijava bolesnika za rehabilitaciju te mogu otkazati ili pomaknuti zakazane termine.
- Pristup i analiza podataka:** Administratori imaju pristup cijelokupnoj bazi podataka, omogućavajući im nadzor i analizu svih aspekata rehabilitacijskog procesa.

### Djelatnici zdravstvene ustanove

- Bilježenje napretka:** Djelatnici mogu bilježiti napredak bolesnika tijekom rehabilitacijskih sesija, što je ključno za praćenje i prilagodbu terapijskih pristupa.

- Upravljanje rasporedom:** Imaju mogućnost pregleda i upravljanja vlastitim rasporedom sesija, kao i pristup podacima o bolesnicima.
- Komunikacija s bolesnicima:** Mogućnost otkazivanja sesija pod određenim uvjetima omogućava djelatnicima fleksibilnost u upravljanju izvanrednim situacijama.

## bolesnici

- Proces registracije i prijave:** Bolesnici se mogu registrirati u sustav, prijaviti se i pristupiti svojim profilima za praćenje rehabilitacije.
- Praktične mogućnosti:** Mogu se prijaviti na rehabilitaciju, odabrati terapiju, termine dolaska (koji prate pravila specifične terapije) te pratiti svoje termine i povijest terapija. U iznimnim situacijama mogu pomaknuti termin.
- Pristup informacijama:** Imaju pristup svom kalendaru s terminima terapije što im omogućava bolje planiranje i organizaciju.

## Baza podataka

- Središnje skladište podataka:** Sve informacije o bolesnicima, djelatnicima i rehabilitacijskim sesijama pohranjuju se u bazi podataka čime se osigurava integracija i efikasnost sustava.

## Poslovni model

### B2B model i implementacija u bolničkom sustavu

- **Orijentacija na bolnički sustav:** Naša aplikacija je primarno namijenjena upotrebi unutar *bolničkog sustava*, što je usko povezano s B2B (business-to-business) modelom. Ovaj pristup podrazumijeva da naša aplikacija nije direktno namijenjena pojedinačnim korisnicima, već bolnicama i poliklinikama kao korporativnim klijentima.
- **Razina implementacije:** Uspješna implementacija aplikacije zahtijeva široku adaptaciju, idealno na nacionalnoj razini ili barem unutar privatnih poliklinika za rehabilitaciju. Takav pristup omogućuje efikasniju koordinaciju i standardizaciju procesa rehabilitacije.
- **Značaj skalabilnosti:** Naš fokus je bio na razvoju aplikacije koja je fleksibilna i skalabilna, sposobna prilagoditi se različitim veličinama i vrstama zdravstvenih ustanova.

### Ekspanzija i skalabilnost

- **Faza početne implementacije:** Trenutno je aplikacija razvijena za upotrebu unutar jedne bolnice ili poliklinike za rehabilitaciju. Ovaj početni pristup omogućuje nam detaljno testiranje i optimizaciju funkcionalnosti aplikacije.
- **Strategije ekspanzije:** U razmatranju načina ekspanzije identificirali smo dva glavna pristupa:
  1. *Individualizirani sustavi:* Razvijanje *custom* verzije aplikacije za svaku bolnicu. Svaka ustanova imala bi svoju prilagođenu verziju aplikacije što bi moglo poslužiti kao dodatni faktor privlačnosti za klijente.
  2. *Centralizirani sustav:* Stvaranje jedne velike centralizirane aplikacije. Ovaj pristup uključuje izbor institucije na početku korištenja aplikacije.
- **Potencijal za daljnju ekspanziju:** Oba pristupa imaju svoje prednosti i izazove te pružaju različite mogućnosti za daljnju ekspanziju i adaptaciju aplikacije, u skladu s potrebama tržišta i specifičnostima korisnika. Lako je moguće da bismo se odlučili za *hibridni pristup* gdje bi svaka bolnica imala narezgled "svoju" aplikaciju "hostiranu" na *custom* domeni, s vizualno prilagođenim

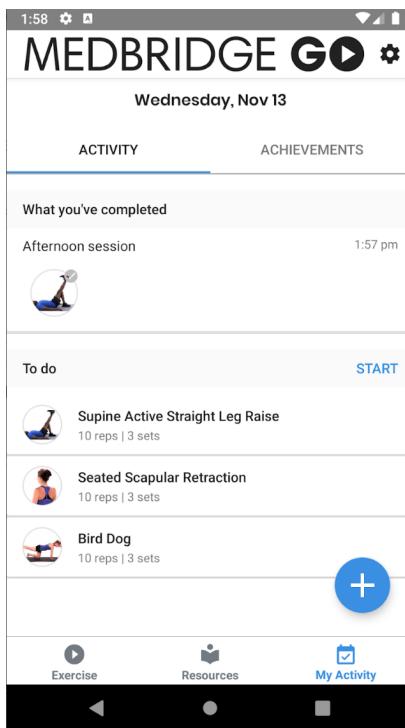
korisničkim sučeljem (UI). Ovaj pristup kombinira prednosti individualiziranih i centraliziranih sustava omogućujući bolnicama da se osjećaju kao da imaju jedinstveno, prilagođeno rješenje dok se u stvarnosti koristi jedan centralizirani sustav.

Pod "haubom", sve bi aplikacije bile zasnovane na istoj osnovnoj arhitekturi i dijelile isti skup funkcionalnosti. To bi omogućilo efikasnije održavanje i ažuriranje sustava dok bi korisnici imali dojam personaliziranog iskustva. Tehnički, ovakav pristup može se postići korištenjem *proxy servera* ili sličnih tehnologija za usmjeravanje prometa na odgovarajuće instance aplikacije omogućujući tako svakoj bolnici da ima svoju unikatnu URL adresu i prilagođen UI. Ovaj hibridni model pruža fleksibilnost u pružanju usluga klijentima, dok istovremeno minimizira troškove i složenost održavanja više odvojenih instanci aplikacije.

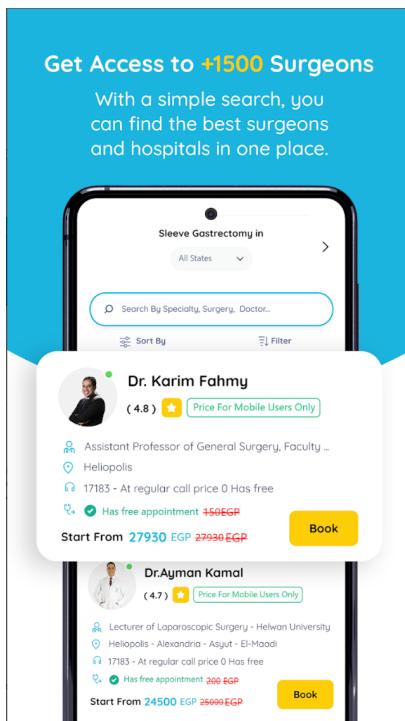
## Analiza konkurenčije:

Konkurenčiju najbolje možemo podijeliti u dvije grupe:

- **Manualni sustavi:** Naša prva i najizazovnija konkurenčija su postojeći manualni sustavi. Promjena navika i entuzijazam djelatnika za učenje nečega novog predstavljaju značajne prepreke.
- **Digitalna rješenja:** Iako je teško pronaći konkretna B2B rješenja putem običnog pretraživanja, neka od rješenja koja smo identificirali uključuju:
  - **MedBridge Go:** Radi se o aplikaciji koja je pretežno usmjerena odradivanju vježbi koje je doktor propisao kod kuće. Također postoje posebni profili za liječnika i pacijenta, ali ovdje se ne delegira resursima odjela za fizikalnu terapiju te zbog toga nije direktna konkurenčija.
  - **BokDoc i BokDoc Partner:** Aplikacijski paket namijenjen rezervaciji termina kod doktora te promociji različitih privatnih klinika. Izgleda kao hibrid društvene mreže i oglasnika s mogućnošću narudžbe. Isto nije slična našem proizvodu. Ponovno postoje različiti profili za liječnika i pacijenta, ovoga puta to su potpuno različite aplikacije.

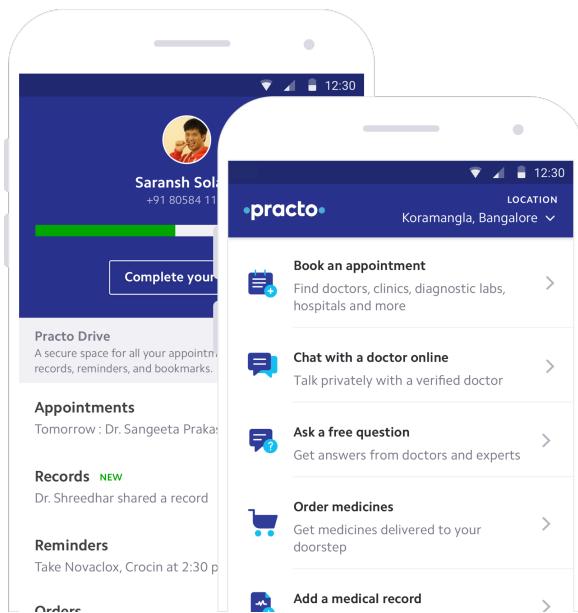


Slika 2.1: Sučelje aplikacije Medbridge GO koja je dostupna na pametnim telefonima



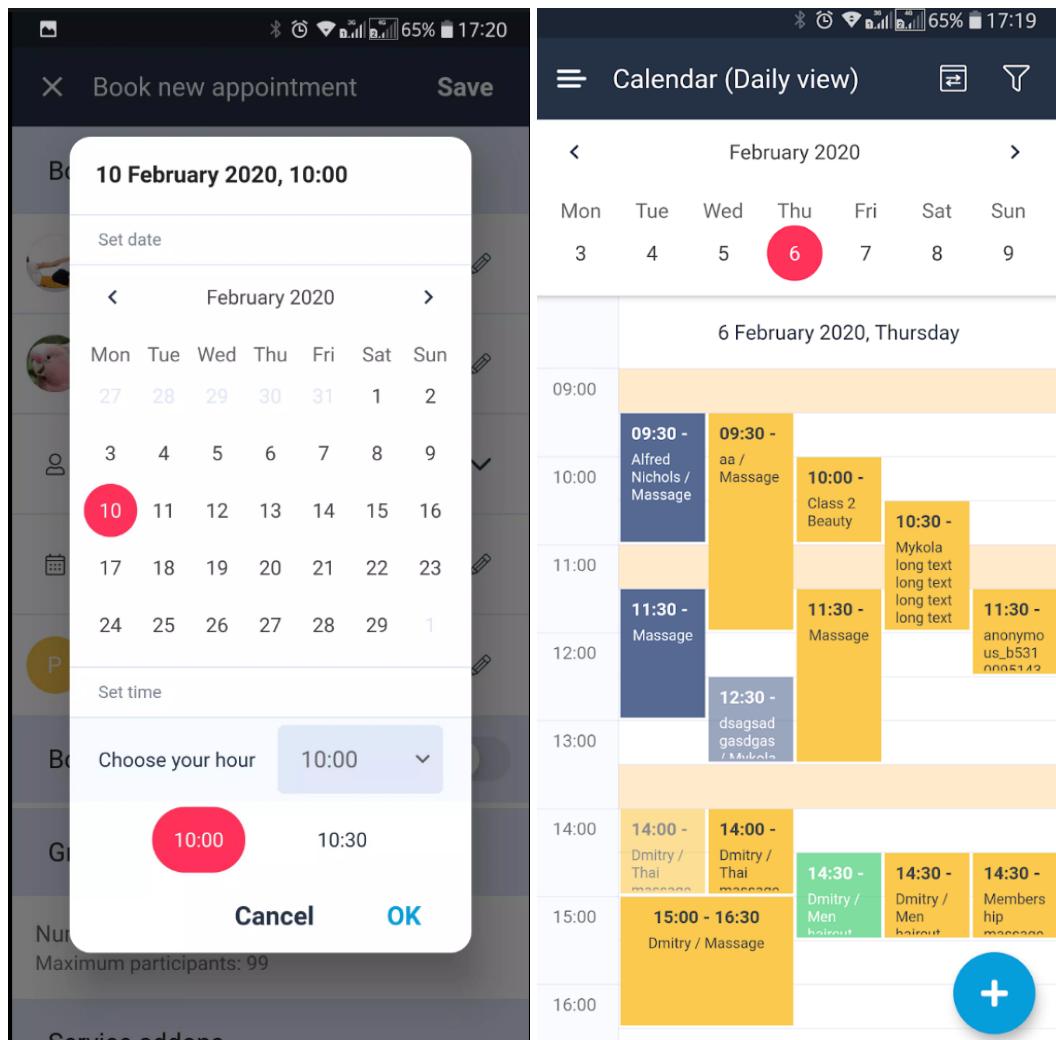
Slika 2.2: Sučelje aplikacije BokDoc koja je dostupna na pametnim telefonima

– **Practo:** Indijska telemedicinska aplikacija koja nudi različite zdravstvene usluge. Uključuje video konzultacije, online zakazivanje, pronađak bolnica i tretmana te čitanje zdravstvenih savjeta. Velika mreža liječnika i pružatelja zdravstvenih usluga dostupna je korisnicima. Razlikuje se od našeg proizvoda jer se fokusira na širok spektar zdravstvenih usluga, dok je naša aplikacija specifična za upravljanje procesima rehabilitacije.



Slika 2.3: Sučelje aplikacije Practo, dostupno na pametnim telefonima

– **SimplyBook.me i SimplyBook.me Admin:** SimplyBook.me je aplikacija za upravljanje terminima, posebno prilagođena korisnicima koji žele lagan pristup svojim postojećim rezervacijama. Omogućuje jednostavno dodavanje klijenata i uređivanje starih rezervacija. Korisnici primaju obavijesti o novim rezervacijama i podsjetnike za nadolazeće termine. Aplikacija je idealna za korisnike koji su često u pokretu, nemaju pristup uređajima s velikim ekranima tijekom radnog vremena ili žele provjeriti nadolazeće termine dok su kod kuće. Razlikuje se od našeg proizvoda jer je usmjerena na općenito upravljanje terminima, dok se naša aplikacija fokusira na specifične potrebe procesa rehabilitacije u zdravstvenim ustanovama.



Slika 2.4: Sučelje aplikacije SimplyBook.me: lijevo - korisnički pogled, desno - admin pogled

## Zaključak

Projekt koji smo razvili predstavlja potencijalni korak prema poboljšanju sustava rehabilitacije unutar zdravstvenog sektora. Naša aplikacija nudi inovativno rješenje koje se bavi ključnim izazovima učinkovitog upravljanja terminima i procesima, a posebno je prilagođena specifičnostima hrvatskog zdravstvenog sustava.

- Inovativni pristup:** Transformacijom manualnih procesa u digitaliziranu, automatiziranu platformu aplikacija donosi povećanu efikasnost, transparentnost i bolje iskorištavanje resursa. To ne samo da unapređuje iskustvo pacijenata, već i olakšava posao djelatnicima i administratorima.
- Fokus na korisnicima:** Razvojem aplikacije s jasnim fokusom na potrebe i

udobnost korisnika posebno smo se posvetili stvaranju intuitivnog sučelja koje je pristupačno i jednostavno za korištenje svim dionicima procesa rehabilitacije.

- **Prilagodljivost i skalabilnost:** Projekt je dizajniran s obzirom na buduću ekspanziju i prilagodbu. Hibridni pristup razvoja aplikacije omogućava nam da odgovorimo na različite potrebe i zahtjeve različitih zdravstvenih ustanova nudeći im prilagođeno rješenje dok istovremeno zadržavamo unificiranu, centraliziranu strukturu.
- **Doprinos zdravstvu:** Ova aplikacija ima potencijal da značajno doprinese hrvatskom zdravstvenom sustavu koji bi osvremenila glede procesa rehabilitacije što će imati dugoročne pozitivne učinke na kvalitetu zdravstvene skrbi.

Zaključno, ovaj projekt predstavlja važan korak naprijed u digitalizaciji zdravstvenih procesa s potencijalom da unese značajne promjene u načinu na koji se upravlja rehabilitacijom i srodnim zdravstvenim uslugama. Kroz kontinuirani razvoj i prilagodbu, ovaj sustav može služiti kao model za buduće inovacije unutar zdravstvenog sektora.

## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

#### *dio 1. revizije*

*Glavni dionici su bolesnici, djelatnici zdravstvene ustanove te administratori sustava. Bolesnici su krajnji korisnici koji se prijavljuju na rehabilitaciju. Djelatnici zdravstvene ustanove provode rehabilitaciju i upravljaju terminima. Administratori sustava nadziru cjelokupno funkcioniranje sustava i upravljaju korisnicima.*

#### **Dionici:**

1. Administrator
2. Djelatnik
3. Bolesnik

#### **Aktori i njihovi funkcionalni zahtjevi:**

##### **1. Administrator može:**

- (a) Prijaviti se u sustav
- (b) Prihvati ili odbiti registraciju bolesnika
- (c) Pregledati i potvrditi prijave bolesnika za rehabilitaciju
- (d) Pregledati sve termine sesija
- (e) Dodavati račune djelatnika
- (f) Brisati račune bolesnika i djelatnika
- (g) Pregledavati, uređivati i brisati podatke računa bolesnika i djelatnika
- (h) Otkazati zakazani termin
- (i) Pomicati zakazane termine

##### **2. Djelatnik može:**

- (a) Prijaviti se u sustav
- (b) Bilježiti napredak svojih bolesnika
- (c) Pregledati raspored svojih sesija
- (d) Pregledati podatke o bolesnicima

(e) Otkazati sesiju pod određenim uvjetima

3. Bolesnik može:

- (a) Registrirati se u sustav
- (b) Prijaviti se u sustav
- (c) Prijaviti se na rehabilitaciju
- (d) Odabrati terapiju i termine dolaska na rehabilitaciju
- (e) Pomaknuti zakazani termin pod određenim uvjetima
- (f) Pregledati vlastiti kalendar s terminima terapije
- (g) Pregledati povijest svojih terapija

4. Baza podataka:

- (a) Pohranjuje sve podatke o bolesnicima i njihovim rehabilitacijama
- (b) Pohranjuje sve potrebne podatke o djelatnicima i njihovim terminima
- (c) Pohranjuje podatke o resursima ustanove

### 3.1.1 Obrasci uporabe

#### dio 1. revizije

##### Opis obrazaca uporabe

Funkcionalne zahtjeve razraditi u obliku obrazaca uporabe. Svaki obrazac je potrebno razraditi prema donjem predlošku. Ukoliko u nekom koraku može doći do odstupanja, potrebno je to odstupanje opisati i po mogućnosti ponuditi rješenje kojim bi se tijek obrasca vratio na osnovni tijek.

