

Programsko inženjerstvo

Ak. god. 2022./2023.

Fizikalna terapija

Dokumentacija, Rev. 2

Grupa: *PROGI_Tim_2022*

Voditelj: *Fran Ogrinšak*

Datum predaje: *13.1.2023.*

Nastavnik: *Miljenko Krhen*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	9
3.1 Funkcionalni zahtjevi	9
3.1.1 Obrasci uporabe	11
3.1.2 Sekvencijski dijagrami	20
3.2 Ostali zahtjevi	23
4 Arhitektura i dizajn sustava	24
4.1 Baza podataka	26
4.1.1 Opis tablica	26
4.1.2 Dijagram baze podataka	29
4.2 Dijagram razreda	30
4.3 Dijagram stanja	32
4.4 Dijagram aktivnosti	34
4.5 Dijagram komponenti	35
5 Implementacija i korisničko sučelje	37
5.1 Korištene tehnologije i alati	37
5.2 Ispitivanje programskog rješenja	38
5.2.1 Ispitivanje komponenti	38
5.2.2 Ispitivanje sustava	38
5.3 Dijagram razmještaja	44
5.4 Upute za puštanje u pogon	46
6 Zaključak i budući rad	50
Popis literature	52
Indeks slika i dijagrama	54

Dodatak: Prikaz aktivnosti grupe

55

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Dominik Đurinić, Fran Hruza	21.10.2022.
0.2	Napisani funkcionalni i nefunkcionalni zahtjevi	Fran Hruza, Dominik Đurinić	21.10.2022.
0.3	Dodan <i>Use Case</i> dijagram	Fran Hruza	26.10.2022.
0.4	Napisan opis projektnog zadatka	Dominik Đurinić	7.11.2022.
0.5	Dodani sekvencijski dijagrami	Fran Hruza	7.11.2022.
0.6	Dodan dijagram razreda	Fran Hruza	15.11.2022.
0.7	Arhitektura i baza podataka (opis i ER dijagram)	Dominik Đurinić	16.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Fran Hruza, Dominik Đurinić	18.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
------	-----------------------	--------	-------

2. Opis projektnog zadatka

Ideja projekta je izrada web aplikacije "Fizikalna terapija" za potrebe zdravstvene ustanove "Zdravljem do znanja".

Djelatnosti koje obavlja ova ustanova su fizikalna terapija i rehabilitacija bolesnika nakon lakših ili težih povreda. Ustanova broji ukupno 15 djelatnika, od kojih su 9 fizikalni terapeuti, 3 medicinske sestre, 2 administratora te voditelj ustanove. Broj djelatnika nije konačan već se može mijenjati.

Zahtjev zdravstvene ustanove je učinkovit informacijski sustav koji omogućuje izradu rasporeda bolesnika na terapije koje ustanova pruža, praćenje podataka o bolesnicima i njihovom napretku u liječenju. Također sustav bi trebao omogućiti praćenje rada svih djelatnika ustanove. Pristup navedenim mogućnostima imaju samo registrirani djelatnici ustanove. Registrirani korisnici usluga zdravstvene ustanove imaju mogućnost uvida u vlastiti termin te popis i opis usluga/tretmana koje dobivaju. Neregistrirani korisnici mogu vidjeti samo popis i opis svih usluga/tretmana koje ustanova pruža.

Razlikujemo četiri vrste korisnika informacijskog sustava:

- vlasnik sustava (administrator)
- voditelj ustanove
- medicinski djelatnici (fizioterapeuti i medicinske sestre)
- bolesnici (korisnici usluga)

Vlasnik sustava (administrator) upisuje sve podatke o uslugama/tretmanima te registrira medicinske djelatnike, tj. unosi podatke o medicinskim djelatnicima. Vlasnik sustava može, ali i ne mora biti zaposlenik ustanove.

Voditelj ustanove upisuje sve podatke o uslugama/tretmanima, registrira medicinske djelatnike te ima mogućnost pregleda aktivnosti u ustanovi: pregled aktivnosti djelatnika, broj pacijenata po djelatniku i broj pacijenata po usluzi/tretmanu te

zauzeće pojedinih uređaja. Sve aktivnosti može pregledavati u proizvoljnom vremenskom intervalu.

Medicinski djelatnik može biti fizioterapeut ili medicinska sestra. Nakon što je registriran od strane administratora ili voditelja ustanove, može se prijaviti u sustav i pristupiti funkcionalnostima sustava. Funkcionalnosti su: izrada rasporeda pacijenata, registracija pacijenata, evidencija dolaska pacijenta i obavljenosti terapije te evidencija stanja pacijenta na kraju terapije. Također može kao i voditelj ustanove pristupiti pregledu broja pacijenata po usluzi/tretmanu, zauzeću pojedinih uređaja te broju pacijenata, ali samo za one pacijente za koje je registriran kao voditelj za određeni dan.

Bolesnici se pri prvom dolasku u ustanovu registriraju od strane medicinskog djelatnika. Nakon registracije, bolesnik ima mogućnost prijave u sustav i pregleda vlastitog rasporeda tretmana, njihovog opisa te potvrde o dolaznosti na terapiju.

Sustav omogućava istovremeni rad svih korisnika sustava te podržava unos hrvatskih dijakritičkih znakova.

Za registraciju medicinskog djelatnika potrebni su sljedeći podaci:

- ime
- prezime
- stručna sprema
- specijalizacija (ukoliko postoji)
- adresa elektroničke pošte
- broj telefona
- nedostupnost medicinskog djelatnika (bolovanje, godišnji odmor i dr.)

Za registraciju bolesnika potrebni su sljedeći podaci:

- ime
- prezime
- broj telefona
- adresa elektroničke pošte (ukoliko postoji)
- dijagnoza (iz liste dijagnoza bolesti prema MKB)
- MBO (matični broj osiguranika)

- dodatno osiguranje (da/ne)
- ime i prezime liječnika koji je uputio bolesnika na terapiju (iz liste liječnika)
- lista odabranih tretmana/usluga (na koje je bolesnik upućen)

Pri prvom dolasku bolesnika u ustanovu, medicinski djelatnik koji ga zaprima izvršava registraciju bolesnika. Ukoliko je bolesnik već prije bio na nekom tretmanu, odnosno ima izvršenu registraciju, onda nije potrebna ponovna registracija već se podaci dohvaćaju iz baze podataka. Nakon registracije, isti medicinski djelatnik koji je zaprimio bolesnika izrađuje personalizirani raspored tretmana za bolesnika. Bitno je naglasiti sljedećih nekoliko uvjeta vezanih uz mogućnost izrade rasporeda.

1. Radno vrijeme ustanove je 8 sati dnevno u razdoblju od 9:00 do 17:00 sati, od ponedjeljka do petka. Unutar tog razdoblja se vrše medicinske usluge pacijentima. Subotom, nedjeljom, blagdanima i praznicima je ustanova zatvorena.
2. Ustanova posjeduje ograničeni broj resursa (uređaja i prostora) te je trajanje pojedine terapije vremenski unaprijed određeno. Svi uređaji mogu raditi paralelno i u isto vrijeme.
3. Svaki bolesnik ima terapiju u trajanju od 10 dana i to uvijek u isto vrijeme u danu. Ne postoji mogućnost nadoknade u slučaju ne dolaska bolesnika na predviđeni termin.

Potrebno je uskladiti izradu rasporeda bolesnika sa navedenim uvjetima, tj. poslovanjem zdravstvene ustanove.

Prilikom svakodnevnog dolaska bolesnika na terapiju, medicinski djelatnik evidentira bolesnikov dolazak. Sustav automatski registrira po danima medicinskog djelatnika koji je prihvatio pojedinog bolesnika.

Nakon izvršenog desetog dana terapije medicinski djelatnik obavlja razgovor s bolesnikom te u sustav unosi stanje bolesnika nakon terapije (napredak, zadovoljstvo, aktualni problem, izjava bolesnika). Također postoji mogućnost unosa zapažanja medicinskog djelatnika u sustav. Nakon razgovora izrađuje se dokument koji sadrži sve podatke o bolesniku, tretmanu/usluzi, broju dolazaka, imenima djelatnika koji su zaprimili bolesnika po danima te podaci uneseni prilikom završnog razgovora. Stvara se dokument u pdf formatu, koji se onda ispisuje te ga

bolesnik potpisuje. Dokument se pohranjuje u arhivu ustanove te se po želji bolesnika može poslati elektronička verzija dokumenta na adresu elektroničke pošte bolesnika. To je ujedno i završna faza tretmana/usluge.

Opasnost od Covid zaraze je još uvijek prisutna u svakodnevnom životu, stoga kao podsjetnik na zarazu sustav preuzima podatke s javno dostupnog servisa o broju zaraženih u Republici Hrvatskoj na taj dan i trend broja zaraza u određenom vremenskom intervalu. Svi navedeni podaci će se prikazati medicinskim djelatnicima prilikom prijave u sustav.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Voditelj ustanove
2. Bolesnici
3. Medicinski djelatnici
 - (a) Fizioterapeuti
 - (b) Medicinske sestre
4. Administrator (vlasnik sustava)
5. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) pregledati popis i opis usluga koje ustanova pruža
2. Registrirani bolesnik (inicijator) može:
 - (a) vidjeti svoj raspored te potvrdu dolaska
 - (b) pregledati popis i opis svojih usluga/tretmana
3. Administrator (inicijator) može:
 - (a) upisati sve podatke o uslugama/tretmanima
 - (b) registrirati podatke o medicinskim djelatnicima
4. Voditelj ustanove (inicijator) može:
 - (a) upisati sve podatke o uslugama/tretmanima
 - (b) registrirati podatke o medicinskim djelatnicima
 - (c) pregledati aktivnosti u ustanovi
 - i. pregled aktivnosti djelatnika
 - ii. broj pacijenata po djelatniku