#### UC1 - Zahtjev za registraciju bolesnika u sustav

- **Glavni sudionik:** Bolesnik
- **Cilj:** Poslati ispravan zahtjev za registraciju u sustav kako bi pristupio procesu rehabilitacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik nema prethodno registriran korisnički račun
- **Opis osnovnog tijeka:**
  1. Bolesnik unosi osobne podatke, adresu elektroničke pošte i lozinku.

#### UC2 - Potvrda registracije bolesnika u sustav

- **Glavni sudionik:** Administrator
- **Cilj:** Provjeriti i potvrditi registraciju bolesnika.
- **Sudionici:** Baza podataka, Bolesnik
- **Preduvjet:** Bolesnik je prethodno poslao zahtjev za registraciju, administrator je prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Administrator provjerava ispravnost podataka.
  2. Sustav pohranjuje podatke i stvara korisnički račun.
  3. Bolesnik prima elektroničku poštu s potvrdom registracije.
- **Opis mogućih odstupanja:** U slučaju neispravne registracije u smislu duplike OIB-a ili MBO-a frontend ne dopušta slanje registracije. Ukoliko nepravilnosti uoči administrator, Bolesnik prima elektroničku poštu na upisanu adresu u kojoj mu se govori da registracija nije uspjela.

### UC3 - Prijava bolesnika/djelatnika/administratora u sustav

- **Glavni sudionik:** Bolesnik/Djelatnik/Administrator
- **Cilj:** Korisnik se želi prijaviti u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnikov račun je u bazi podataka
- **Opis osnovnog tijeka:**
  1. Korisnik unosi adresu elektroničke pošte i lozinku.
  2. Provjerava se postojanost dane kombinacije u bazi
  3. Ovisno o tipu profila povezanog s unesenim podacima, korisniku se otvara jedno od 3 moguća sučelja (bolesnik, djelatnik, administrator)
- **Opis mogućih odstupanja:** Sustav reagira na neispravne podatke te obavještava korisnika o netočnosti podataka.

### UC4 - Zaboravljena lozinka za prijavu

- **Glavni sudionik:** Bolesnik/Djelatnik/Administrator
- **Cilj:** Prijaviti se unatoč zaboravljenoj lozinci
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnikova e-mail adresa je u bazi podataka
- **Opis osnovnog tijeka:**
  1. Korisnik je zaboravio lozinku te se ne može prijaviti u sustav
  2. Korisnik pritiše gumb za dobivanje nove lozinke
  3. Korisnik na e-mail adresu dobiva upute za postavljanje nove lozinke

### UC5 - Prijava bolesnika na rehabilitaciju

- **Glavni sudionik:** Bolesnik
- **Cilj:** Prijaviti se za proces rehabilitacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik je registriran i prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Bolesnik odabire opciju za prijavu na rehabilitaciju.
  2. Bolesnik unosi informacije o svom oboljenju, potrebnom postupku liječenja i liječniku koji ga je uputio na rehabilitaciju.

3. S obzirom na unesenu vrstu rehabilitacije, sustav provjerava koji su sve termini slobodni (slobodan termin: slobodan resurs i slobodan specijalizirani djelatnik)
  4. Bolesnik odabire datum i vrijeme svojih termina na temelju dostupnih termina
  5. Bolesnik čeka odobrenje administratora (UC18)
  6. Korisnik prima skočni prozor s obavijesti o administratorovoju odluci te sukladno istoj ili može ponoviti prijavu na rehabilitaciju ili je preusmjeren na kontrolnu ploču (dashboard) za novonastalu rehabilitaciju
- **Opis mogućih odstupanja:** Dok sve nužne informacije nisu upisane blokirano je slanje forme na provjeru kod administratora. Korisniku nije dopušteno odabrati sesije koje su vremenski udaljene manje od minimalne razlike između dvije sesije koja je definirana terapijom. Korisniku nije dopušteno slanje forme ukoliko je vremenska razlika između zadnje i prve sesije duža od maksimalnog trajanja terapije koje je definirano terapijom.

#### **UC6 - Promjena zakazanog termina (bolesnik)**

- **Glavni sudionik:** Bolesnik
- **Cilj:** Promijeniti zakazani termin za rehabilitaciju.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik je registriran, prijavljen i ima zakazane termine.
- **Opis osnovnog tijeka:**
  1. Bolesnik ulazi na kontrolnu ploču.
  2. Bolesnik pregledava svoj raspored i odabire termin koji bi htio promijeniti.
  3. Bolesnik u informacijama o terminu odabire opciju za promjenu termina.
  4. Bolesnik pregledava raspored slobodnih termina i odabire zamjenski termin.
  5. Sustav potvrđuje regularnost promjene i ona se evidentira na kontrolnoj ploči djelatnika i bolesnika.
- **Opis mogućih odstupanja:** Ako promjena nije napravljena minimalno 48 sati prije termina, otkazivanja su onemogućena u sustavu i bolesnik se upućuje da u slučaju hitnosti kontaktira administratora direktno.

### UC7 - Pregled relevantnih informacija za budući termin

- **Glavni sudionik:** Bolesnik
- **Cilj:** Dobiti više informacija o određenom terminu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, prijavljen na terapiju i ima barem jedan preostali termin
- **Opis osnovnog tijeka:**
  1. Korisnik na svojem rasporedu odabire jedan termin
  2. Korisnik pritišće gumb "Prikaži više" za taj termin
  3. Korisniku se pojavljuje novi prozor s informacijama o terminu kao što su broj sesije, opis sesije, ime djelatnika te mapa ustanove s naznačenim mjestom odvijanja te sesije

### UC8 - Otkazivanje zakazanog termina (djelatnik)

- **Glavni sudionik:** Djelatnik
- **Cilj:** Otkazati zakazani termin za rehabilitaciju.
- **Sudionici:** Baza podataka
- **Preduvjet:** Djelatnik je prijavljen u sustav. Bolesnik ima zakazane termine.
- **Opis osnovnog tijeka:**
  1. Djelatnik ulazi na kontrolnu ploču i odabire termin koji želi promijeniti.
  2. Djelatnik odabire opciju za otkazivanje termina
  3. Izvršava se promjena u bazi te se obavještava bolesnik koji odabire novi termin
- **Opis mogućih odstupanja:** Djelatnik ne može otkazivati termine koji nisu njegovi.

### UC9 - Otkazivanje zakazanog termina (administrator)

- **Glavni sudionik:** Administrator
- **Cilj:** Otkazati zakazani termin za rehabilitaciju.
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav. Bolesnik ima zakazane termine.
- **Opis osnovnog tijeka:**

1. Administrator odlazi na kalendar i pretražuje termin po djelatniku i/ili po bolesniku.
2. Administrator odabire opciju za otkazivanje termina
3. Izvršava se promjena u bazi te se obavještava djelatnika i bolesnika koji odabire novi termin

#### **UC10 - Pregled kalendarja vlastitih terapija**

- **Glavni sudionik:** Bolesnik/Djelatnik
- **Cilj:** Pregledati kalendar s terminima terapije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik/Djelatnik je registriran i prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Bolesnik/Djelatnik odabire opciju za pregled vlastitog kalendarja.
  2. Sustav prikazuje raspored svih zakazanih termina terapije.
  3. Bolesnik može pregledati sesije i vidjeti napomene, djelatnik ih može upisivati

#### **UC11 - Pregled podataka o bolesnicima**

- **Glavni sudionik:** Djelatnik
- **Cilj:** Pregledati podatke o bolesnicima.
- **Sudionici:** Baza podataka
- **Preduvjet:** Djelatnik je prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Djelatnik pretražuje ime bolesnika ili pregledava listu svih bolesnika.
  2. Sustav prikazuje popis bolesnika i osnovne informacije o njima.
  3. Djelatnik može pregledati detaljnije podatke o svakom bolesniku.

#### **UC12 - Pregled rasporeda svih bolesnika, djelatnika i sesija**

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati raspored bolesnika i sesija rehabilitacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka:**

1. Administrator odabire opciju za pregled rasporeda.
2. Sustav prikazuje raspored svih bolesnika, djelatnika i sesija.
3. Administrator može filtrirati raspored po različitim kriterijima.

#### UC13 - Bilježenje napretka bolesnika

- **Glavni sudionik:** Djelatnik
- **Cilj:** Bilježiti i evidentirati napredak bolesnika tijekom procesa rehabilitacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik je registriran, prijavljen i odradio sesiju rehabilitacije s određenim djelatnikom.
- **Opis osnovnog tijeka:**
  1. Djelatnik odabire bolesnika za bilježenje napretka.
  2. Djelatnik unosi podatke o sesiji rehabilitacije.
  3. Djelatnik potvrđuje unos i spremi podatke.

#### UC14 - Pregled podataka o terapijama

- **Glavni sudionik:** Bolesnik
- **Cilj:** Pregledati podatke o terapijama.
- **Sudionici:** Baza podataka
- **Preduvjet:** Bolesnik je registriran i prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Bolesnik odabire opciju za pregled podataka.
  2. Sustav prikazuje relevantne informacije o terapijama.

#### UC15 - Pregled statistike o djelatnicima i resursima

- **Glavni sudionik:** Administrator
- **Cilj:** Praćenje dostupnosti osoblja, opreme i kapaciteta za rehabilitaciju.
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Administrator iz kontrolne ploče odabire Pregled statistike o djelatnicima i resursima
  2. Sustav prikazuje zadani pogled (engl. *view*).

### **UC16 - Dodavanje djelatnika u bazu**

- **Glavni sudionik:** Administrator
- **Cilj:** Stvoriti korisnički račun za novog djelatnika u sustavu medicinske rehabilitacije.
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Administrator iz kontrolne ploče odabire opciju za stvaranje novog djelatnika.
  2. Sustav prikazuje obrazac za unos osobnih podataka novog djelatnika.
  3. Administrator unosi sve potrebne podatke o novom djelatniku.
  4. Ako su svi podaci ispravni, sustav stvara novi korisnički račun za djelatnika.
- **Opis mogućih odstupanja:** Sustav reagira na neispravne podatke, već postojeće zaposlenike te obavještava administratora o neuspjelom upisu.

### **UC17 - Promjena podataka o djelatnicima i bolesnicima**

- **Glavni sudionik:** Administrator
- **Cilj:** Izmijeniti informacije o djelatnicima i bolesnicima u bazi.
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka:**
  1. Administrator iz kontrolne ploče odabire pregled svih djelatnika ili bolesnika.
  2. Administrator odabire djelatnika/bolesnika i uređuje informacije o istom.
  3. Prije potvrde o promjeni administrator ponovno unosi svoju šifru

### **UC18 - Potvrda o ispravnoj prijavi na rehabilitaciju**

- **Glavni sudionik:** Administrator
- **Cilj:** Potvrditi bolesnikovu prijavu na određenu rehabilitaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav i bolesnik je predao zahtjev za prijavu na rehabilitaciju

- **Opis osnovnog tijeka:**

1. Administrator je obaviješten kako je dobio novi zahtjev za prijavu
2. Administrator analizira podatke o bolesniku i o prijavi te zajedno s informacijama iz baze podataka donosi odluku o ispravnosti zahtjeva
3. S obzirom na rezultat svoje odluke, administrator ili omogućuje prijavu na rehabilitaciju ili dojavljuje bolesniku poruku o pogrešci

#### UC19 - Deaktivacija korisničkog računa djelatnika/bolesnika

- **Glavni sudionik:** Administrator

- **Cilj:** Deaktivirati korisnički račun djelatnika/bolesnika.

- **Sudionici:** Baza podataka

- **Preduvjet:** Administrator je prijavljen u sustav.

- **Opis osnovnog tijeka:**

1. Administrator iz kontrolne ploče odabire pregled svih djelatnika ili bolesnika.
2. Administrator odabire djelatnika/bolesnika čiji korisnički račun želi deaktivirati.
3. Prije potvrde o promjeni administrator ponovno unosi svoju šifru

#### UC20 - Deaktivacija vlastitog korisničkog računa

- **Glavni sudionik:** Bolesnik

- **Cilj:** Deaktivirati vlastiti korisnički račun.

- **Sudionici:** Baza podataka

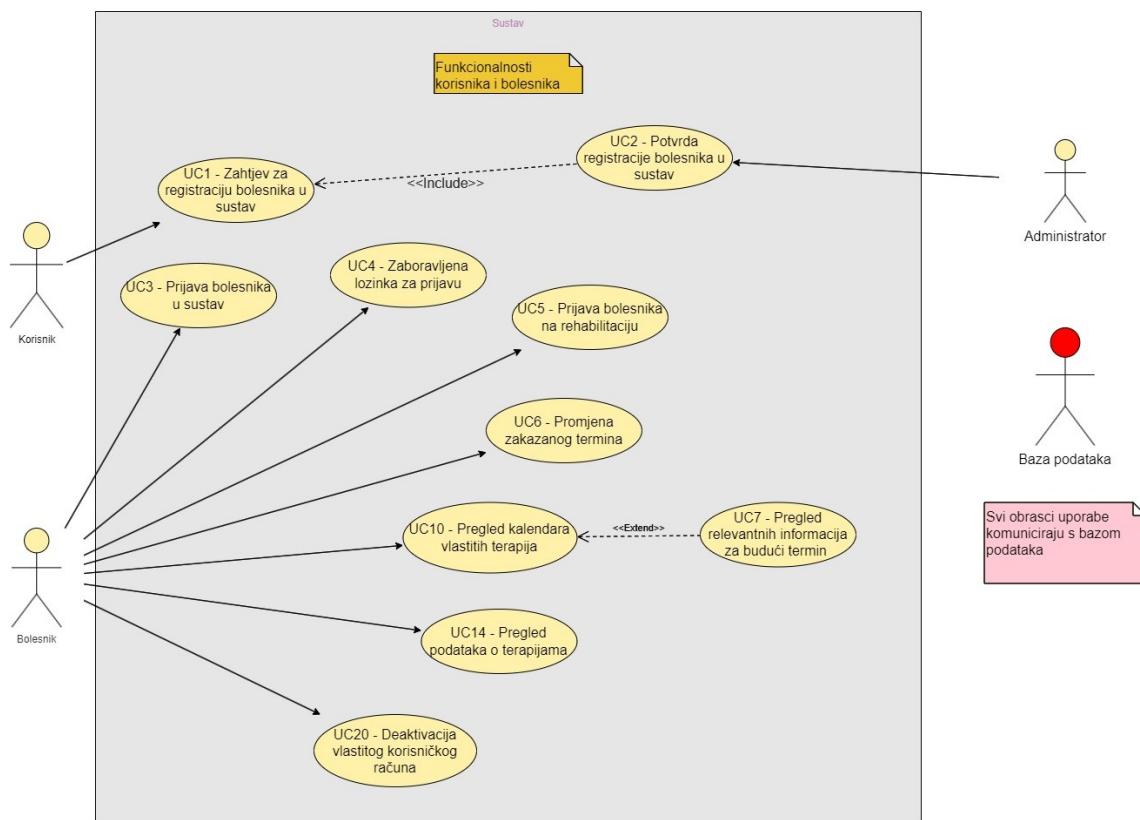
- **Preduvjet:** Bolesnik je prijavljen u sustav.

- **Opis osnovnog tijeka:**

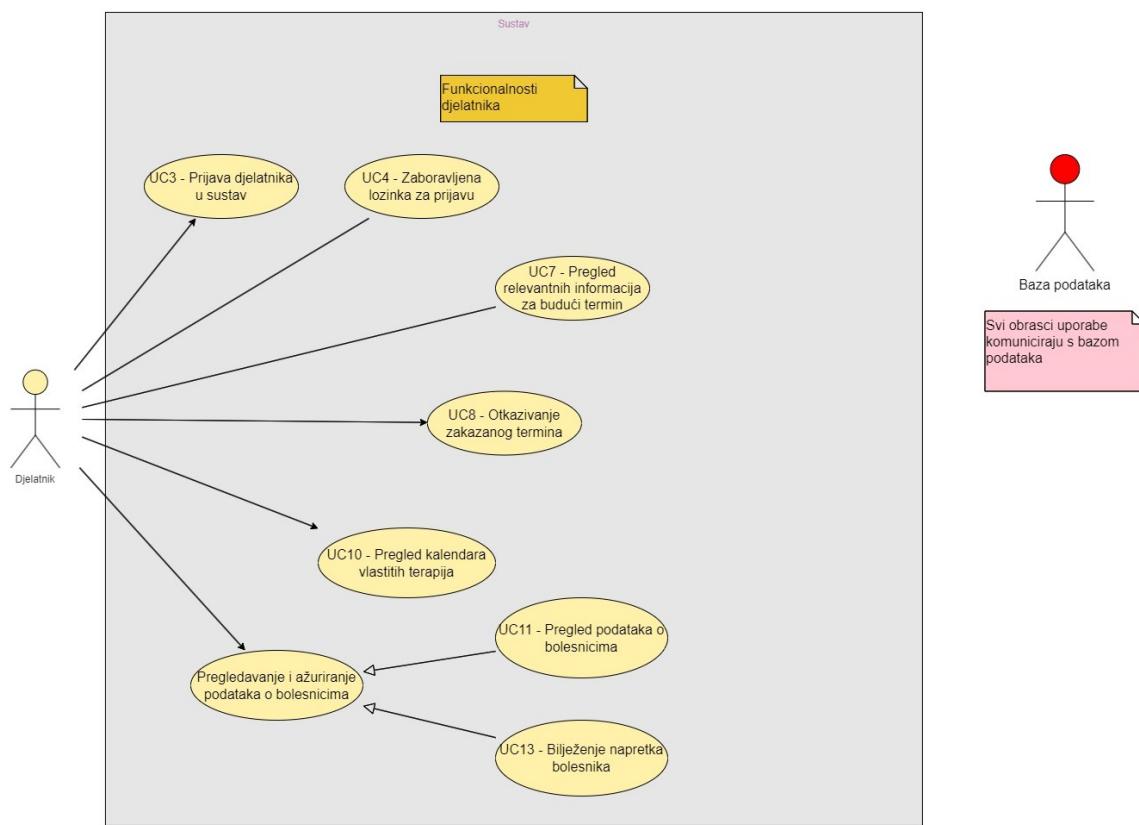
1. Bolesnik odabire opciju za deaktivaciju računa.
2. Sustav traži bolesnikovu lozinku za potvrdu zahtjeva.
3. Sustav je deaktivirao bolesnikov korisnički račun.

- **Opis mogućih odstupanja:** Sustav reagira na neispravnu lozinku te obavještava bolesnika o neuspjelom pokušaju.

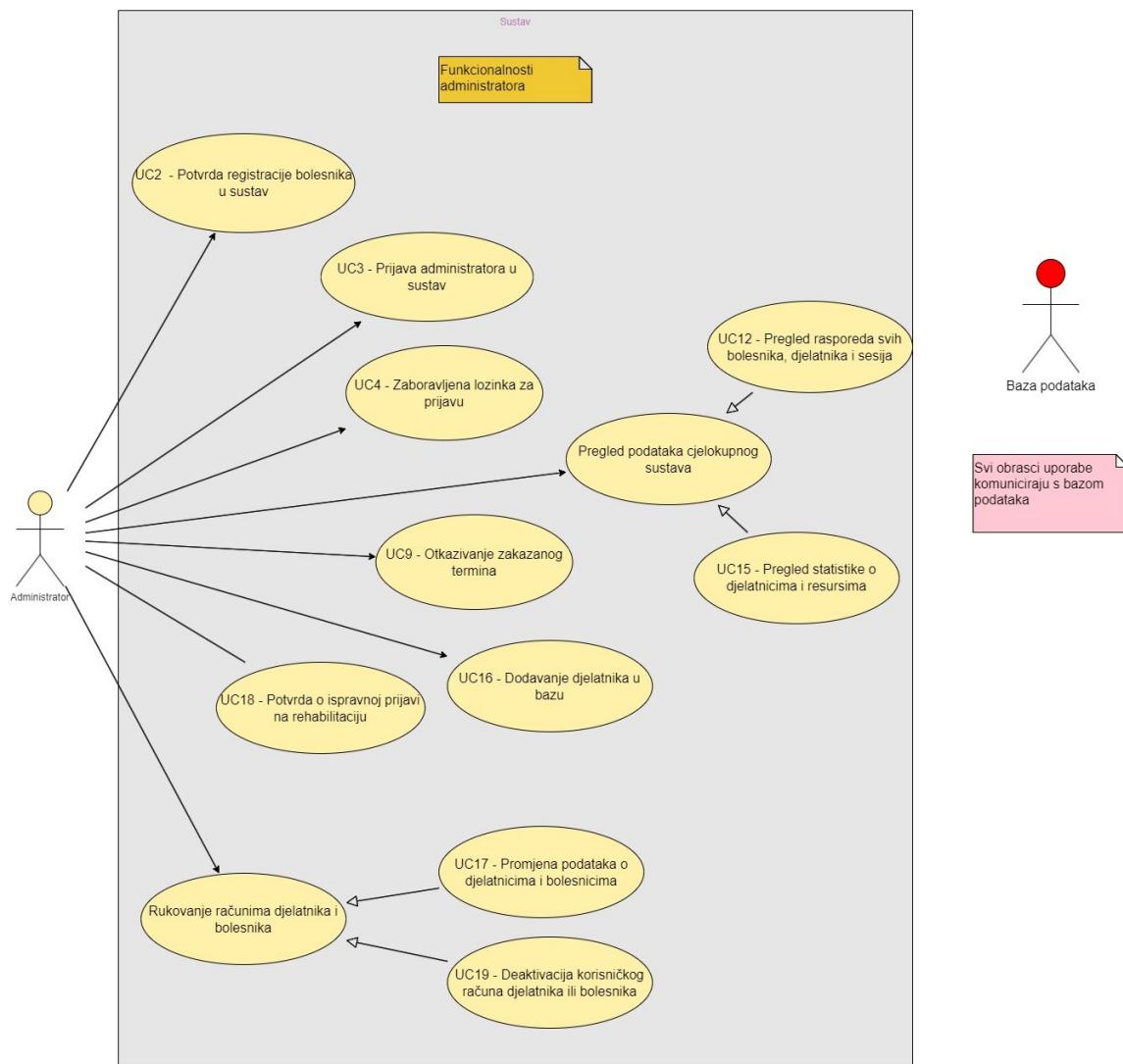
## Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost bolesnika



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost djelatnika



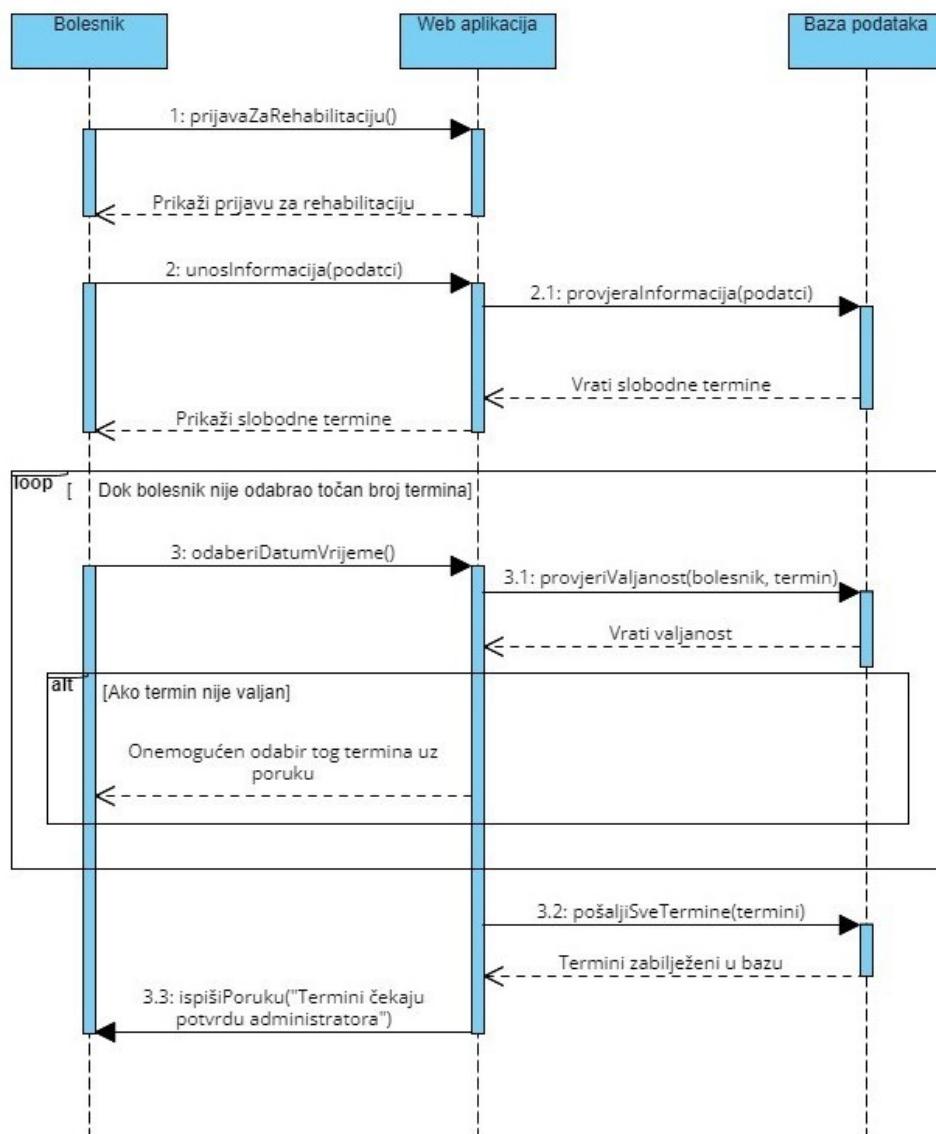
Slika 3.3: Dijagram obrasca uporabe, funkcionalnost administratora

### 3.1.2 Sekvencijski dijagrami

#### *dio 1. revizije*

##### **Obrazac uporabe UC5 – Prijava bolesnika na rehabilitaciju**

Za prijavu na rehabilitaciju, bolesnik pritišće na gumb s kojim dobiva od web aplikacije prikaz za prijavu te zatim unosi sve podatke o rehabilitaciji na koju se želi prijaviti. Te informacije se šalju bazi kako bi se provjerile te se, ako nije došlo do pogreške, vraćaju svi slobodni termini koje onda web aplikacija koristi za kreaciju kalendara slobodnih termina i to prikazuje bolesniku. Zatim, bolesnik mora odabrati onoliko termina koliko je naveo u prijavi te odabire svoje termine sve dok ne odabere toliko točnih termina. Svaki termin je određen svojim datumom i vremenom, jednom kada bolesnik odabere jedan termin, u bazi se provjerava slijedi li taj termin sva pravila (postoji minimalan razmak među terminima koji se mora poštovati) te ako odabrani termin nije valjan, sustav bolesniku vraća odgovarajuću poruku, a taj se termin ne uzima u obzir. Ako je termin valjan, bolesnik može prijeći na izabiranje sljedećeg termina. Na kraju, kada je bolesnik odabrao točan broj valjanih termina, svi termini se šalju i zapisuju u bazu, a bolesniku se ispisuje poruka kako mora pričekati potvrdu administratora sustava.

sd UC5 - Prijava bolesnika na rehabilitaciju

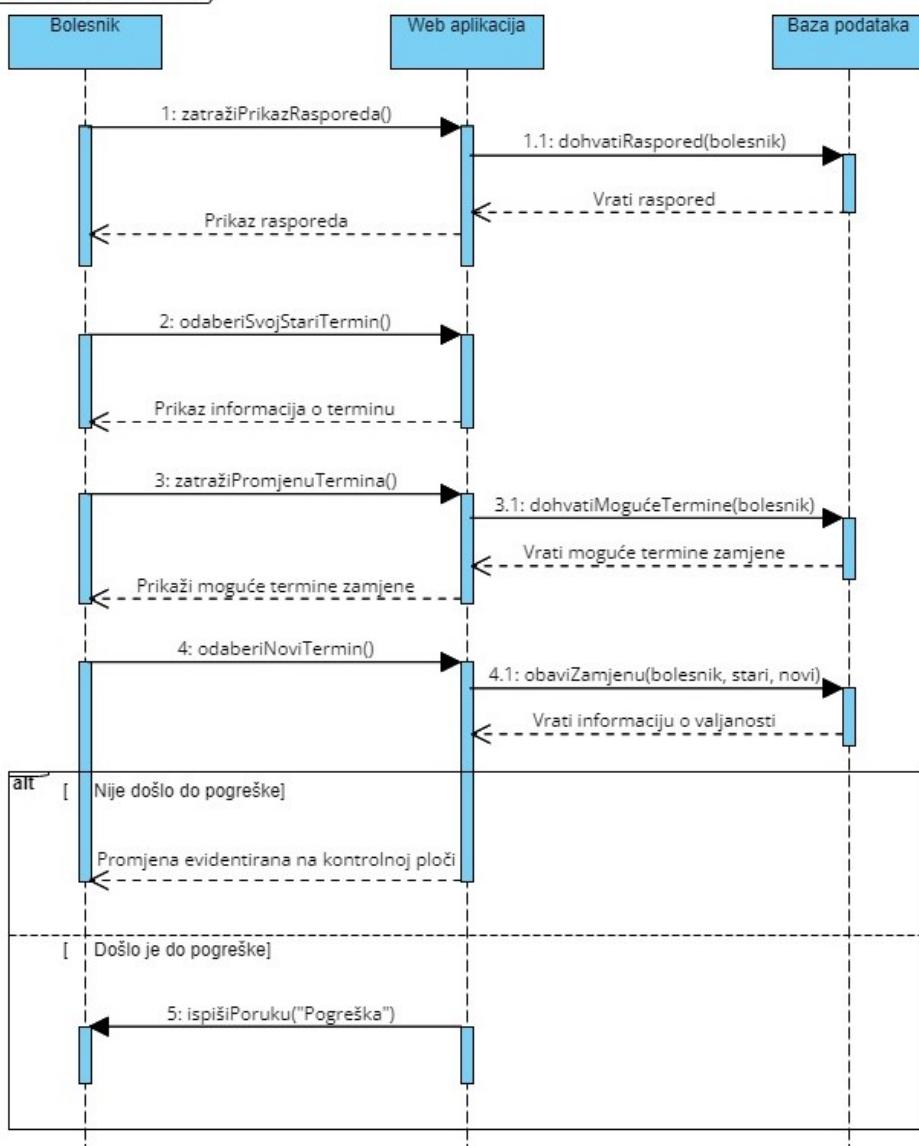
Slika 3.4: Sekvencijski dijagram za UC5

**Obrazac uporabe UC6 – Promjena zakazanog termina**

Bolesnik pritiskom na odgovarajući gumb dobiva prikaz svojeg rasporeda koji aplikacija dohvaća iz baze te na tom rasporedu odabire termin koji mu je dodijeljen, a koji bi želio promijeniti. Pojavljuju se dodatne informacije o odabranom terminu i također gumb za mijenjanje termina, pritiskom na taj gumb, web aplikacija dohvaća sve moguće termine zamjene za tog bolesnika i za tu rehabilitaciju i prikazuje ih bolesniku. Od ponuđenih termina, bolesnik odabire jedan te web aplikacija obavlja zamjenu u bazi podataka. Ako je zamjena prošla bez pogrešaka, promjena se odmah evidentira na kontrolnoj ploči dok u suprotnom bolesniku do-

lazi poruka o pogrešci.

sd UC6 - Promjena zakazanog termina

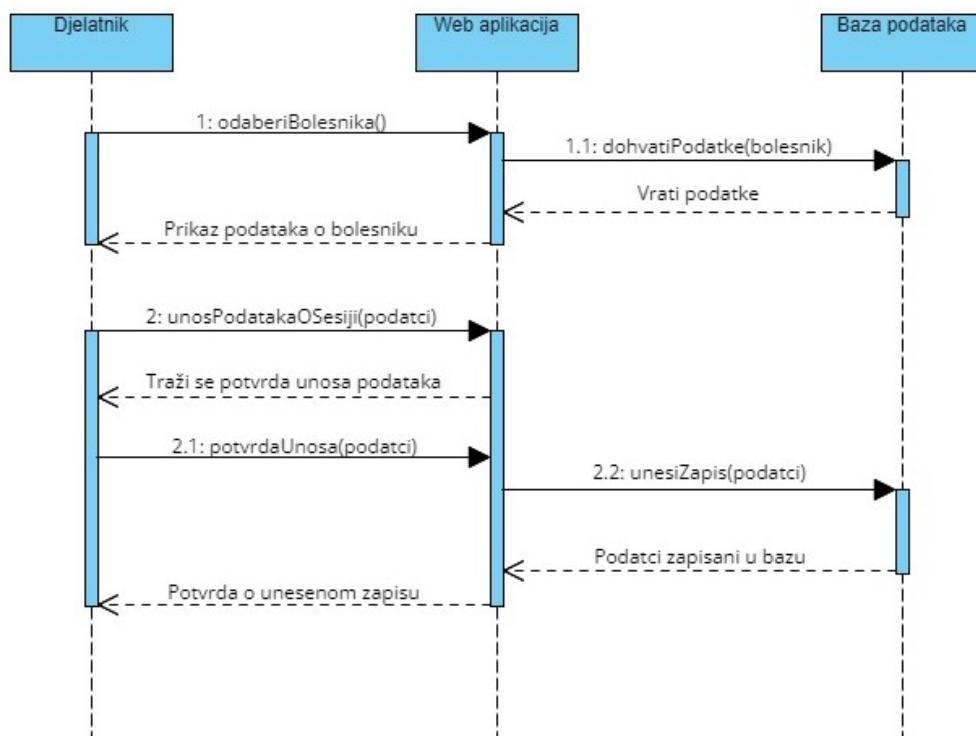


Slika 3.5: Sekvencijski dijagram za UC6

### Obrazac uporabe UC13 – Bilježenje napretka bolesnika

Djelatnik odabire bolesnika kojem želi zabilježiti napredak, a sustav vraća iz baze dohvaćene podatke o tom bolesniku. Djelatnik zatim piše svoj zapis o napretku, a kada završi, sustav od njega traži potvrdu. Jednom kada djelatnik potvrdi svoj zapis, web aplikacija ga šalje bazi podataka gdje se i upisuje. genijalan i prekoristan sekvencijski dijagram.

sd UC13 - Bilježenje napretka bolesnika

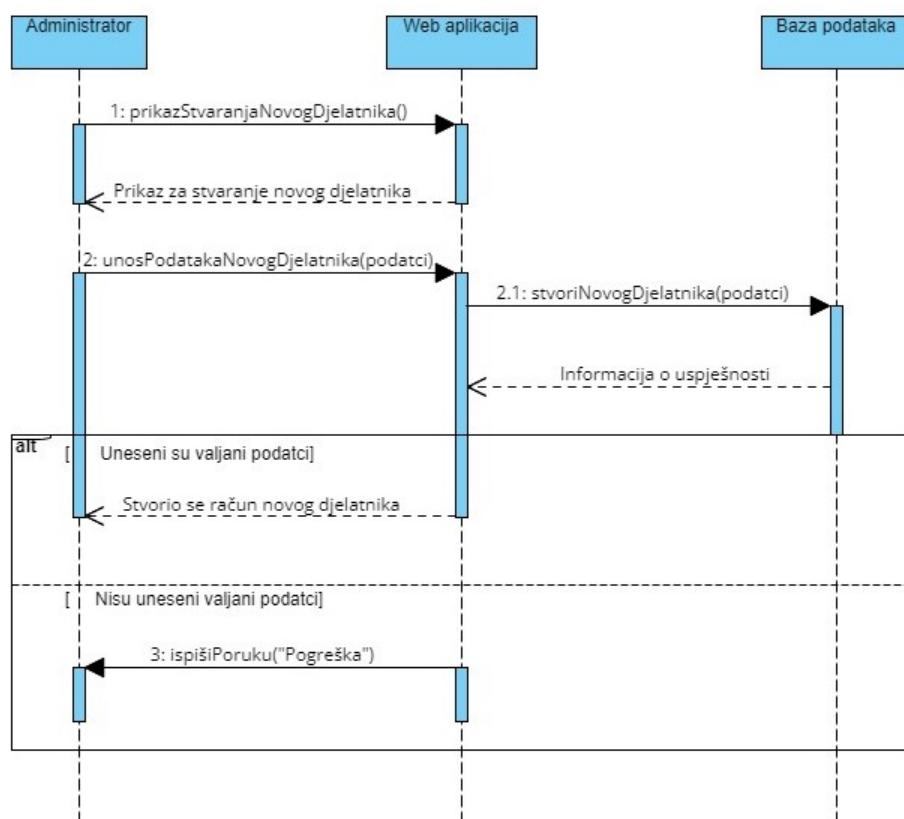


Slika 3.6: Sekvencijski dijagram za UC13

### Obrazac uporabe UC16 - Dodavanje djelatnika u bazu

Administrator odabire prikaz za stvaranje novog djelatnika, a web aplikacija mu ga vraća. Administrator zatim unosi sve podatke novog djelatnika kojeg želi dodati u sustav te se ti podatci s web aplikacije šalju bazi podataka. Baza podataka tada, ako nije došlo do pogreške, dodaje novog djelatnika u sustav, ako je do pogreške slučajno došlo, administrator se odgovarajućom porukom obavještava o njoj.

sd UC16 - Dodavanje djelatnika u bazu



Slika 3.7: Sekvencijski dijagram za UC16

## 3.2 Ostali zahtjevi

### dio 1. revizije

*Nefunkcionalni zahtjevi i zahtjevi domene primjene ključni su za uspješno funkcioniranje našeg sustava. Oni definiraju kako se sustav treba ponašati, koje standarde kvalitete mora zadovoljiti, te koje sigurnosne i tehničke ograničenja treba poštivati.*

### Performanse i pouzdanost

- Sustav treba podržavati simultani rad više korisnika bez gubitka performansi, osiguravajući pouzdano iskustvo u stvarnom vremenu.
- Vrijeme odziva prilikom pristupa bazi podataka i drugim ključnim funkcijama sustava ne smije trajati duže od nekoliko sekundi. Time se osigurava brz i efikasan rad sustava.