- iii. broj pacijenata po tretmanu/usluzi
- iv. zauzeće pojedinih uređaja

5. Medicinski djelatnik (inicijator) može:

- (a) upisati podatke o pacijentu
- (b) raditi raspored pacijenata
- (c) evidentirati dolazak pacijenata
- (d) evidentirati stanje pacijenta nakon terapije
 - i. napredak pacijenta
 - ii. zadovoljstvo pacijenta uslugom
 - iii. izjava bolesnika
 - iv. vlastiti komentar i zapažanja
- (e) pregledati aktivnosti u ustanovi (kao voditelj tog dana)
 - i. broj pacijenata za koje je voditelj tog dana
 - ii. broj pacijenata po tretmanu/usluzi
 - iii. zauzeće pojedinih uređaja

6. Baza podataka (sudionik):

- (a) pohranjuje podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje podatke o uslugama koje ustanova nudi
- (c) pohranjuje raspored terapija

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija medicinskog djelatnika

- **Glavni sudionik:** Administrator, voditelj ustanove
- **Cilj:** Registracija i unos podataka medicinskog djelatnika
- **Sudionici:** Baza podataka, medicinski djelatnik
- **Preduvjet:** Prijavljenost u sustav
- **Opis osnovnog tijeka:**
 1. Administrator/Voditelj odabire opciju za registraciju medicinskog djelatnika
 2. Administrator/Voditelj upisuje potrebne podatke o djelatniku
 3. Administrator/Voditelj dobiva obavijest o uspješnoj registraciji djelatnika
- **Opis mogućih odstupanja:**
 - 2.a Unos podataka u nedozvoljenom formatu
 1. Sustav obavještava administratora/voditelja o krivom formatu podataka
 2. Administrator/Voditelj mijenja podatke u ispravan format ili odustaje od registracije

UC2 - Unos tretmana/usluga u ponudu

- **Glavni sudionik:** Administrator, voditelj ustanove
- **Cilj:** Ažurirati popis/opis tretmana/usluga koje ustanova nudi
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava voditelja/administratora u sustav
- **Opis osnovnog tijeka:**
 1. Administrator/Voditelj odabire opciju za ažuriranje ponude
 2. Administrator/Voditelj odabire želi li stvoriti novi zapis ili odabire tretman/uslugu koju želi ažurirati
 3. Administrator/Voditelj upisuje naziv i opis usluge

UC3 - Pregled aktivnosti u ustanovi

- **Glavni sudionik:** Voditelj ustanove
- **Cilj:** Pregled aktivnosti djelatnika, broj bolesnika po djelatniku te broj pacijenata po tretmanu/usluzi
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava voditelja u sustav
- **Opis osnovnog tijeka:**
 1. Voditelj odabire funkcionalnost pregleda aktivnosti u ustanovi
 2. Voditelj odabire u kojem vremenskom intervalu želi informacije
 3. Sustav prikazuje tražene informacije

UC4 - Prijava medicinskog djelatnika

- **Glavni sudionik:** Medicinski djelatnik
- **Cilj:** Dobiti pristup funkcijama medicinskog djelatnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija medicinskog djelatnika
- **Opis osnovnog tijeka:**
 1. Medicinski djelatnik unosi svoj adresu elektroničke pošte i lozinku koju mu je dodijelio administrator/voditelj
 2. Potvrda o ispravnosti unesenih podataka
 3. Sustav prikazuje aktualne podatke o broju zaraženih Covidom-19 taj dan te trend broja zaraženih u određenom intervalu
 4. Pristup funkcijama medicinskog djelatnika
- **Opis mogućih odstupanja:**
 - 2.a Unos neispravne adrese elektroničke pošte/lozinke
 1. Sustav obavještava medicinskog djelatnika o neispravnom unosu podataka i vraća ga na stranicu za prijavu u sustav

UC5 - Registracija bolesnika

- **Glavni sudionik:** Medicinski djelatnik
- **Cilj:** Registracija i unos podataka bolesnika
- **Sudionici:** Baza podataka, Bolesnik
- **Preduvjet:** Prijava medicinskog djelatnika
- **Opis osnovnog tijeka:**
 1. Medicinski djelatnik odabire opciju za registraciju bolesnika

2. Medicinski djelatnik upisuje MBO bolesnika
3. Sustav provjerava postoji li zapis o bolesniku s unesenim MBO-om (bolesnik je prethodno odradio jednu ili više terapija) i automatski registrira bolesnika
4. Ako je bolesnik prvi put u ustanovi, medicinski djelatnik upisuje podatke

- **Opis mogućih odstupanja:**

2.a Unos podataka u nedozvoljenom formatu

1. Sustav obavještava medicinskog djelatnika o krivom formatu podataka
2. Medicinski djelatnik mijenja podatke u ispravan format ili odustaje od registracije

UC6 - Stvaranje rasporeda tretmana za bolesnika

- **Glavni sudionik:** Medicinski djelatnik
- **Cilj:** Stvoriti raspored tretmana/usluga za bolesnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija bolesnika
- **Opis osnovnog tijeka:**
 1. Medicinski djelatnik odabire bolesnika iz liste registriranih bolesnika
 2. Medicinski djelatnik unosi tretmane i njihov termin
 3. Sustav potvrđuje ispravnost podataka s obzirom na ograničenja

- **Opis mogućih odstupanja:**

2.a Unos podataka koji su u koliziji s rasporedima drugih bolesnika

1. Sustav obavještava medicinskog djelatnika o zauzetom terminu te ga vraća na stranicu za izradu rasporeda

UC7 - Prihvat i evidencija dolaska bolesnika

- **Glavni sudionik:** Medicinski djelatnik
- **Cilj:** Prihvatiti i evidentirati dolazak bolesnika na terapiju
- **Sudionici:** Baza podataka, bolesnik
- **Preduvjet:** Prijava medicinskog djelatnika
- **Opis osnovnog tijeka:**
 1. Medicinski djelatnik odabire funkcionalnost za evidenciju dolaska bolesnika
 2. Medicinski djelatnik unosi ime i prezime te MBO bolesnika
 3. Sustav potvrđuje ispravnost podataka

4. Sustav osvježava podatke u rasporedu bolesnika te podatke tko je pojedinog bolesnika taj dan prihvatio

- **Opis mogućih odstupanja:**

2.a Unos podataka koji su u koliziji s rasporedima drugih bolesnika

1. Sustav obavještava medicinskog djelatnika o neispravnosti podataka (bolesnik nije registriran ili nema termin taj dan)
2. Sustav medicinskog djelatnika vraća na stranicu za evidenciju dolaska bolesnika

UC8 - Evidencija kraja tretmana/usluge

- **Glavni sudionik:** Medicinski djelatnik

- **Cilj:** Stvoriti izvještaj iz informacija završnog razgovora

- **Sudionici:** Baza podataka, bolesnik

- **Preduvjet:** Prijava medicinskog djelatnika

- **Opis osnovnog tijeka:**

1. Medicinski djelatnik odabire bolesnika za kojeg želi napraviti izvješće
2. Medicinski djelatnik odabire funkcionalnost za izradu izvješća po završetku terapije
3. Medicinski djelatnik unosi podatke iz intervjua te opcionalno svoje komentare i zapažanja
4. Sustav stvara dokument pdf dokument
5. Sustav nudi djelatniku mogućnost slanja dokumenta na adresu elektroničke pošte bolesnika

- **Opis mogućih odstupanja:**

2.a Odabrani bolesnik nije završio terapiju

1. Sustav obavještava medicinskog djelatnika da nije moguće stvoriti izvještaj za odabranog bolesnika
2. Sustav medicinskog djelatnika vraća na popis bolesnika

3.a Neispravan format podataka

1. Sustav obavještava medicinskog djelatnika o neispravnom formatu podataka
2. Sustav medicinskog djelatnika vraća na stranicu za izradu izvješća

4.a Ne postoji zapis o adresi elektroničke pošte bolesnika

1. Sustav nudi medicinskom djelatniku da upiše adresu elektroničke pošte bolesnika
2. Medicinski djelatnik upisuje adresu koju je dobio od bolesnika

UC9 - Pregled aktivnosti/informacija o uređajima i terminima

- **Glavni sudionik:** Medicinski djelatnik/Voditelj
- **Cilj:** Omogućiti djelatniku prikaz broja bolesnika po određenoj usluzi/tretmanu i broja bolesnika za koje su oni registrirani kao voditelji za određeni dan.
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava medicinskog djelatnika
- **Opis osnovnog tijeka:**
 1. Medicinski djelatnik odabire opciju za pregled aktivnosti
 2. Sustav prikazuje broj bolesnika po određenoj usluzi/tretmanu, zauzeće pojedinih uređaja i broj bolesnika za koje je djelatnik voditelj taj dan

UC10 - Pregled usluga

- **Glavni sudionik:** Bolesnik
- **Cilj:** Pregledati popis i opis usluga ustanove
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prikazuje se popis svih usluga koje ustanova nudi