### Korisničko iskustvo

- Korisničko sučelje treba biti intuitivno i jednostavno za korištenje, uz minimalnu potrebu za opsežnim instrukcijama ili obukom.
- Sustav mora podržavati više jezika i odgovarajuće posebne znakove u različitim jezicima (starija populacija govori samo materinjim jezikom i sl.). Prva verzija aplikacije bit će na engleskom.
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava.

### Sigurnost i zaštita podataka

- Sustav mora osigurati visoku razinu sigurnosti, koristeći enkripciju i druge sigurnosne protokole, posebice u komunikaciji s bazom podataka i tijekom prijenosa osjetljivih podataka.
- Pritup sustavu mora biti omogućen kroz sigurne protokole kao što je HTTPS čime se osigurava zaštita podataka i privatnosti korisnika.

## Platforma i tehnička implementacija

- Sustav će biti implementiran kao web aplikacija, koristeći objektno-orientirane jezike, što omogućuje fleksibilnost i lakše održavanje.
- Aplikacija treba biti prilagodljiva i kompatibilna s različitim web i mobilnim platformama, osiguravajući široku dostupnost i pristupačnost.

## Skalabilnost i nadogradnje

- Sustav mora biti skalabilan kako bi podržao rast broja korisnika i povećanje opterećenja bez gubitka performansi.
- Nadogradnje sustava trebaju biti implementirane na način da ne narušavaju postojeće funkcionalnosti i da se mogu efikasno integrirati u operativni sustav.

## Standardi Kvalitete i usklađenost

- Sustav treba biti usklađen s nacionalnim i međunarodnim standardima kvalitete i sigurnosti, uključujući regulative vezane za zdravstvene informacijske sisteme.
- Redovite provjere kvalitete i revizije trebaju osigurati da aplikacija kontinuirano zadovoljava sve postavljene standarde.

## Dostupnost i pouzdanost

- Sustav treba osigurati visoku dostupnost i minimalno vrijeme prekida, posebno tijekom radnih sati zdravstvenih ustanova.
- Plan oporavka u slučaju kvara ili prekida treba biti jasno definiran i implementiran, osiguravajući brz povratak u normalan rad.

## Pristupačnost i inkluzivnost

- Sustav treba biti dizajniran s obzirom na pristupačnost, omogućavajući laku upotrebu osobama s različitim stupnjevima sposobnosti, uključujući stariju populaciju i osobe s invaliditetom.

- Također, treba podržavati prilagodbe za korisnike s posebnim potrebama poput prilagođenih fontova, kontrasta i pomoćnih tehnologija.

## Interoperabilnost i integracija

- Sustav treba biti interoperabilan s drugim zdravstvenim informacijskim sistemima, omogućujući razmjenu podataka i integraciju s postojećim IT infrastrukturnama.
- Integracija s vanjskim API-ima i servisima treba biti omogućena za proširivanje funkcionalnosti sustava kao što su integracije s laboratorijskim sustavima, elektroničkim zdravstvenim kartonima i slično.

## Održivost i ekološka osjećajnost

- Sustav treba biti razvijen s obzirom na ekološke aspekte, promovirajući smanjenje upotrebe papira i drugih materijala u zdravstvenim ustanovama.
- Digitalizacija procesa treba doprinijeti održivijem načinu rada, smanjujući otpad i potrebu za fizičkim resursima.

## Prilagodba zakonodavnim okvirima

- Sustav treba biti u skladu s lokalnim i međunarodnim zakonodavnim okvrima, uključujući propise o zaštiti osobnih podataka (npr. GDPR u EU).
- Moraju se implementirati mehanizmi za usklađenost s pravnim zahtjevima u vezi sa zdravstvenim informacijama kao što su prava na pristup, ispravak i brisanje osobnih podataka.

## Praćenje i izvještavanje

- Sustav treba omogućiti generiranje detaljnih izvještaja o korištenju resursa, učinkovitosti i performansama procesa rehabilitacije.
- Treba biti omogućeno praćenje korištenja aplikacije, uključujući analitiku i povratne informacije od korisnika za kontinuirano poboljšanje i prilagodbu sustava.

## Prilagodljivost budućim tehnološkim trendovima

- Sustav treba biti dizajniran s mogućnošću laganog ažuriranja i prilagodbe novim tehnološkim trendovima i inovacijama.
- Treba postojati mogućnost integracije s budućim tehnologijama poput umjetne inteligencije, strojnog učenja i napredne analitike podataka.

## 4. Arhitektura i dizajn sustava

*dio 1. revizije*

### Uvod u Arhitekturu Sustava

Naš sustav je dizajniran da bude efikasan, skalabilan i pouzdan. U nastavku detaljno opisujemo arhitekturu našeg sustava.

#### Izbor Arhitekture

- Naša odluka da koristimo trodijelnu arhitekturu temelji se na principima oblikovanja predstavljenim na predavanjima, gdje se ističe važnost modularnosti i odvajanja odgovornosti. Ovaj temeljni pristup oblikovanju sustava reflektira našu predanost stvaranju arhitekture koja je ne samo funkcionalna, već i prilagodljiva, jednostavna za održavanje te otporna na buduće izazove.
- Princip modularnosti naglašava važnost razdvajanja sustava na manje, samostalne dijelove, ili module. Ovakav pristup omogućava da svaki modul ima jasno definiranu funkcionalnost, što pojednostavljuje razvoj, testiranje i održavanje. Svaki modul može biti razvijen neovisno, što pridonosi većoj fleksibilnosti i olakšava integraciju novih značajki ili promjena u sustavu.
- Odvajanje odgovornosti, s druge strane, znači da svaki dio sustava ima precizno definiranu ulogu ili odgovornost. Na primjer, frontend se odvaja od backenda, čime se postiže jasna granica između korisničkog sučelja i poslovne logike. Ovo olakšava praćenje i održavanje svake komponente zasebno, smanjuje rizik od pogrešaka i omogućava paralelno razvijanje dijelova sustava.
- Trodijelna arhitektura, koja obuhvaća backend, frontend i bazu podataka, idealno se uklapa u ove principe. Backend je odgovoran za poslovnu logiku,

frontend za korisničko sučelje, dok je baza podataka centralno mjesto za pohranu podataka. Ovaj pristup omogućava svakom dijelu sustava da obavlja svoju specifičnu funkciju, čime se postiže bolja organizacija, održavanje i skalabilnost.

## Organizacija Sustava

- Naš sustav je pažljivo organiziran na visokoj razini apstrakcije, koristeći klijent-poslužitelj model, što predstavlja ključni element naše arhitekture. Ova organizacija ima za cilj efikasno upravljanje različitim dijelovima sustava, pružajući jasno odvojeni pristup korisničkom sučelju i poslovnoj logici.
- **Klijent-poslužitelj** model je arhitektonski oblik koji omogućuje razdvajanje funkcionalnosti između dviju osnovnih komponenti: klijenta i poslužitelja. Klijent predstavlja korisničko sučelje, dok poslužitelj sadrži poslovnu logiku i podatke. Ovaj model omogućuje svakoj komponenti da obavlja svoje specifične zadatke, što rezultira modularnošću i skalabilnošću sustava.

### Prednosti klijent-poslužitelj modela:

1. Razdvajanje odgovornosti: Klijent i poslužitelj imaju jasno definirane uloge, čime se postiže precizno odvajanje korisničkog sučelja od poslovne logike. To olakšava održavanje, poboljšava sigurnost i doprinosi boljoj organizaciji koda.
2. Fleksibilnost i skalabilnost: Modularnost klijent-poslužitelj arhitekture omogućava prilagodbu svake komponente neovisno. Na primjer, možemo nadograditi ili zamijeniti korisničko sučelje bez narušavanja poslovne logike i obrnuto. Ovo čini sustav fleksibilnim i lako skalabilnim.
3. Efikasna komunikacija: Klijent i poslužitelj komuniciraju putem standardiziranih protokola, često kroz HTTP. Ova jasna komunikacija omogućava brzu i pouzdanu razmjenu podataka između dijelova sustava.

Odvajanje korisničkog sučelja od poslovne logike donosi dodatne prednosti. Korisničko sučelje, koje može biti web aplikacija ili mobilna aplikacija, fokusira se na prezentaciju podataka i interakciju s korisnicima. S druge strane, poslovna logika centralizirana je na poslužitelju, gdje se vrše obrade podataka, donose poslovne odluke i upravlja cjelokupnim tokom aplikacije.

## Detalji Podsustava

### Backend (Spring Boot)

- Backend predstavlja ključnu osnovu našeg sustava, pružajući infrastrukturu za obradu poslovnih zahtjeva. Organiziran je kao skup podsustava, uključujući servise i kontrolere, čija je svrha efikasno rukovanje različitim aspektima poslovnih funkcionalnosti.

**Servisi:** Funkcionalne jedinice sustava odgovorne za izvođenje specifičnih poslovnih operacija. Svaki servis fokusira se na određenu funkcionalnost, pridonoseći modularnosti i omogućavajući bolje upravljanje kompleksnošću sustava. Na primjer, možemo imati servis za korisničke operacije, proizvode ili neki drugi poslovni entitet.

**Kontroleri:** Odgovorni za obradu i usmjeravanje HTTP zahtjeva koji dolaze od klijenata. Kontroleri predstavljaju sučelje između korisničkog sučelja (frontend) i poslovnih operacija koje izvode servisi. Ovo odvajanje odgovornosti omogućava bolju organizaciju koda i olakšava testiranje.

### Komunikacija s bazom podataka (PostgreSQL):

- Backend komunicira s bazom podataka putem PostgreSQL sustava, koji omogućava efikasnu organizaciju i manipulaciju podacima zbog svoje pouzdane i robustne arhitekture. Relacijski model podataka pruža strukturu temeljenu na tablicama, što olakšava povezivanje podataka između različitih dijelova sustava.
- Ovaj pristup omogućava lako preslikavanje podataka iz aplikacije u bazu i obrnuto, pridonoseći dosljednosti podataka i olakšavajući njihovo upravljanje. Kroz PostgreSQL, ostvarujemo centralno spremište podataka koje je ključno za sve relevantne informacije u sustavu.

Backend, kao središnja komponenta sustava, osigurava da poslovna logika bude učinkovito izvršavana, a podaci pravilno organizirani i održavani. Ovaj dio arhitekture pridonosi integraciji svih poslovnih funkcionalnosti, čime se postiže koherentan i efikasan rad sustava.

**Frontend (Node.js, JavaScript, React)** Frontend, kao ključna korisnička strana našeg sustava, organiziran je prema modernim principima razvoja web aplikacija.

Glavni tehnološki alati koje koristimo su Node.js, JavaScript i React, pružajući nam moćne alate za stvaranje fleksibilnog i intuitivnog korisničkog sučelja.

Organizacija u komponente i kontejnere:

- Komponente su modularni dijelovi koji obavljaju specifične zadatke, kao što su prikazivanje određenih podataka ili omogućavanje korisničkih interakcija.
- Kontejneri služe kao viši slojevi koji upravljaju komponentama, pružajući organiziran pristup i poboljšavajući skalabilnost.

REST API za komunikaciju s backendom:

- Frontend komunicira s backendom putem REST API-ja (Representational State Transfer), standardnog protokola za komunikaciju između klijenta i poslužitelja koji omogućava brzu, standardiziranu i pouzdanu razmjenu podataka između frontend i backend dijelova sustava.

React kao glavni framework:

- Koristimo React, popularni JavaScript framework za izgradnju korisničkih sučelja. React omogućava brzo renderiranje stranica, poboljšava učinkovitost i olakšava manipulaciju komponentama.
- Kroz React, frontend postaje reaktivno sučelje koje brzo reagira na promjene stanja podataka, pružajući korisnicima ugodno iskustvo korištenja aplikacije.

### Baza podataka (PostgreSQL)

- Baza podataka predstavlja ključnu komponentu našeg sustava, pružajući centralno spremište za sve relevantne podatke.
- U našem slučaju, koristimo PostgreSQL, robustan i napredan relacijski sustav upravljanja bazama podataka, kako bismo osigurali pouzdanu pohranu i učinkovit pristup podacima.
- Struktura baze je dizajnirana da podržava složene upite i relacije između različitih podataka.

## Mrežni Protokoli i Komunikacija

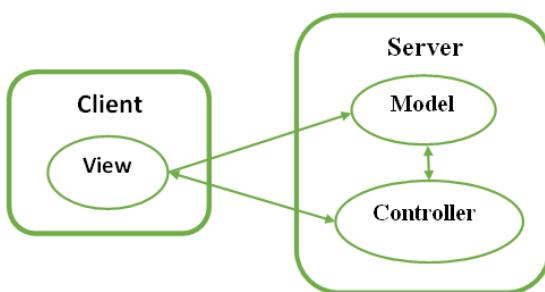
- Komunikacija između klijenta i poslužitelja odvija se preko HTTP protokola, osiguravajući sigurnost i standardiziran protokol informacija.
- Kroz HTTP, klijent šalje zahtjeve poslužitelju, a poslužitelj šalje odgovore.

## Globalni Upravljački Tok

- Globalni upravljački tok u našem sustavu određen je kroz primjenu MVC (Model-View-Controller) arhitekture, pružajući strukturu i organizaciju za učinkovitu interakciju između različitih dijelova aplikacije.

### MVC Arhitektura:

- **Model:** Model predstavlja središnju komponentu sustava koja upravlja podacima. Ovdje se nalaze podaci i poslovna logika aplikacije. Model je odvojen od korisničkog sučelja i interakcije te služi kao zasebna jedinica koja obrađuje podatke neovisno o korisničkom sučelju.
- **View:** View je odgovoran za prikazivanje podataka korisnicima. Ova komponenta generira korisničko sučelje na temelju podataka iz Modela. View je pasivan i ne obavlja izravne operacije na podacima, već ih prikazuje korisnicima na razumljiv način.
- **Controller:** Controller predstavlja upravljački sloj koji obrađuje korisničke zahtjeve i interakcije. Kada korisnik obavi neku akciju, Controller reagira, a zatim komunicira s Modelom i Viewom prema potrebi. Ova komponenta osigurava odvajanje korisničkih akcija od same logike sustava.



Slika 4.1: MVC arhitektura

### Prednosti MVC Arhitekture:

- Odvajanje odgovornosti: MVC omogućava jasno odvajanje odgovornosti između različitih dijelova sustava. Model, View i Controller imaju specifične uloge, čime se olakšava razvoj, testiranje i održavanje koda.
- Lako proširivost i održavanje: Zbog modularnosti i odvajanja odgovornosti, dodavanje novih funkcionalnosti ili promjena postojećih dijelova sustava postaje jednostavno i manje riskantno.
- Jasna struktura: MVC pruža jasnu strukturu sustava, olakšavajući suradnju između različitih timova i programera. Ova organizacija doprinosi boljoj preglednosti i razumijevanju koda.

### Sklopovalno-Programski Zahtjevi

- Integracijom sklopovalno-programskih zahtjeva, sustav omogućuje stabilnost te se uspješno priprema za izazove suvremenog informacijskog okruženja.

**Prilagođenost modernim računalnim platformama:** Sustav je optimiziran za rad na suvremenim računalnim platformama, uključujući njihove specifičnosti i prednosti. Koristeći najnovije tehnologije i prakse, osiguravamo kompatibilnost s različitim operativnim sustavima i konfiguracijama računala.

**Visoka dostupnost:** Dostupnost sustava ključna je za neprekidno pružanje usluga korisnicima. Implementiramo strategije visoke dostupnosti kako bismo osigurali minimalne prekide u radu sustava. Ovo uključuje redundanciju ključnih komponenti i mehanizme automatskog oporavka od mogućih kvarova.

**Optimizacija performansi:** Performanse su ključne za korisničko iskustvo i učinkovit rad sustava. Kroz optimizaciju koda, uporabu efikasnih algoritama i praćenje performansi u stvarnom vremenu, postižemo visoku razinu odziva i brze obrade podataka.

**Prilagodljivost za skaliranje:** Sustav je dizajniran kako bi se lako skalirao prema rastućim potrebama. Uz mogućnost horizontalnog i vertikalnog skaliranja, omogućujemo prilagodljivu infrastrukturu koja se može dinamički prilagoditi

povećanju opterećenja.

**Sigurnosni aspekti:** Sklo povsko-programski zahtjevi također obuhvaćaju sigurnosne aspekte. Sustav implementira mjere zaštite podataka, enkripciju komunikacije i autentikaciju kako bi osigurao siguran rad i spriječio neovlašteni pristup.

#### Kontinuirano ažuriranje i praćenje trendova:

S obzirom na brzu evoluciju tehnologije, sustav se kontinuirano prilagođava novim standardima i trendovima. Redovita ažuriranja omogućuju iskorištavanje najnovijih značajki, poboljšanje sigurnosti te održavanje konkurentske prednosti.

Kroz integraciju navedenih sklo povsko-programskih zahtjeva, naš sustav ne samo da osigurava stabilnost i performanse, već je i spreman suočiti se s izazovima suvremenog informacijskog okruženja. Ovaj pristup jamči dugoročnu održivost, skalabilnost i optimalno iskorištavanje resursa računalnih platformi.

## 4.1 Baza podataka

### *dio 1. revizije*

Baza podataka predstavlja ključnu komponentu našeg sustava, pružajući centralno spremište za sve relevantne podatke. U našem slučaju, koristimo PostgreSQL, relacijski sustav upravljanja bazama podataka. PostgreSQL je snažan i visoko priлагodljiv sustav za upravljanje bazama podataka, što ga čini prikladnim izborom za učinkovito i sigurno upravljanje zdravstvenim podacima, osiguravajući da su informacije o pacijentima, terminima, zdravstvenom osoblju i opremi sigurno pohranjene i dostupne.

#### Pouzdana pohrana i pristup podacima:

PostgreSQL pruža visoku razinu pouzdanosti i integriteta podataka. Njegova transakcijska podrška osigurava dosljednost podataka čak i u slučaju neplaniranih prekida rada sustava ili pogrešaka. Centralizirano pohranjivanje podataka omogućava nam učinkovito upravljanje svim informacijama relevantnim za rad sustava.

#### Struktura baze za složene upite i relacije:

- Dizajn baze podataka ključan je za podršku složenim upitima i održavanje

relacija između različitih podataka.