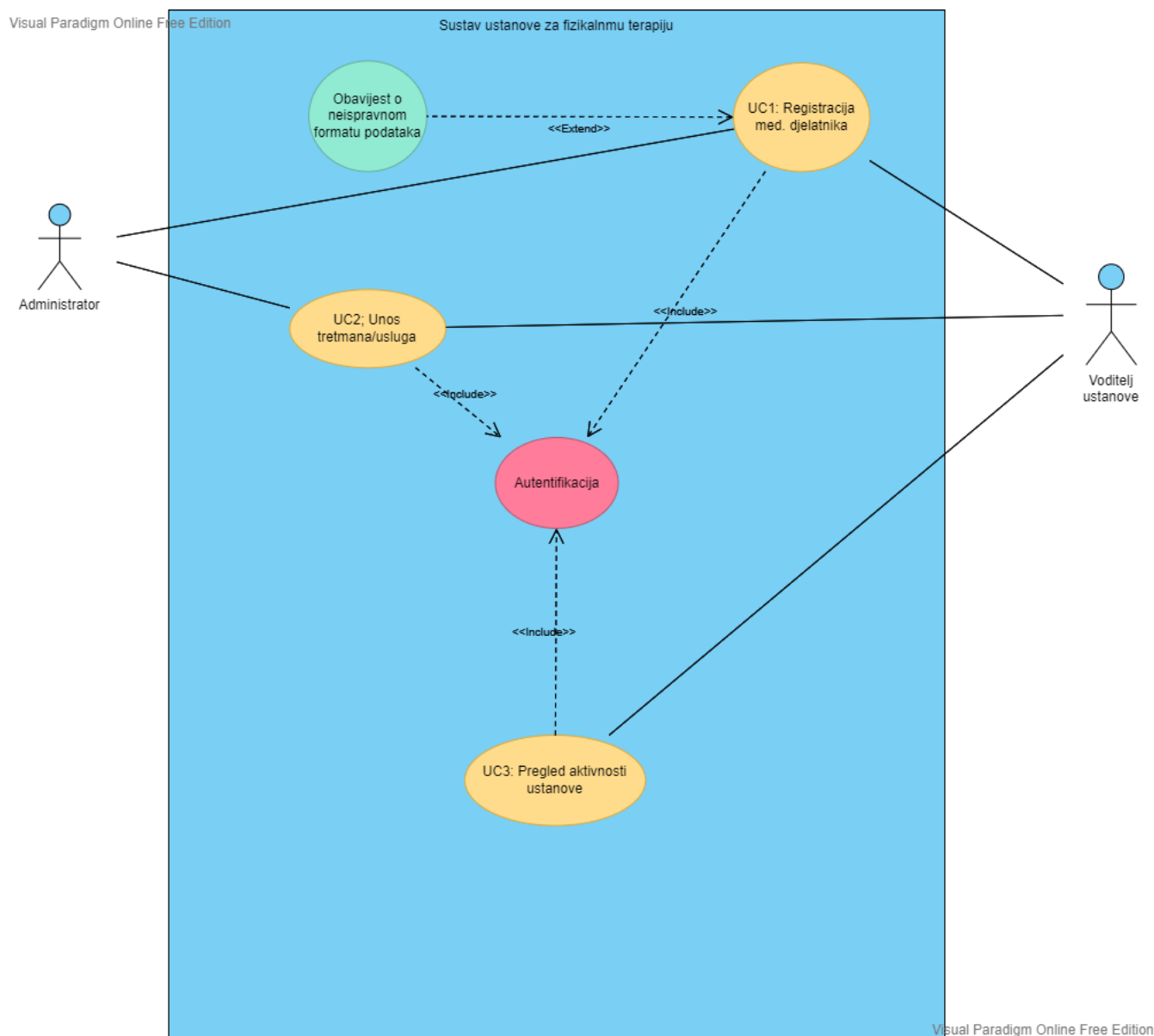
UC11 - Prijava bolesnika u sustav

- **Glavni sudionik:** Bolesnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija bolesnika
- **Opis osnovnog tijeka:**
 1. Bolesnik unosi svoj MBO i lozinku koju stvara medicinski djelatnik
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup funkcijama prijavljenog bolesnika
- **Opis mogućih odstupanja:**
 - 2.a Neispravan MBO/lozinka
 1. Sustav obavještava bolesnika o krivim podacima i vraća ga na stranicu za prijavu u sustav