- PostgreSQL, kao relacijski sustav, koristi tablice kako bi organizirao podatke. Svaka tablica sastoji se od redova i stupaca, omogućujući jasno definiranje veza između entiteta.
- Veze između tablica omogućavaju složene upite koji spajaju informacije iz različitih dijelova sustava. PostgreSQL također podržava napredne mogućnosti poput indeksiranja, pomažući ubrzati upite i optimizirati performanse baze podataka.
- Struktura baze podataka je pažljivo oblikovana kako bi odražavala logičke relacije i potrebe sustava, čime se postiže efikasno upravljanje podacima.

#### 4.1.1 Opis tablica

Svaku tablicu je potrebno opisati po zadanom predlošku. Lijevo se nalazi točno ime varijable u bazi podataka, u sredini se nalazi tip podataka, a desno se nalazi opis varijable. Svjetlozelenom bojom označite primarni ključ. Svjetlo plavom označite strani ključ

**\_User** Ovaj entitet predstavlja korisnike sustava s različitim ulogama. Sadrži atribute kao što su status korisnika (active), jedinstveni identifikator (id), e-mail adresa (email), ime (first\_name), prezime (last\_name), lozinka (password) i uloga korisnika (role). Provjerava se uloga korisnika kroz ograničenje \_user\_role\_check. Ovaj entitet omogućuje upravljanje korisnicima sustava i dodjeljivanje uloga kako bi se pristup određenim funkcionalnostima kontrolirao. Primarni ključ (\_user\_pkey) je definiran na atributu id.

<b>_user</b>		
<b>id</b>	BIGINT	Jedinstveni identifikator korisnika
active	BOOLEAN	Označava je li korisnik aktivan
email	VARCHAR	E-mail adresa korisnika
first_name	VARCHAR	Ime korisnika
last_name	VARCHAR	Prezime korisnika
password	VARCHAR	Lozinka korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>_user</b>		
role	VARCHAR	Rola korisnika (mora biti jedna od predefiniranih uloga)

**Imenik** Entitet koji služi za pohranu informacija o liječnicima. Sadrži atribute kao što su jedinstveni identifikator liječnika (hlkid), ime (first\_name), prezime (last\_name), specijalizacija (specialization) i informacija o aktivnosti liječnika (active). Primarni ključ (Imenik\_pkey) je definiran na atributu hlkid.

<b>imenik</b>		
hlkid	VARCHAR	Jedinstveni identifikator liječnika
first_name	VARCHAR	Ime liječnika
last_name	VARCHAR	Prezime liječnika
specialization	VARCHAR	Specijalizacija liječnika
active	BOOLEAN	Označava je li liječnik u imeniku aktivna

**Appointment** Čuva informacije o terminima unutar sustava. Sadrži atribute kao što su datum i vrijeme termina (date\_time), identifikator zaposlenika (employee\_id), jedinstveni identifikator termina (id), identifikator pacijenta (patient\_id), identifikator sesije (session\_id), identifikator terapije (therapy\_id) i status termina (status). Ovaj entitet je povezan s drugim entitetima (employee, patient, session, therapy) putem različitih Many-to-One veza prema atributima employee\_id, patient\_id, session\_id, i therapy\_id.

<b>appointment</b>		
id	BIGINT	Jedinstveni identifikator termina
date_time	TIMESTAMP	Datum i vrijeme termina
employee_id	BIGINT	Identifikator zaposlenika koji obavlja termin
patient_id	BIGINT	Identifikator pacijenta koji ima termin
session_id	BIGINT	Identifikator sesije kojoj pripada termin

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

appointment		
therapy_id	BIGINT	Identifikator terapije koja se izvodi tijekom termina
status	VARCHAR	Status termina

**Employee** Ovaj entitet predstavlja tablicu koja sadrži informacije o zaposlenicima. Sadrži atribute kao što su specijalizacija (specialization) i identifikator (id). Ograničenje employee\_specialization\_check provjerava ispravnost specijalizacije. Primarni ključ (employee\_pkey) je definiran na atributu id. Povezan je One-to-Many vezom s entitetom session preko atributa employee\_id.

employee		
id	BIGINT	Jedinstveni identifikator zaposlenika
specialization	SMALLINT	Stručnost zaposlenika

**Equipment** Ovaj entitet predstavlja tablicu koja sadrži informacije o opremi. Svaka oprema ima svoj jedinstveni identifikator (id) za precizno praćenje. Atribut capacity označava kapacitet opreme, dok description pruža dodatne opise ili karakteristike. Naziv opreme (name) jedinstveno označava svaki komad, olakšavajući identifikaciju i korištenje unutar medicinskog okruženja.

equipment		
id	BIGINT	Jedinstveni identifikator opreme
capacity	INTEGER	Kapacitet opreme
description	VARCHAR	Opis opreme
name	VARCHAR	Naziv opreme

**Patient** Ovaj entitet predstavlja tablicu koja sadrži informacije o pacijentima. Sadrži atribute kao što su datum rođenja, identifikator (id), adresa, MBO (matični broj osiguranika) i broj telefona. Primarni ključ (patient\_pkey) je definiran na atributu id.

patient		
id	BIGINT	Jedinstveni identifikator pacijenta
date_of_birth	DATE	Datum rođenja pacijenta
address	VARCHAR	Adresa pacijenta
mbo	VARCHAR	Matični broj osiguranika pacijenta
phone_number	VARCHAR	Broj telefona pacijenta

**Session** Ovaj entitet predstavlja tablicu koja sadrži informacije o sesijama. Sadrži atributе као што су датум и vrijeme, идентификатор зaposlenika, идентификатор сесије и повратна информација (feedback). Primarni ključ (session\_pkey) je definiran на атрибуту id. Пovezan je Many-to-One vezom s entitetом employee preko атрибута employee\_id.

session		
id	BIGINT	Jedinstveni identifikator sesije
date_time	timestamp	Vrijeme sesije
employee_id	BIGINT	Identifikator zaposlenika
feedback	character varying(255)	Povratna informacija o sesiji

**Therapy** Ovaj entitet predstavlja tablicu која садржи информације о терапијама. Садржи атрибут идентификатор (id) и атрибут therapy\_type\_id. Primarni ključ (therapy\_pkey) je definiran на атрибуту id. Такође садржи јединствено ограничење therapy\_therapy\_type\_id\_key на атрибуту therapy\_type\_id. Повезан је One-to-One vezom са entitetom appointment preko атрибута therapy\_id.

therapy		
id	BIGINT	Jedinstveni идентификатор терапије
therapy_type_id	BIGINT	Идентификатор врсте терапије

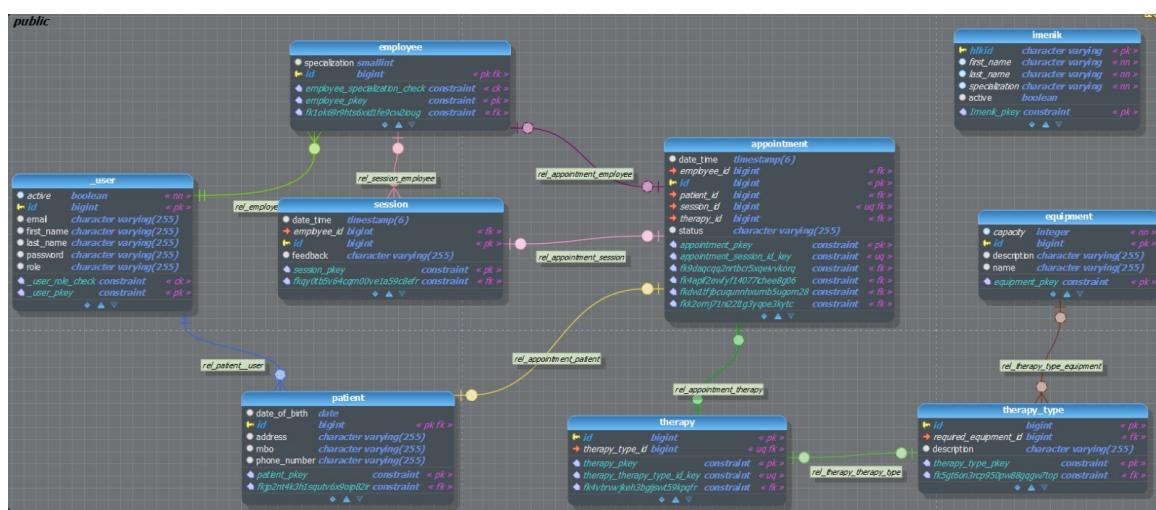
**Therapy\_type** Ovaj entitet predstavlja tablicу која садржи информације о врстама терапија. Садржи атрибуте као што су идентификатор (id), идентификатор потребне опреме (required\_equipment\_id) и опис. Primarni ključ (therapy\_type\_pkey) je definiran на

atributu id. Povezan je Many-to-One vezom s entitetom therapy preko atributa required\_equipment\_id i s entitetom equipment preko atributa required\_equipment\_id.

therapy_type		
id	BIGINT	Jedinstveni identifikator vrste terapije
required_equipment_id	BIGINT	Identifikator potrebnog opreme
description	character varying(255)	Opis vrste terapije

#### 4.1.2 Dijagram baze podataka

Svaka tablica ima primarni ključ koji jedinstveno identificira zapise unutar tablice. Baza podataka uspostavlja odnose između tablica koristeći strane ključeve. Ograničenja su postavljena kako bi se očuvao integritet podataka.

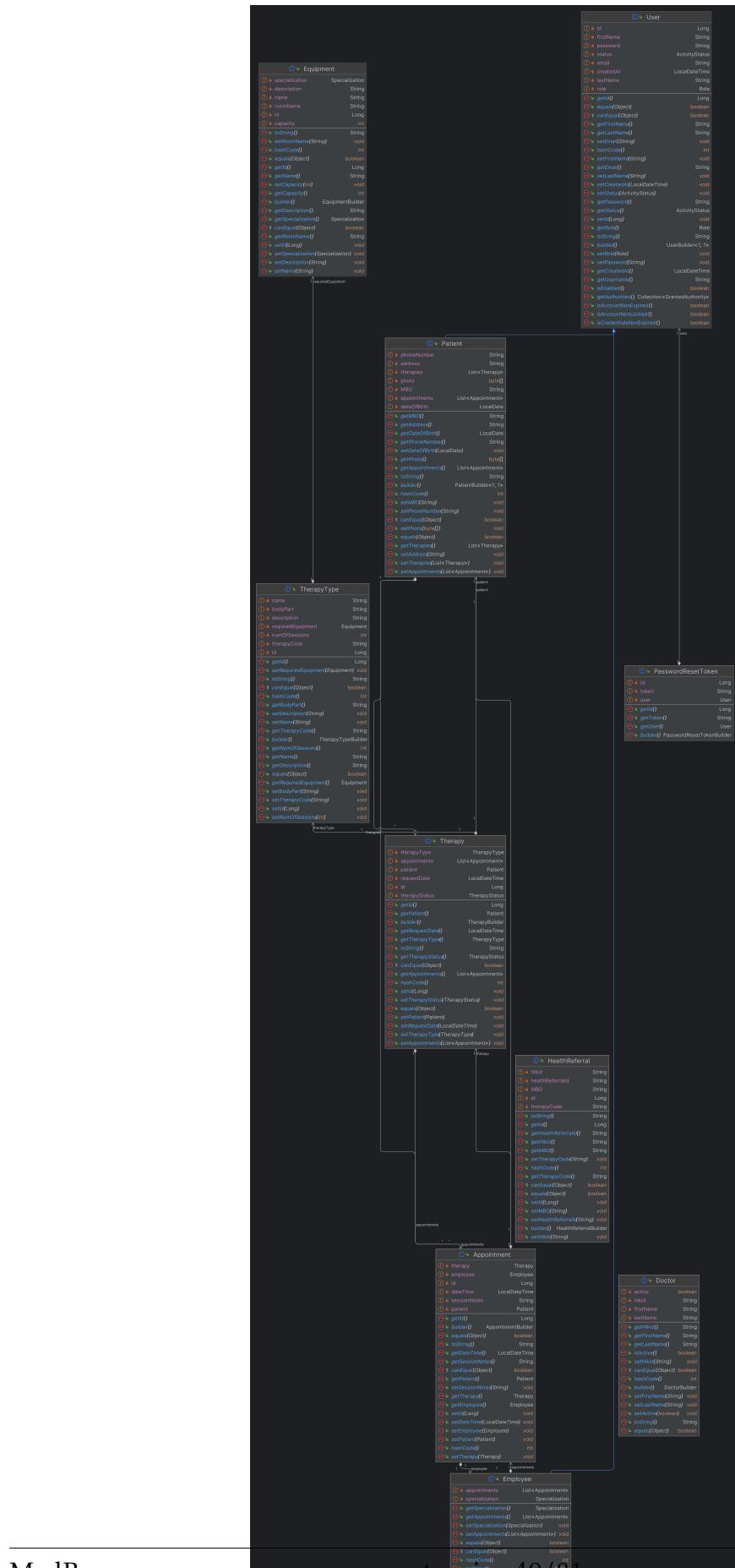


Slika 4.2: Dijagram baze podataka

## 4.2 Dijagram razreda

Dijagram razreda prikazan je na slici 4.3. Implementirane metode direktno komuniciraju s bazom podataka te vraćaju tražene podatke poput popisa dijelatnika, opreme, provjere je li dijelatnik i dalje aktivan.

Razredi Employee i Patient generalizacija su razreda User pa nasljeđuju njegove public atrIBUTE i metode. Razred User predstavlja osnovni model korisnika u sustavu. Ovaj razred obuhvaća osnovne informacije o korisniku, uključujući ime, prezime, korisničko ime profila, email adresu, lozinku i status aktivnosti korisničkog računa. Dodatno, svaki korisnik ima dodijeljenu ulogu koja se koristi za upravljanje ovlastima unutar sustava, a ta uloga je opisana pomoću enumeracije putem razreda Role. Razred Employee predstavlja model zaposlenika u medicinskom sustavu. Proširuje osnovni razred User, dodajući specifične atrIBUTE i odnose relevantne za zaposlenike u sustavu. Razred Patient predstavlja model pacijenta u medicinskom sustavu. Razred Specialization predstavlja enumeraciju koja sadrži različite vrste specijalizacija dijelatnika u medicinskom sustavu. Razred Session predstavlja radnu sesiju unutar medicinskog sustava. Ovaj razred omogućuje praćenje sesija između zaposlenika i pacijenata. Zahtjev za terapijom se sastavlja pomoću vrste terapije i identifikatora pacijenta, zaposlenika i terapije. Razred Equipment predstavlja opremu koja se koristi u medicinskom sustavu. Ovaj razred pruža informacije o različitim vrstama opreme, uključujući jedinstveni identifikator opreme, kapacitet, opis i naziv opreme. Razred Therapy predstavlja terapiju unutar medicinskog sustava. Razred TherapyType predstavlja vrstu terapije u sustavu.



Slika 4.3: Dijagram razreda

Svaki od ovih entiteta slijedi obrazac Controller -> Service -> Repository za obavljanje operacija nad podacima.

Na primjer, entitet Employee posjeduje EmployeeController, EmployeeService i EmployeeRepository. Ovi razredi zajednički omogućuju upravljanje informacija o zaposlenicima, uključujući njihovu autentikaciju. Kroz ovaj ciklus, Controller komunicira s Servisom, a Servis dalje s Repozitorijem, osiguravajući konzistentan i siguran pristup podacima.

Osim toga, entiteti poput Therapy, Appointment i Session imaju svoje setove Controllera, Servisa i Repozitorija koji omogućuju operacije vezane uz terapije, narudžbe pacijenata i sesije.

Bitna karakteristika ovog organiziranog pristupa je da Controller vraća odgovarajući HTTP status i tražene informacije (klasa, string ili ništa) u obliku JSON-a. Ovaj proces olakšava komunikaciju između različitih dijelova sustava, pružajući jasnu i strukturiranu komunikaciju među različitim slojevima aplikacije.

Kad Patient izvrši registraciju ili prijavu, procesi autentikacije i autorizacije upravljaju se putem SecurityControllera i SecurityServicea. Ovi kontroleri i servisi surađuju s SecurityConfig-om, implementiranim pomoću WebConfig, kako bi provjerili je li pacijent, administrator ili zdravstveni djelatnik autoriziran i autenticiran za svaku radnju koju žele izvršiti u sustavu.

SecurityController omogućuje upravljanje zahtjevima vezanim uz sigurnost, poput registracije i prijave korisnika, dok SecurityService sadrži logiku autentikacije i autorizacije. Ove komponente zajednički rade kako bi osigurale siguran pristup podacima i funkcionalnostima sustava.

Kroz SecurityConfig i WebConfig, postavke sigurnosti konfiguiraju se prema tipu korisnika. Ovaj postupak osigurava da pacijenti, administratori i zdravstveni djelatnici imaju odgovarajuće privilegije za izvršavanje želenih radnji, uzimajući u obzir autentikaciju i autorizaciju.

Ova struktura omogućuje efikasno i sigurno upravljanje pristupom resursima u sustavu, osiguravajući da svaki korisnik ima odgovarajuće ovlasti za obavljanje svojih funkcija u skladu s postavljenim sigurnosnim pravilima.

## 4.3 Dijagram stanja

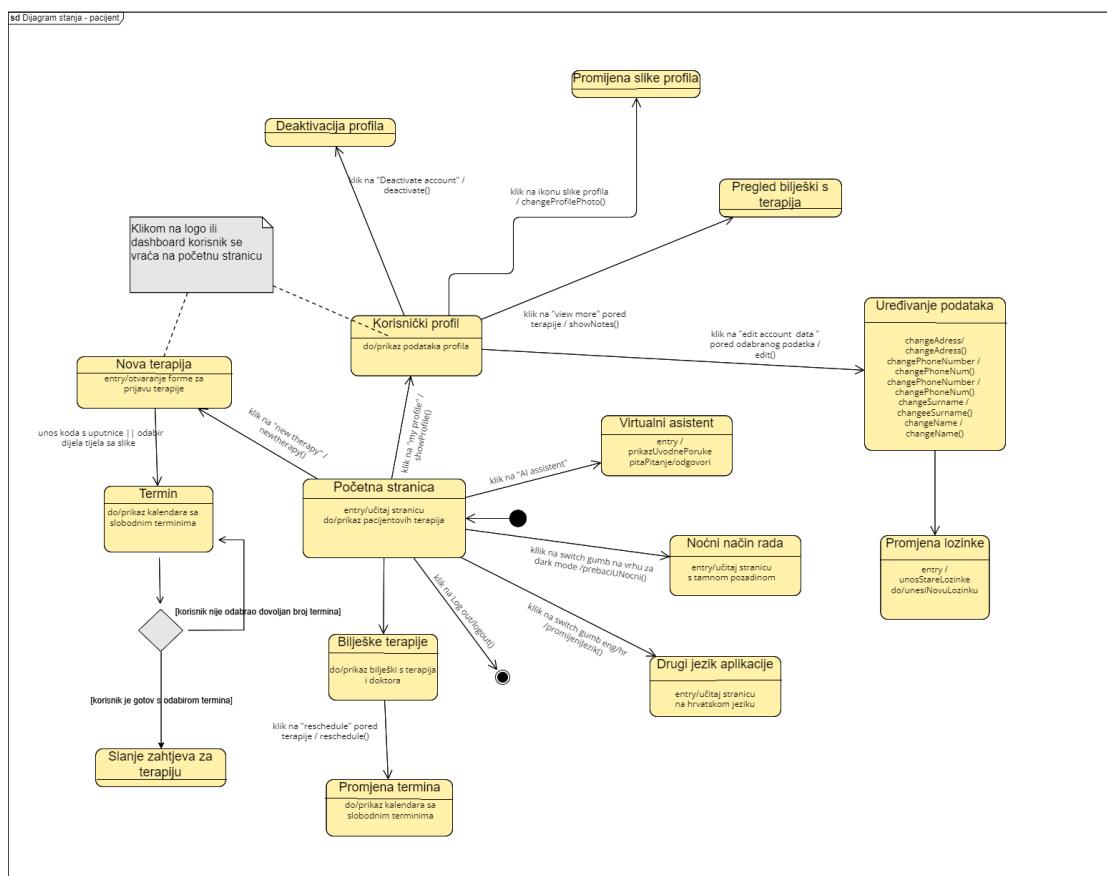
Dijagram stanja za registriranog i prijavljenog korisnika(pacijenta) prikazan je na Slici 4.4. Početno stanje označava da je korisnik već uspješno prijavljen u sustav te se preusmjerava na dashboard s rasporedom terapija. Ovdje korisnik može vidjeti trenutnu terapiju koja mu je dodijeljena.

U gornjem desnom kutu nalaze se gumbići za pristup korisničkom profilu pacijenta, kreiranje nove terapije ili odjavu. Klikom na korisnički profil, korisnik može pregledati osobne podatke i bilješke liječnika o terapijama. Također, korisnik ima mogućnost ažuriranja svojih osobnih podataka, uključujući promjenu broja mobitela ili postavljanje nove profilne slike. Odabirom opcije "New Therapy", korisnik može unijeti kod ili odabrati dio tijela sa skice za koji želi terapiju. Nakon toga, prikazuje se kalendar s dostupnim terminima za odabranu terapiju, gdje korisnik bira jedan ili više termina. Slanjem zahtjeva s MBO-om, šifrom terapije i odabranim datumom, korisnik podnosi zahtjev za novom terapijom.

U gornjem desnom kutu smješteni su gumbi koji omogućavaju prijelaz u tamni način rada te promjenu jezika između engleskog i hrvatskog, čime se postiže dvojezičnost u našoj aplikaciji.

Na dnu stranice smješten je virtualni asistent. MedBot je zadužen za pružanje općih informacija o terapijama, dok BayBot odgovara na pitanja korisnika vezana uz korištenje aplikacije.

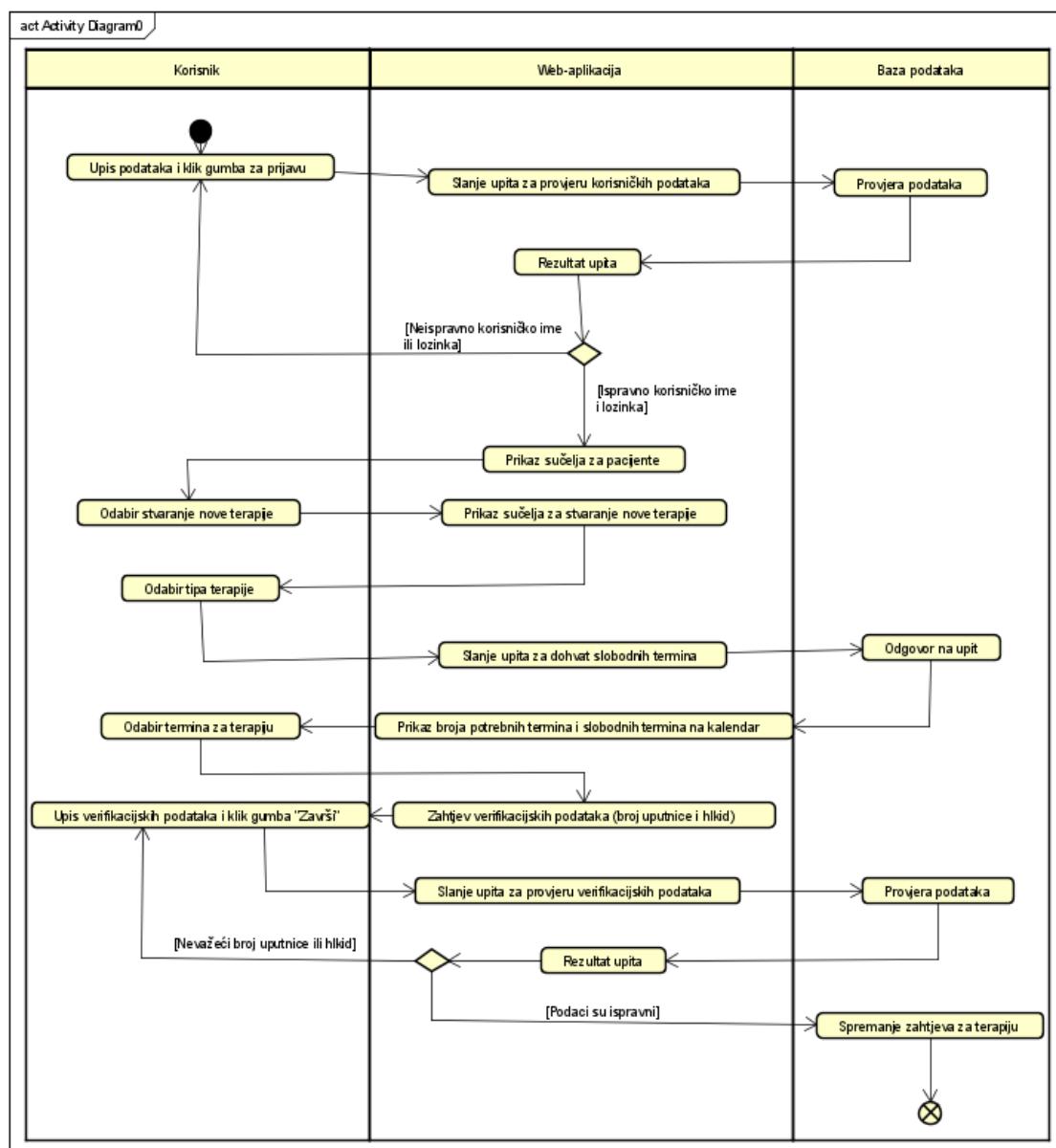
Korisnik također ima opciju odjave, što vodi u završno stanje. U svakom trenutku korisnik može i deaktivirati račun.



Slika 4.4: Dijagram stanja - pacijent

## 4.4 Dijagram aktivnosti

Na dijagramu aktivnosti 4.5 prikazan je proces kreiranja nove terapije. Korisnik (pacijent) se prijavljuje u sustav. Bira opciju za kreiranje nove terapije, unosi kod terapije ili bira iz dostupnih opcija. Odabire termine na kalendaru, unosi broj uputnice i identifikacijski broj liječnika. Nakon unosa svih podataka, korisnik klikne na gumb "Završi" čime se zahtjev šalje administratoru na odobrenje.

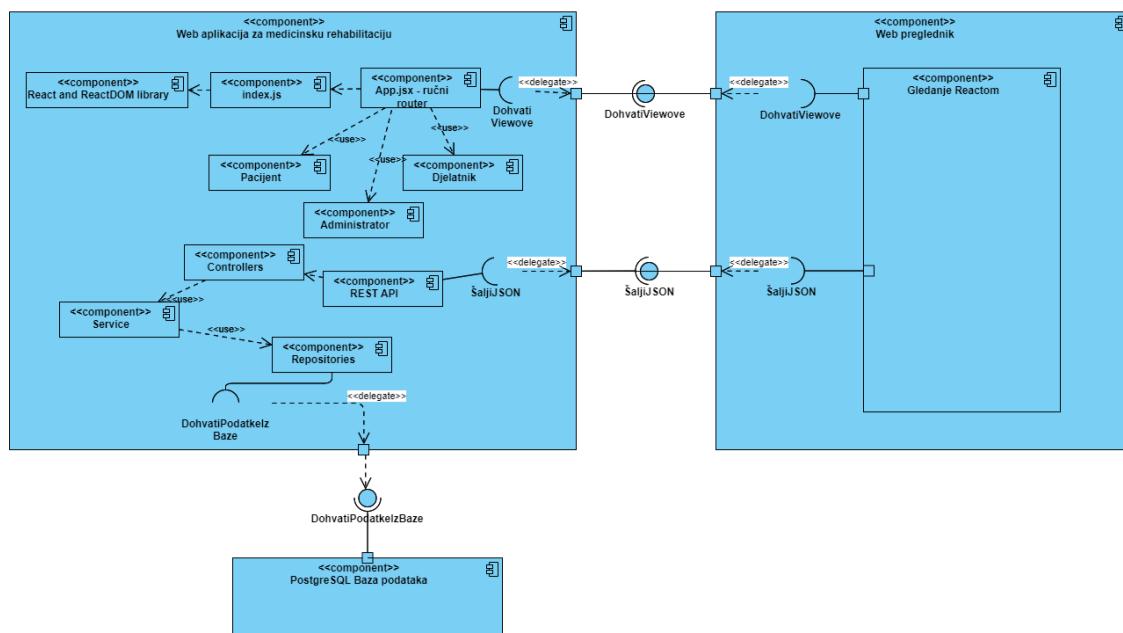


Slika 4.5: Dijagram aktivnosti

## 4.5 Dijagram komponenti

dio 2. revizije

Dijagram komponenti prikazuje od kojih komponenti je sustav sastavljen, kako te komponente međusobno surađuju i ovise jedna o drugoj. Iz sustava se informacije dobivaju na dva načina: upitima *frontendu* i *backendu*. *Frontend* dio (HTML, CSS, JS) dohvata se preko datoteke App.jsx koja vodi brigu o tome što će se prikazati. App.jsx ovisan je o bibliotekama Reacta. *Frontend* u sebi sadrži segmente koji mogu poslati podatke *backendu* na obradu. Tu *backend* prima zahtjeve s određenim podatcima, analizira ih kroz *Controllere*, Servise i Repozitorije (zadnji komunicira s PostgreSQL bazom podataka). Na taj način korisnik dobiva sučelja i informacije koje se njega tiču.



Slika 4.6: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

### KORIŠTENE TEHNOLOGIJE

Prilikom izrade web aplikacija za upravljanje rehabilitacijom koristili smo se raznim tehnologijama kako bismo pružili korisnicima učinkovit i interaktivan doživljaj. Backend aplikacije razvili smo koristeći Spring Boot, Java bazirani framework, koji omogućava brz i jednostavan razvoj server-side logike. Java, kao programski jezik, doprinosi robustnosti i skalabilnosti našeg backend sustava.

Za dinamičke i interaktivne elemente na korisničkom sučelju, koristili smo JavaScript, dok je React, popularni JavaScript framework, omogućio izgradnju efikasnog korisničkog sučelja. React, koristeći koncept komponenti, pojednostavljuje organizaciju i održavanje koda, pridonoseći poboljšanju korisničkog iskustva i olakšavajući upravljanje stanjem naše aplikacije.

Podaci o pacijentima, terapijama i terminima pohranjeni su u PostgreSQL bazi podataka koja pruža pouzdanu podršku. PostgreSQL, kao snažan objektno-relacijski sustav upravljanja bazama podataka, omogućava nam efikasno upravljanje informacija uz podršku za kompleksne upite i transakcije. Osim toga, HTML i CSS koriste se za strukturiranje sadržaja web stranice i stilizaciju, stvarajući tako funkcionalno i atraktivno korisničko sučelje.

Dokumentacija projekta oblikovana je pomoću LaTeX sustava za pripremu dokumenta, pružajući precizno formatiranje i organizaciju. Za organizaciju sastanaka i suradnju koristili smo Markdown format, pružajući jednostavan i čitljiv način za pisanje i dijeljenje informacija.

U procesu zajedničkog rada, pisanja koda i praćenja promjena koristili smo Git. Omogućavao je stvaranje, pregledavanje i spajanje promjena koda, čime se postizavao učinkovit timski rad. Sve ove tehnologije integrirane su u našem projektu kako bismo osigurali da interakcija između bolesnika, djelatnika zdravstvene ustanove i administratora bude glatka, učinkovita i prilagođena specifičnim ulogama i funkcionalnim zahtjevima svakog dionika.

Python je bio ključan u razvoju našeg AI chat bota, iskorištavajući njegove spo-

sobnosti u strojnom učenju i obradi prirodnog jezika za stvaranje interaktivnog i inteligentnog sučelja za korisnike. Python smo još koristili u kombinaciji sa Selenium WebDriverom za automatizaciju testiranja korisničkog sučelja, omogućavajući simulaciju stvarnih interakcija korisnika i osiguravajući pouzdanost naše web aplikacije.

## KORIŠTENI ALATI

U procesu razvoja naše aplikacije koristili smo raznovrsne alate kako bismo unaprijedili različite aspekte projekta. Alat Heroku, poznat po svojoj cloud platformi, omogućio nam je jednostavno deployanje, skaliranje i učinkovito upravljanje web aplikacijama. Korištenjem Heroku-a, značajno smo pojednostavili proces implementacije i održavanja naše aplikacije.

Za potrebe pisanja, testiranja i debugiranja koda koristili smo Visual Studio Code (VSCode), integrirano razvojno okruženje (IDE). VSCode je pružio alate koji su doprinijeli efikasnosti našeg tima tijekom razvoja aplikacije. IntelliJ IDEA, kao razvojno okruženje specifično za Java programski jezik, optimizirao je rad na backend dijelu naše aplikacije.

GitHub, platforma za upravljanje verzijama koda i suradnju timova, poslužila nam je za praćenje promjena, upravljanje zadacima te implementaciju pull requestova, unapređujući suradnju tima. Za brzu i jednostavnu komunikaciju te video razgovore koristili smo Discord, pružajući središnje mjesto za razmjenu informacija među članovima tima.

Notion, kao alat za organizaciju zadataka, vođenje bilješki sastanaka i općenito upravljanje projektom, korišten je kako bi pridonio boljoj organizaciji i praćenju napretka. Overleaf, online platforma za suradničko pisanje dokumenata u LaTeX formatu, olakšala je stvaranje i uređivanje dokumentacije našeg projekta. Visual Paradigm pruža alate za izradu različitih dijagrama, a mi smo ga koristili za izradu dijagrama obrasca uporabe, sekvencijskih dijagrama, dijagrama stanja, aktivnosti i komponenata potrebnih za dokumentaciju.

Figma, alat za dizajn, poslužio nam je za suradnju u izradi vizualnih planova naše web aplikacije. Kroz Figma-u definirali smo izgled frontend dijela naše aplikacije. Adobe Premiere Pro i Audition koristili smo za video i audio produkciju, stvarajući marketinške materijale i prezentacije vezane uz našu aplikaciju.

Naša interakcija s ChatGPT-om bila je višestruko korisna, pružajući podršku u različitim aspektima, uključujući punjenje baze podataka te pružanje informacija i savjeta u tijeku razvoja projekta.

Napomena: Prilikom klika u tekstu na nazine korištenih alata i tehnologija otvara se internet poveznica na kojoj se može saznati više informacija o njima. Unatoč tome ovdje su navedene internet poveznice na kojima je moguće saznati više: <https://www.spring.io/projects/spring-boot>, <https://www.oracle.com/java>, <https://www.deve> US/docs/Web/JavaScript, <https://www.reactjs.org>, <https://www.postgresql.org>, <https://www.devel> US/docs/Web/HTML, <https://www.developer.mozilla.org/en-US/docs/Web/CSS>, <https://www.overleaf.com/learn/latex>, <https://www.markdownguide.org>, <https://git-scm.com>, <https://www.heroku.com>, <https://www.visualstudio.com>, <https://www.jetbrains.com/ide>, <https://github.com>, <https://discord.com>, <https://www.notion.so>, <https://www.overleaf.com>, <https://www.visual-paradigm.com>, <https://www.figma.com>, <https://www.adobe.com/products/pr>, <https://www.adobe.com/products/audition.html>, <https://openai.com/gpt>

## 5.2 Ispitivanje programskog rješenja

Ispitivanje komponenti provedeno je na razredima TherapyTypeService i TherapyService u okruženju com.medbay. Slijede rezultati pojedinih ispitnih slučajeva:

### Rezultati ispitivanja komponenti za TherapyTypeService

#### 1. Test: getTherapyType\_ReturnsListOfTherapyTypes

- *Opis:* Provjera vraćanja liste tipova terapija.
- *Rezultat:* Test uspješan. Vraćena lista tipova terapija.

```
1 @Test
2 void getTherapyType_ReturnsListOfTherapyTypes() {
3     List<TherapyType> expectedTherapyTypes = Arrays.asList(
4         buildTherapyType("1"), buildTherapyType("2"));
5     when(therapyTypeRepository.findAll()).thenReturn(
6         expectedTherapyTypes);
7     ResponseEntity<List<TherapyType>> response = therapyTypeService.
8         getTherapyType();
9     assertEquals(HttpStatus.OK, response.getStatusCode());
10    assertEquals(expectedTherapyTypes, response.getBody());
11 }
```

#### 2. Test: getTherapyType\_WhenNoTherapyTypes\_ReturnsEmptyList

- *Opis:* Provjera vraćanja prazne liste kada nema tipova terapija.
- *Rezultat:* Test uspješan. Vraćena prazna lista.

```
1 @Test
2 void getTherapyType_WhenNoTherapyTypes_ReturnsEmptyList() {
3     when(therapyTypeRepository.findAll()).thenReturn(Collections.
4         emptyList());
5     ResponseEntity<List<TherapyType>> response = therapyTypeService.
6         getTherapyType();
7     assertEquals(HttpStatus.OK, response.getStatusCode());
8     assertTrue(Objects.requireNonNull(response.getBody()).isEmpty());
9 }
```

#### 3. Test: getTherapyType\_WhenRepositoryThrowsException\_ReturnsErrorResponse

- *Opis:* Provjera odziva na iznimku iz repozitorija.

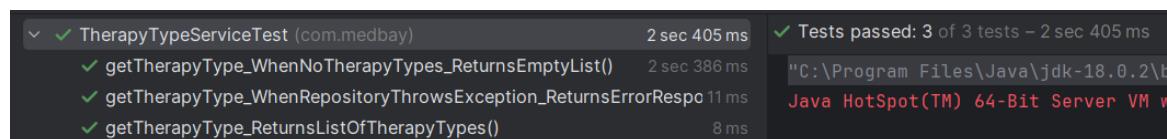
- *Rezultat:* Test uspješan. Izazvana iznimka.

```

1 @Test
2 void getTherapyType_WhenRepositoryThrowsException_ReturnsErrorResponse()
3 {
4     when(therapyTypeRepository.findAll()).thenThrow(new
5         RuntimeException("Database error"));
6     assertThrows(RuntimeException.class, () -> therapyTypeService.
7         getTherapyType());
8 }

```

Svi ispitni slučajevi su prošli uspješno, što ukazuje na pouzdanost i stabilnost implementiranih funkcionalnosti u razredu TherapyTypeService.