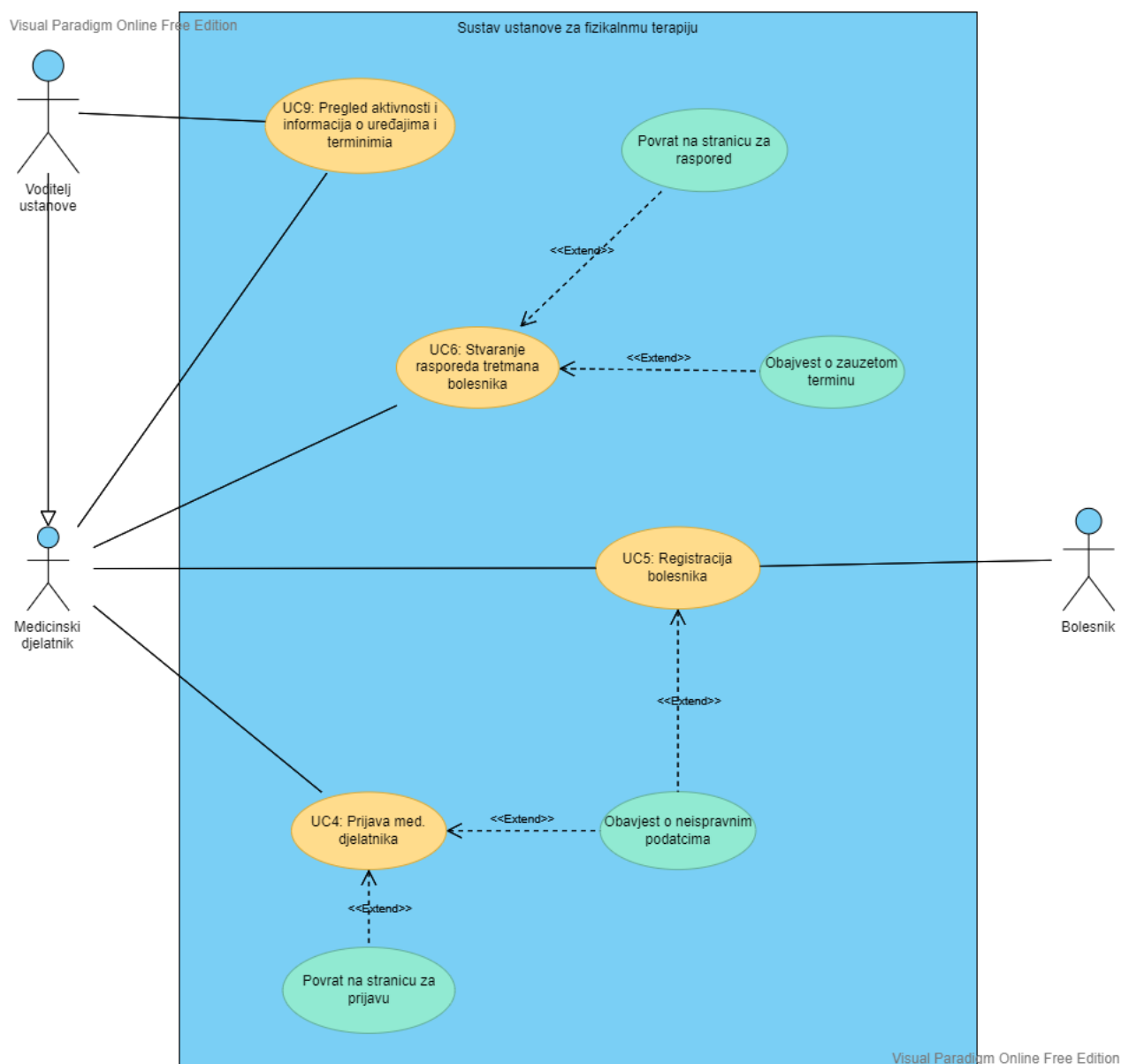
UC12 - Pregled termina tretmana i evidencije dolaznosti

- **Glavni sudionik:** Bolesnik
- **Cilj:** Pregledati preostale tretmane i evidentirane dolaske
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava bolesnika
- **Opis osnovnog tijeka:**
 1. Bolesnik odabire opciju za prikaz termina i dolaznosti
 2. Sustav bolesnika preusmjerava na stranicu sa popisom termina tretmana te evidentiranim dolaskom