Slika 5.1: Rezultati testova za TherapyTypeService

### Rezultati ispitivanja komponenti za TherapyService

#### 1. Test: getTherapies>ReturnsListOfTherapies

- *Opis:* Provjera vraćanja liste terapija.
- *Rezultat:* Test uspješan. Vraćena lista terapija.

```

1 @Test
2 void getTherapies_ReturnsListOfTherapies() {
3     List<Therapy> expectedTherapies = Arrays.asList(buildTherapy("1")
4         , buildTherapy("2"));
5     when(therapyRepository.findAll()).thenReturn(expectedTherapies);
6     ResponseEntity<List<Therapy>> response = therapyService.
7         getTherapies();
8     assertEquals(HttpStatus.OK, response.getStatusCode());
9     assertEquals(expectedTherapies, response.getBody());
10 }

```

#### 2. Test: getTherapies\_WhenNoTherapies\_ReturnsEmptyList

- *Opis:* Provjera vraćanja prazne liste kada nema terapija.

- *Rezultat:* Test uspješan. Vraćena prazna lista.

```
1 @Test
2 void getTherapies_WhenNoTherapies_ReturnsEmptyList() {
3     when(therapyRepository.findAll()).thenReturn(Collections.
4         emptyList());
5     ResponseEntity<List<Therapy>> response = therapyService.
6         getTherapies();
7     assertEquals(HttpStatus.OK, response.getStatusCode());
8     assertTrue(Objects.requireNonNull(response.getBody()).isEmpty());
9 }
```

### 3. Test: deleteTherapy\_WhenTherapyExists\_DeletesTherapy

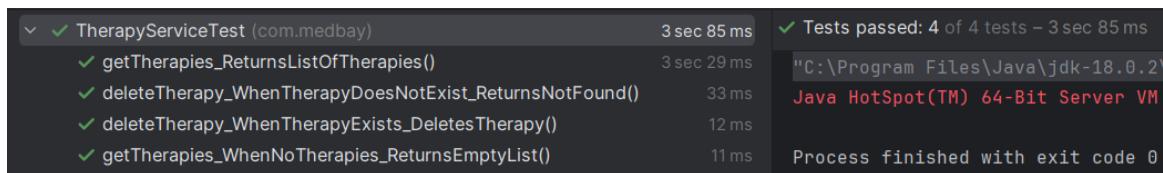
- *Opis:* Provjera brisanja postojeće terapije.
- *Rezultat:* Test uspješan. Terapija izbrisana.

```
1 @Test
2 void deleteTherapy_WhenTherapyExists_DeletesTherapy() {
3     Long therapyId = 1L;
4     when(therapyRepository.existsById(therapyId)).thenReturn(true);
5     ResponseEntity<Void> response = therapyService.deleteTherapy(
6         therapyId);
7     assertEquals(HttpStatus.OK, response.getStatusCode());
8     verify(therapyRepository).deleteById(therapyId);
9 }
```

### 4. Test: deleteTherapy\_WhenTherapyDoesNotExist\_ReturnsNotFound

- *Opis:* Provjera odziva kada terapija ne postoji.
- *Rezultat:* Test uspješan. Vraćen status "Nije pronađeno".

```
1 @Test
2 void deleteTherapy_WhenTherapyDoesNotExist_ReturnsNotFound() {
3     Long therapyId = 1L;
4     when(therapyRepository.existsById(therapyId)).thenReturn(false);
5     ResponseEntity<Void> response = therapyService.deleteTherapy(
6         therapyId);
7     assertEquals(HttpStatus.NOT_FOUND, response.getStatusCode());
8     verify(therapyRepository, never()).deleteById(therapyId);
9 }
```



Slika 5.2: Rezultati testova za TherapyService

Svi testovi su uspješno prošli, što ukazuje na pouzdanost i ispravnost implementacije u razredu `TherapyService`.

Napomena: Prilikom klika na naziv razreda korištenih za testiranje otvara se internet poveznica na kojoj se može vidjeti izvorni kod testova.

### 5.2.1 Ispitivanje sustava

Svi testovi osim registracije i zaboravljenih lozinki izvršeni su uz pomoć Selenium WebDrivera. Ispitivanje se radilo po obrascima uporabe kako bi se provjerile funkcionalnosti sustava. Prikazivanje ispitivanja UC1, UC3, UC4, UC5, UC6, UC i UC13 (uključujući UC7).

## Ispitni sučaj 1: Registracija

1. Otvaranje stranice za registraciju.
  2. Unos obveznih podataka: ime, prezime, email, datum rođenja, adresa, broj telefona, MBO, lozinka.
  3. Riješavanje Google reCAPTCHA.
  4. Klik na gumb za registraciju.

### Očekivani izlaz:

1. Prikaz poruke o uspjehu slanja zahtjeva za registraciju.

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

## Ispitni sučaj 2: Prijava na stranicu Ulaz:

1. Otvaranje početne stranice.
  2. Unos e-maila i lozinka.
  3. Klik na gumb za prijavu.

**Očekivani izlaz:**

1. Uspješna prijava i preusmjeravanje na početnu stranicu za pacijente.

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

**Ispitni slučaj 3: Promjena termina terapije**

1. Klik na prvi termin terapije.
2. Klik na opciju "Odgodi termin".
3. Promjena datuma termina iza posljednjeg termina terapije.
4. Potvrda odabira i klik na gumb za potvrdu odgode.
5. Klik na poslijednji termin terapije.
6. Klik na opciju "Odgodi termin".
7. Promjena datuma termina na orginalni datum.
8. Potvrda odabira i klik na gumb za potvrdu odgode.

**Očekivani izlaz:**

- 1.a Uspješna promjena datuma termina.
- 1.b Informacije o terminu promijenjene i odgovaraju novom datumu.
- 2.a Uspješna promjena datuma termina.
- 2.b Informacije o terminu promijenjene i odgovaraju novom datumu.

**Rezultat:** Očekivani rezultat [2.b] nije zadovoljen jer se redni broj termina prikazuje kao posljednji umjesto kao prvi, iako je u kalendaru na točnoj poziciji. Ostala očekivanja su zadovoljena. **Aplikacija nije prošla test.**

**Ispitni slučaj 4: Stvaranje novog zahtjeva za terapiju**

1. Klik na opciju "Nova terapija".
2. Odabir vrste terapije i unos potrebnih datuma.
3. Unos verifikacijski podataka terapije sa uputnice.
4. Potvrda odabira i klik na gumb za završetak.

5. Prikaz poruke o uspješnom zahtjevu za novu terapiju.

**Očekivani izlaz:**

1. Uspješno stvaranje terapije i prikaz potvrde.
2. Dodavanje termina u raspored korisnika.

**Rezultat:** Sva očekivanja su zadovoljena. **Aplikacija je prošla test.**

**Ispitni slučaj 5: Zaboravljena lozinka**

1. Otvaranje početne stranice.
2. Klik na "Zaboravljena lozinka".
3. Upis e-mail adrese za oporavak lozinke.
4. Klik na poveznicu koja je došla e-mailom.
5. Upis i potvrda nove lozinke.
6. Klik na gumb "Reset password".
7. Spremanje promjena.

**Očekivani izlaz:**

1. Sustav šalje elektronsku poštu s obrascem za promjenu lozinke.
2. Nakon ispunjavanja obrasca uspješno promijenjena lozinka.

**Rezultat:** Sva očekivanja su zadovoljena. **Aplikacija je prošla test.**

**Ispitni slučaj 6: Promjena bilješke o terminu pacijenta**

1. Odabir termina kojem želimo promijeniti bilješku o terminu.
2. Klik na "Uredi".
3. Promjena bilješke o terminu.
4. Spremanje promjena.

**Očekivani izlaz:**

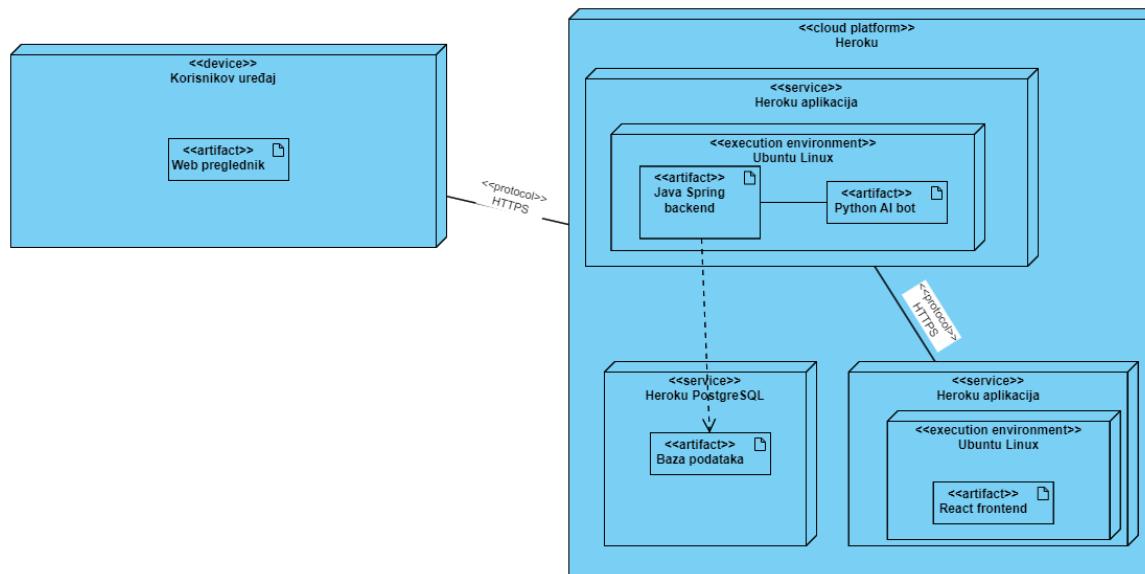
1. Ažuriranje bilješke o terminu.

**Rezultat:** Očekivani rezultat je ostvaren. **Aplikacija je prošla test.**

## 5.3 Dijagram razmještaja

### dio 2. revizije

Dijagram razmještaja opisuje topologiju sustava, fizičku arhitekturu i razmještaj programskih sustava te kako te cjeline komuniciraju. Na dijagramu imamo dva glavna čvora. Prvo imamo uređaj korisnika (PC, mobitel) preko čijeg web preglednika korisnik putem HTTPS-a dohvaća informacije iz drugog čvora, mjesto gdje je pohranjen kod aplikacije, baza podataka i gdje je upogonjena aplikacija. Riječ je o Herokuu, platformi za pogonjenje aplikacija na oblaku. Heroku funkcioniра pomoću *dynoa*, fleksibilnih i izoliranih kontejnera koji sadrže Linux. Naša web aplikacija koristi tri *dynoa*. Na prvoj se nalazi frontend dio aplikacije koji je napisan u okviru React. Druga komponenta je kontejner na kojem se sadrži *backend* dio aplikacije. Tu je najbitnija Spring aplikacija koja reagira na zahtjeve *frontenda* i komunicira s bazom podataka, a prisutan je i kratki kod koji omogućuje rad AI bota na našoj aplikaciji. Tu se još nalazi i zadnji *dyno*, Herokuova PostgreSQL usluga gdje se nalazi baza podataka koju aplikacija koristi. Kontejneri za *frontend* i *backend* komuniciraju HTTPS protokolom, a *backend* i baza podataka s protokolima koje osmišlja i definira sam Heroku.



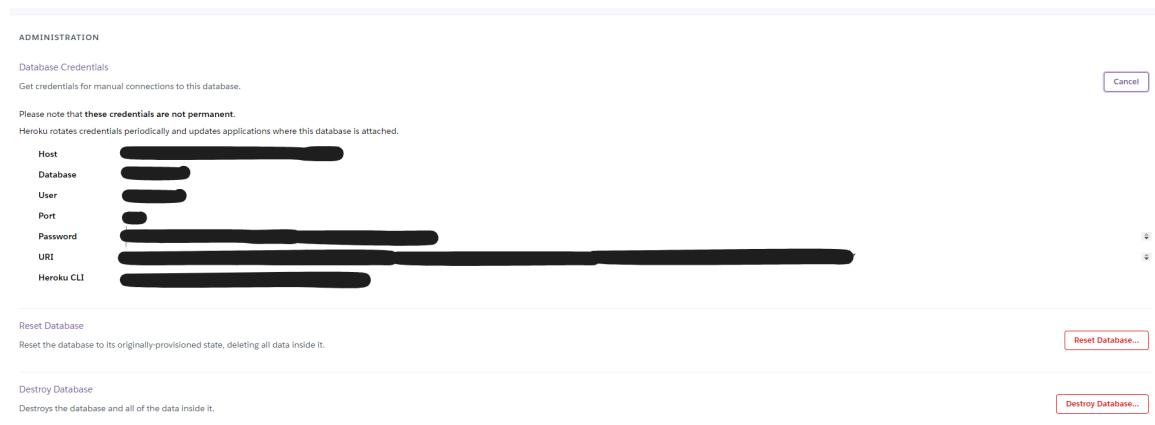
Slika 5.3: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

U postavljanju i puštanju u pogon naše aplikacije ključnu ulogu igra Heroku CLI (Command Line Interface), moćan naredbeni alat koji pojednostavljuje interakciju s Heroku platformom izravno iz terminala. Instalirali smo Heroku CLI na lokalnom računalu kako bismo iskoristili njegove funkcionalnosti za upravljanje aplikacijama na Heroku platformi.

Naša integrirana strategija razvoja i puštanja u pogon koristi niz ključnih značajki Heroku platforme:

1. Automatsko puštanje u pogon s GitHub-a: - GitHub repozitorij povezan je s Heroku, omogućujući jednostavno automatsko puštanje u pogon naše aplikacije. - Svaka promjena na glavnoj grani GitHuba automatski pokreće puštanje u pogon na Heroku platformi.
2. Konfiguracijske Datoteke na Heroku: - Heroku nam omogućava preciznu konfiguraciju putem posebnih datoteka koje definiraju okruženje za našu aplikaciju. - Detaljno smo definirali parametre poput verzija jezika (Java, Python) i aktivnih profila za Spring.
3. Korištenje Financijske Potpore od Heroku:
  - Naša organizacija prima financijsku potporu od Heroku platforme, pružajući nam sredstva za korištenje resursa.
4. Baza Podataka na Heroku: - Integrirali smo Heroku Postgres "add-on" za potrebe baze podataka. - Heroku Postgres pruža siguran pristup bazama podataka putem pridruženih akreditiva.



Slika 5.4: Akreditivi za našu aplikaciju

5. Dinamičko Okruženje s dinamičkim kontejnerima: - Korištenjem dinamičkih

kontejnera na Heroku stvaramo virtualno okruženje koje omogućuje izvršavanje i hostanje naše aplikacije. - Ime domene "medbay.life" registrirano je putem "Namecheap"-a, dok smo HTTPS certifikat dobili kroz Cloudflare za sigurnu komunikaciju.

6. Praćenje i evidentiranje: - Heroku pruža napredne alate za praćenje performansi, uključujući detaljne evidencije za svaki zahtjev poslan aplikaciji. - Metrike i analize omogućuju nam praćenje ponašanja aplikacije u stvarnom vremenu.



The screenshot shows the Heroku Application Logs interface. It displays a list of log entries with timestamp, source, method, path, status, and duration. The logs include requests for static files like 'green\_shape-f4d129ed.png' and '73c5-49e1-8009-77b703e9848d', as well as requests for services like 'dnservice-1'. The logs also show database queries and system events. A checkbox for 'Autoscroll with output' is checked at the bottom left.

Slika 5.5: Aplikacijsko evidentiranje



Slika 5.6: Metrika aplikacija

7. Paketi za izgradnju: - Integrirali smo različite Pakete za izgradnju prilagođene jezicima koje koristimo (npr., Python, Java). - Automatski konfiguriraju okolinu i podržavaju različite jezične specifikacije.

Ovaj sveobuhvatan pristup omogućava nam efikasan ciklus razvoja i puštanja u pogon aplikacije. Kombinacijom GitHuba, Heroku platforme, i alata poput He-

roku CLI-a, postižemo stabilan i automatski proces implementacije koji podržava dinamično okruženje naše aplikacije.

**Općenito o načinu korištenja Heroku CLI (Command Line Interface):**

1. Instalacija Heroku CLI: Heroku CLI smo instalirali na lokalnom računalu, omogućujući nam jednostavan pristup naredbama i funkcionalnostima koje pruža. Za instalaciju, slijedili smo upute dostupne na [službenoj Heroku stranici za instalaciju CLI-a](<https://devcenter.heroku.com/articles/heroku-cli>).

2. Povezivanje s Heroku Računom: Pomoću Heroku CLI-a uspostavili smo vezu s našim Heroku računom, omogućavajući autentikaciju i autorizaciju za izvršavanje različitih operacija. Ova faza je ključna za pristup resursima i upravljanje aplikacijom.

3. Korištenje CLI-a za puštanje u pogon aplikacije: Heroku CLI smo aktivno koristili za inicijalizaciju puštanje u pogon naših aplikacija. Naredbe poput 'git push heroku main' omogućuju nam jednostavno i automatsko puštanje u pogon na Heroku platformu.

4. Upotreba CLI-a za Konfiguraciju Aplikacije: Heroku CLI pruža alate za konfiguraciju različitih aspekata aplikacije. Kroz naredbe poput 'heroku config:set' postavljali smo i mijenjali okružne varijable te druge konfiguracijske postavke prema potrebama naše aplikacije.

5. Praćenje evidencija i Performansi: Za praćenje evidencija i performansi aplikacije koristili smo Heroku CLI. Naredbe poput 'heroku logs' omogućuju nam pregled događaja u stvarnom vremenu, pružajući važne informacije o ponašanju aplikacije.

6. Dodatne Naredbe za Upravljanje Resursima: Heroku CLI pruža širok spektar dodatnih naredbi za upravljanje resursima. Skaliranje aplikacije, upravljanje "add-on"-ima i druge operacije mogu se jednostavno izvršiti putem ovog sučelja.

Heroku CLI je postao ključan alat u našem procesu razvoja i deploya. Omogućuje nam jednostavnu interakciju s Heroku platformom direktno iz terminala, čime poboljšavamo produktivnost i imamo veću kontrolu nad našim aplikacijama.

## 6. Zaključak i budući rad

### *dio 2. revizije*

*U ovom poglavlju potrebno je napisati osvrt na vrijeme izrade projektnog zadatka, koji su tehnički izazovi prepoznati, jesu li riješeni ili kako bi mogli biti riješeni, koja su znanja stečena pri izradi projekta, koja bi znanja bila posebno potrebna za brže i kvalitetnije ostvarenje projekta i koje bi bile perspektive za nastavak rada u projektnoj grupi.*

*Potrebno je točno popisati funkcionalnosti koje nisu implementirane u ostvarenoj aplikaciji.*

Projektni zadatak je bio usmjeren na razvoj web aplikacije za praćenje i upravljanje procesima medicinske rehabilitacije, obuhvaćajući vremenski period od 23. listopada 2023. do 17. siječnja 2024. godine. Redovito smo održavali tjedne sastanke i konzultacije tijekom cijelog trajanja projekta kako bismo pratili napredak i uskladivali aktivnosti. Na početku svakog sastanka članovi tima iznosili su svoja postignuća iz prethodnog tjedna, dok smo na kraju sastanka planirali zadatke za sljedeći tjedan.

Prvu fazu projekta obilježila je podjela tima na frontend i backend skupine. U prvom tjednu posvetili smo se učenju Spring Boot-a i React-a, ovisno o odabranoj skupini. Studenti koji nisu bili upoznati s Git-om ili GitHub-om prošli su odgovarajuće vježbe. Nakon stjecanja temeljnog znanja, fokusirali smo se na razvoj ideja i mogućnosti aplikacije te definiranje funkcionalnih zahtjeva.

Suradnja unutar tima bila je ključna u oblikovanju funkcionalnosti aplikacije. Veći dio tima posvetili smo i dokumentaciji kako bismo je temeljito pripremili. Prvi izazov dogodio se sredinom studenog kada smo naišli na problem na GitHubu vezan uz nepravilno usklađivanje sekundarnih grana s glavnom granom. Sekundarne grane su već imale dodatne funkcionalnosti, ali su bile u zaostatku u odnosu na glavnu granu. Nakon uspješnog rješavanja problema, unaprijedili smo svoje vještine u radu s Git-om.

Frontend tim koristio je alat Figma za oblikovanje izgleda aplikacije, dok se backend tim usredotočio na implementaciju registracije, prijave i povezivanje s

bazom podataka. Frontend tim uspješno je izvršio implementaciju svog dijela koristeći React, pridonoseći ciljevima za prvu predaju projekta. Zadaci vezani uz dokumentaciju bili su raspoređeni prema kraju faze, gdje smo izradili dijagrame poput obrazaca uporabe, sekvencijskih dijagrama, modela baze podataka i dijagrama razreda. Iako smo postigli cilj završetka prije predviđenog roka, posljednji tjedan posvetili smo ispravcima i unapređenjima.

Za početak druge faze projekta, održan je redoviti sastanak na kojem smo raspodijelili zadatke i postavili okvirni cilj završetka do 10. siječnja, iako je predviđeni rok za zadnju predaju bio dva tjedna kasnije. Odlučili smo odmah krenuti s radom, a frontend tim koristeći alat Figma, oblikovao je dizajn preostalog dijela aplikacije.

Nastavili smo s tjednim sastancima za praćenje napretka. Backend i frontend timovi održavali su dodatne sastanke nekoliko puta tjedno kako bi riješili moguće probleme i međusobno si pomogli u implementaciji. Backend tim predvodio je Ian, dijeleći zadatke unutar tima s obzirom na svoje iskustvo s Spring Bootom. Nakon implementacije, svaki član bi poslao "pull request", a Ian bi pregledao i davao povratne informacije o potrebnim ispravkama.

Tijekom izrade aplikacije, stalno smo dobivali nove ideje, a svaki put kad smo mislili da se približavamo kraju, odlučili smo implementirati dodatne mogućnosti. Implementacija ovih dodatnih značajki zahtjevala je značajan dodatni rad i na backendu i na frontendu. Frontend tim bio je izložen dodatnom zadatku osmišljavanja novog dizajna kako bi se integrirale sve nove ideje. Ovaj dinamičan proces kreativnog razvoja doprinio je bogatstvu funkcionalnosti i poboljšanju korisničkog iskustva u konačnom proizvodu. Uključili smo opciju tamnog načina rada za bolju preglednost i implementirali virtualnog asistenta koji pomaže korisnicima u navigaciji aplikacijom i pruža informacije o terapijama. Sve funkcionalnosti definirane na početku, uspješno smo implementirali. Kako smo se približavali završetku, pojedini članovi backend tima vratili su se dokumentaciji i dovršili dijagrame stanja, aktivnosti, komponenata te razmještaja.

Projekt smo uspješno završili usklađujući naše ideje, rješavajući izazove uz pomoć dobre komunikacije. Svjesno smo se suočavali s izazovima, pratili napredak i prilagođavali se novim idejama tijekom cijelog procesa razvoja. Uloženi trud i međusobna podrška doveli su nas do izvrsne aplikacije koja odražava naše zajedničke napore i posvećenost projektu.

# Popis literature

## ***Kontinuirano osvježavanje***

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

1. Programsко 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	Sučelje aplikacije Medbridge GO koja je dostupna na pametnim telefonima . . . . .	10
2.2	Sučelje aplikacije BokDoc koja je dostupna na pametnim telefonima	10
2.3	Sučelje aplikacije Practo, dostupno na pametnim telefonima . . . . .	11
2.4	Sučelje aplikacije SimplyBook.me: lijevo - korisnički pogled, desno - admin pogled . . . . .	12
3.1	Dijagram obrasca uporabe, funkcionalnost bolesnika . . . . .	24
3.2	Dijagram obrasca uporabe, funkcionalnost djelatnika . . . . .	25
3.3	Dijagram obrasca uporabe, funkcionalnost administratora . . . . .	26
3.4	Sekvencijski dijagram za UC5 . . . . .	28
3.5	Sekvencijski dijagram za UC6 . . . . .	29
3.6	Sekvencijski dijagram za UC13 . . . . .	30
3.7	Sekvencijski dijagram za UC16 . . . . .	31
4.1	MVC arhitektura . . . . .	40
4.2	Dijagram baze podataka . . . . .	47
4.3	Dijagram razreda . . . . .	49
4.4	Dijagram stanja - pacijent . . . . .	52
4.5	Dijagram aktivnosti . . . . .	53
4.6	Dijagram komponenti . . . . .	54
5.1	Rezultati testova za TherapyTypeService . . . . .	59
5.2	Rezultati testova za TherapyService . . . . .	61
5.3	Dijagram razmještaja . . . . .	64
5.4	Akreditivi za našu aplikaciju . . . . .	65
5.5	Aplikacijsko evidentiranje . . . . .	66
5.6	Metrika aplikacija . . . . .	66

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### Kontinuirano osvježavanje

U ovom dijelu potrebno je redovito osvježavati dnevnik sastanja prema predlošku.

#### 1. Sastanak: Tjedni Sastanak

- Datum: 23. listopada 2023.
- Prisustvovali: Tea, Ian, Nikola, Ivan, Niko, Lovro, Karlo
- Teme sastanka:
  - Prethodno Čitanje i Ažuriranja Tima:
    - \* Ian: Napisao flow, pomagao s SpringBootom, komunikacija s asistentom
    - \* Tea: Odradila Git i Crash Courseve
    - \* Ivan: Istraživanje Gita, učenje Spring Boota
    - \* Lovro: Pregled video materijala, početak React tečaja
    - \* Niko: Rad s Gitom i konzolom, istraživanje Notiona
    - \* Nikola: Pregled sadržaja, planiranje dodatnih projekata
    - \* Karlo: Postavljanje Notiona, slušanje Spring predavanja, diskusija o flowu
  - Dnevni Red:
    - \* Provjere gut osjećaja članova tima
    - \* Predavanja CROZ - prisustvovali Nikola i Karlo
    - \* Diskusija o procesu terapije i mogućnosti sudjelovanja djelatnika
  - Zaključci:
    - \* Pacijenti biraju između preddefiniranih procedura terapije
    - \* Specijalizacija djelatnika za različite vrste procedura
    - \* Uloge liječnika i administrativnih djelatnika
    - \* Proces verifikacije podataka liječnika i registracije pacijenata

- \* Implementacija sučelja za administracijske funkcije
  - **Trajanje:** 60 minuta

## 2. Konzultacije s Asistentom

- Datum: 25. listopada 2023.
- Prisustvovali: Tea, Ian, Nikola, Ivan, Niko, Lovro (Karlo nije prisustvao)
- Teme konzultacija:
  - **Razrada i Pitanja:** [Detalji o postavljenim pitanjima i razgovoru]
- Zaključci sastanka:
  - Git repozitorij treba biti javan s dodatnim suradnicima
  - Vođenje dnevnika sastanaka na Gitu s osnovnim informacijama
  - Korištenje lažne (fejk) baze podataka za provjeru liječnika
  - Šifriranje lozinki u bazi podataka
  - "Brisanje" računa u bazi podataka putem statusa
  - Referenciranje prethodnih terapija pomoću vanjskog ključa
  - Status liječnika odnosi se na aktivnost (npr. u penziji)
  - Izmisliti opremu, uređaje i terapije s realnim brojem resursa
  - Ciljevi do kraja prvog ciklusa: Deployment stranice, ostvarenje logina, minimalni prikaz podataka iz baze
- **Trajanje:** 60 minuta

## 3. Diskusija: Use-Cases

- Datum: 28. listopada 2023.
- Prisustvovali: Ian, Karlo (Niko, Nikola, Ivan, Lovro, Tea nisu prisustvovali)
- Teme diskusije:
  - **Real-Time Kalendar za Odabir Termina:**
    - \* Pacijenti biraju termine s obzirom na dostupnost djelatnika i opreme
  - **Oprema i Prostor:**
    - \* Oprema i prostor tretirani su kao jedinstveni resurs (npr. bazen, elektroterapija)

- **Pravila za Odabir i Izmjenu Termina:**
  - \* Mogućnost izmjene termina s minimalnim vremenskim ograničenjem
  - \* Minimalni razmak između terapija definiran varijablama procedure
- **Proces Odobravanja Terapije:**
  - \* Provjera zdravstvenog osiguranja i ispravnosti odabrane terapije
- **Dilema oko Verifikacije:**
  - \* Diskusija o potrebi i načinu provođenja verifikacije tijekom registracije i prijave
- **Zaključci sastanka:**
  - Detaljno definiranje procesa odabira i izmjene termina
  - Razrada mehanizma odobravanja terapije
  - Rasprava o verifikaciji identiteta i osiguranja u kontekstu registracije i prijave
- **Trajanje:** 60 minuta

#### 4. Sastanak: Tjedni Sastanak

- Datum: 29. listopada 2023.
- Prisustvovali: Tea, Ian, Nikola, Ivan, Niko, Lovro, Karlo
- Teme sastanka:
  - **Provjeravanje i Ažuriranja Tima:**
    - \* Tea: Analiza i proučavanje Ianovog koda, istraživanje LaTeX-a
    - \* Nikola: Analiza Ianovog koda
    - \* Niko: Prati Scrimbu za dodatno učenje
    - \* Ivan: Proučava Ianov kod, instalira LaTeX editor
    - \* Lovro: Napreduje s Scrimbu tečajem
    - \* Ian: Objašnjava SpringBoot timu, priprema backend, uvoz podataka u bazu, planira deployment
    - \* Karlo: Ažuriranje GitHuba i Notiona, razrada flowa i use caseova
  - **Dnevni Red i Ključna Pitanja:**
    - \* Stanje s dokumentacijom
    - \* Verifikacijski procesi pri registraciji i prijavi
    - \* Mogućnosti i funkcionalnosti pacijentovog dashboarda
    - \* Postojanje i funkcije liječničkog accounta

- \* Mogućnosti i ovlasti djelatnika
- **Zaključci:**
  - \* Registracija pacijenata uz MBO i OIB
  - \* Definicija resursa i termina u kontekstu terapije
  - \* Funkcije i preglednost rasporeda za djelatnike
  - \* Detalji vezani uz profile pacijenata i djelatnika
  - \* Specifičnosti liječničkog accounta i pacijentovih mogućnosti na dashboardu
- **Trajanje:** 1 sat i 45 minuta

## 5. Tjedni Sastanak

- Datum: 5. studenog 2023.
- Prisustvovali: Tea, Ian, Ivan, Niko, Lovro, Karlo (Nikola odsutan zbog operacije)
- Teme sastanka i Ažuriranja Članova Tima:
  - Tea: Uredila use-caseove u LaTeXu, postavila ih na GitHub, istraživanje baza podataka s Iantom
  - Niko: Nastavak s tečajem Scrimbe
  - Ivan: Napisao prvih 15 use-caseova
  - Lovro: Završio Scrimbu, proučavao dokumentaciju
  - Ian: Prošao use-caseove s Nikolom, razvio backend za registraciju i login, priprema za deployment
  - Karlo: Organizacija Git-a, pisanje u Notionu, pregled use-caseova
- Razmatrana Pitanja:
  - Analiza use-caseova, posebno registracija i prijava bolesnika
  - Pitanje upotrebe OIB-a i MBO-a u registraciji
  - Use-case za prijavu u sustav i rezervaciju termina rehabilitacije
- Detaljni Zaključci:
  - Terapija veže uz sebe resurse: kombinacija slobodnog termina, resursa i djelatnika
  - Djelatnici imaju opciju pisanja bilješki nakon termina
  - Pacijenti imaju mogućnost pregleda i uređivanja svojih termina, s uvjetom promjene termina više od 48 sati unaprijed
  - Administrativna provjera osiguranja bolesnika prema MBO-u pri

prijava na rehabilitaciju

- Otkazivanje termina uključuje dvosmjernu komunikaciju među svim ulogama
  - Definiranje minimalnog razmaka između sesija i maksimalnog trajanja terapije
  - Djelatnici u kalendaru vide samo svoje termine, s mogućnošću pretrage po pacijentu
  - Nema privilegiranog djelatnika; svi djelatnici su ravnopravni u sustavu
  - Specifični ciljevi za sljedeći ciklus: login, registracija, osnovna komunikacija s bazom
  - Preispitivanje i optimizacija određenih use-caseova
- **Trajanje:** 2 sata
  - **Todo:** Specifični zadaci dodijeljeni članovima tima

## 6. Sastanak za UI/UX

- Datum: 12. studenog 2023.
- Teme Sastanka: UX/UI Dizajn i Dijagrami
- Prisustvovali: Niko, Karlo (Ivan, Nikola, Ivan, Lovro, Tea nisu prisustvovali)
- Teme sastanka i Diskusije:
  - **Razvoj Korisničkog Sučelja:**
    - \* Razvijen UI za ključne stranice aplikacije, fokus na intuitivnosti i pristupačnosti
    - \* Detaljna diskusija o dizajnu i korisničkom iskustvu, uzimajući u obzir ciljanu publiku
  - **Dijagrami i Use-Caseovi:**
    - \* Diskutirani dijagrami s naglaskom na poboljšanje razumijevanja procesa unutar aplikacije
    - \* Niko zabilježio promjene i ažuriranja potrebna za use-caseove i dijagrame
- Zaključci sastanka:
  - Uspješno razvijeno korisničko sučelje za početne stranice, usklađenost s općim ciljevima projekta

- Razrađeni dijagrami i vizualizacije procesa unutar aplikacije, povećanje transparentnosti i razumijevanja
- Ažuriranje i optimizacija use-caseova i dijagrama, osiguravajući njihovu usklađenost s trenutnim razvojem projekta
- **Trajanje:** 3 sata
- **Nastavak rada:** Fokus na daljnji razvoj UI/UX-a i finalizaciju dijagrama

## 7. Tjedni Sastanak

- Datum: 13. studenog 2023.
- Prisustvovali: Tea, Nikola, Ivan, Niko, Lovro, Karlo (Ian nije prisustvao zbog bolesti)
- Teme sastanka i Ažuriranja Članova Tima:
  - Tea: Uključivanje baze podataka u dokumentaciju, opis arhitekture, pisanje natuknica o projektu, priprema poglavlja 4.1 s opisom tablica
  - Nikola: Pregled dijagrama i osiguravanje njihove točnosti
  - Niko: Rad na use-case i sekvencijskim dijagramima, ispravci dokumentacije, suradnja s Karlom i Lovrom na Figmi, završetak Scrimba tečaja
  - Ivan: Suradnja na use-case dijagramima, doprinos sekvencijskim dijagramima
  - Lovro: Istraživanje Figme, razvoj login stranice, planiranje registrovanih funkcionalnosti
  - Karlo: Dizajniranje UI-ja u Figmi, ažuriranje Notiona, studij dokumentacije
- Diskutirane Točke Dnevnog Reda:
  - Problemi s commitanjem na Git: Hitno rješenje za nepravilno korištenje grana
  - Planiranje razvoja frontenda i deploja projekta
  - Važnost pravilnog ažuriranja Notiona i dokumentacije
  - Integracija dijagrama u službenu dokumentaciju
- Detaljni Zaključci i Upute:
  - Obavezno vođenje dnevnika promjena za svaki važan commit
  - Detaljni individualni zadaci i upiti za nadolazeći sastanak

- Fokus na rješavanju konflikata, dopuni dokumentacije, i razvoju UI-ja
- Organizacija i koordinacija timskih obveza i rokova
- **Trajanje:** 1 sat
- **Naredni koraci:** Specifični zadaci dodijeljeni članovima tima za kontinuirani napredak projekta

## Tablica aktivnosti

### *Kontinuirano osvježavanje*

*Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.*

	Karlo Vrančić	Ian Balen	Nikola Baretić	Tea Ćetojević-Tisaj	Lovro Dujjić	Niko Kaštelan	Ivan Kordić
Upravljanje projektom	18	4					
Opis projektnog zadatka	11	8					
Funkcionalni zahtjevi	7	7	2	2	4	2	5
Opis pojedinih obrazaca	8		5		3	8	8
Dijagram obrazaca						8	6
Sekvencijski dijagrami						5	5
Opis ostalih zahtjeva							
Arhitektura i dizajn sustava		9		3			
Baza podataka		8		7			
Dijagram razreda							
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati		4					
Ispitivanje programskog rješenja							
Dijagram razmještaja							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Karlo Vrančić	Ian Balen	Nikola Baretić	Tea Ćetojević-Tisaj	Lovro Dujić	Niko Kaštelan	Ivan Kordić
Upute za puštanje u pogon							
Dnevnik sastajanja	6						
Zaključak i budući rad							
Popis literature	1						
<i>Dodatne stavke kako ste podijelili izradu aplikacije</i>							
<i>izrada dizajna</i>	5				2	5	
<i>izrada baze podataka</i>		5	4				
<i>spajanje s bazom podataka</i>		2					
<i>back end</i>		10					
<i>front end</i>					15		
<i>učenje tehnologija React, SpringBoot i Git</i>	3		20	18	30	35	10

## Dijagrami pregleda promjena

### *dio 2. revizije*

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