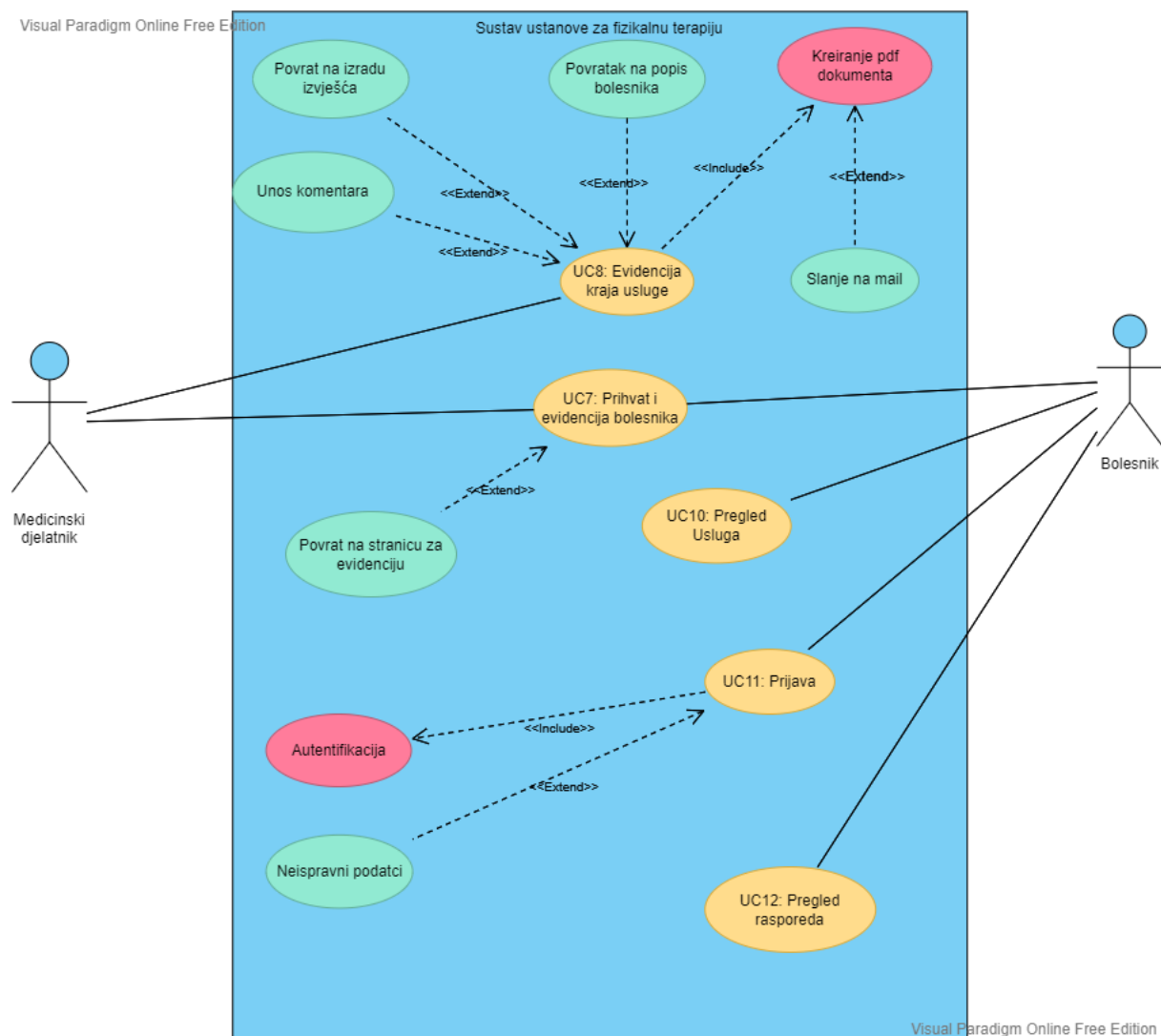
Dijagrami obrazaca uporabe



Slika 3.1: UML obrazaca korištenja UC1 do UC3



Slika 3.2: UML obrazaca korištenja UC4 do UC6 te UC9

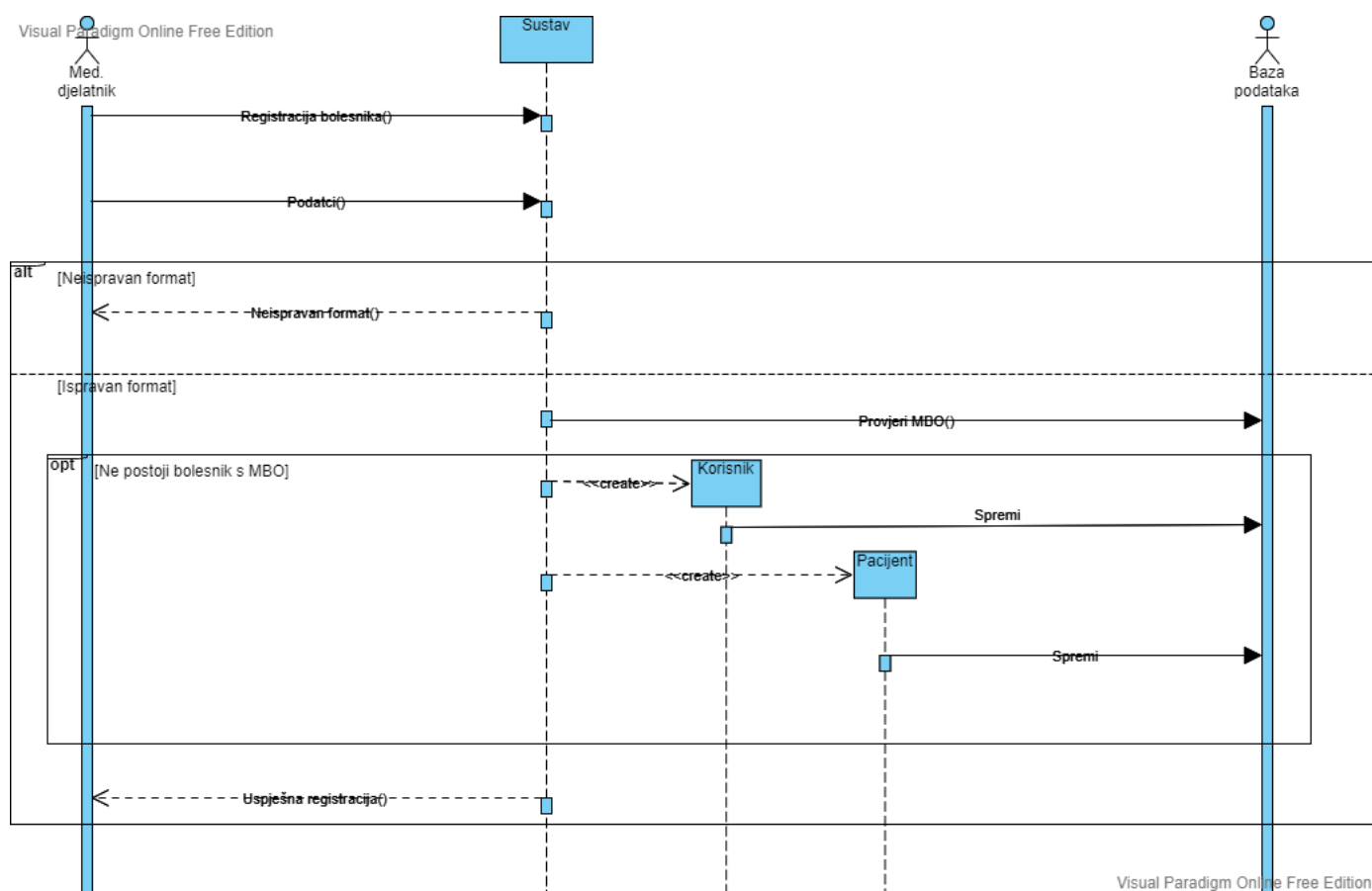


Slika 3.3: UML obrazaca korištenja UC7 do UC8 te od UC10 do UC12

3.1.2 Sekvencijski dijagrami

Opis sekvencijskog dijagrama UC5

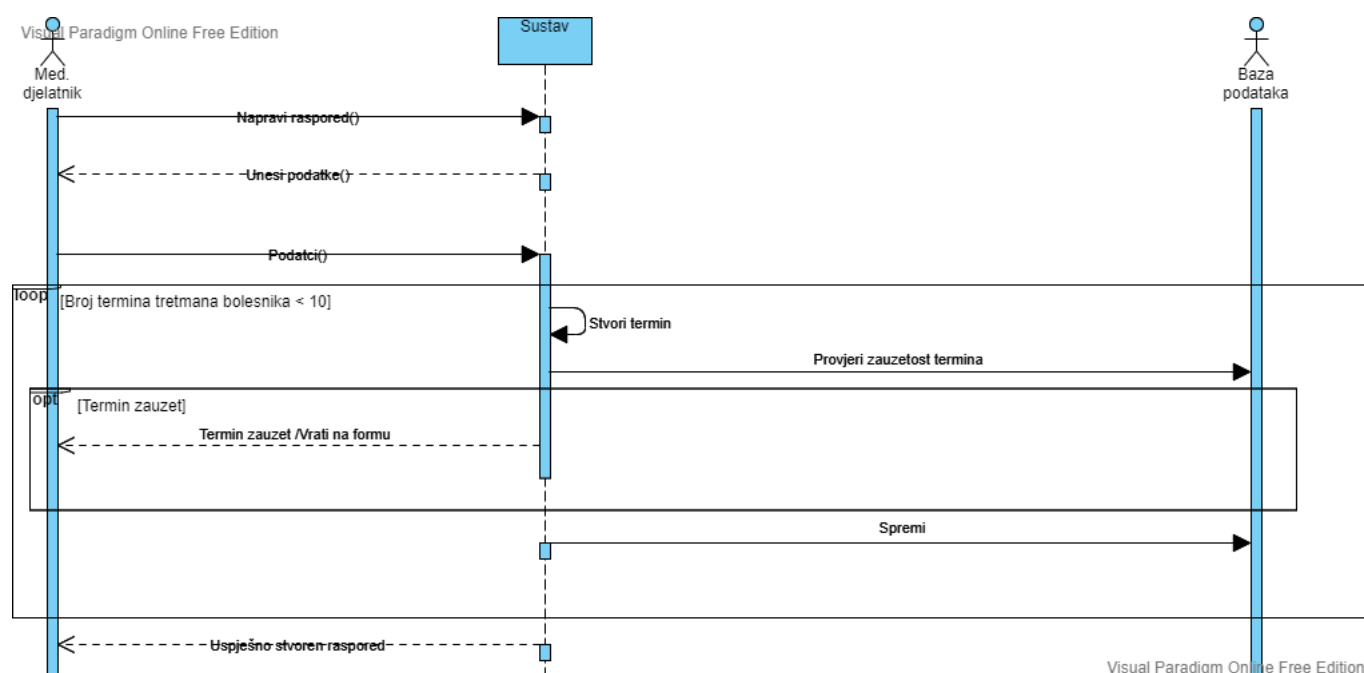
Medicinski djelatnik započinje registraciju bolesnika odabirom opcije na svom korisničkom sučelju. Djelatnik šalje podatke potrebne za registraciju te sustav, ako su podatci ispravni, šalje upit bazi o postojanju zapisa bolesnika s unesenim MBO-om. Ako takav zapis postoji, sustav javlja medicinskom djelatniku da je registracija uspješna. Ako zapis ne postoji, stvara objekt Korisnik koji se sprema u bazu i kasnije se koristi za autentifikaciju te nakon toga stvara objekt Pacijent koji se također sprema u bazu i kasnije koristi za upravljanje funkcionalnostima pacijenta. Na kraju sustav dojavljuje djelatniku da je registracija uspješna.



Slika 3.4: Sekvencijski dijagram UC5

Opis sekvencijskog dijagrama UC6

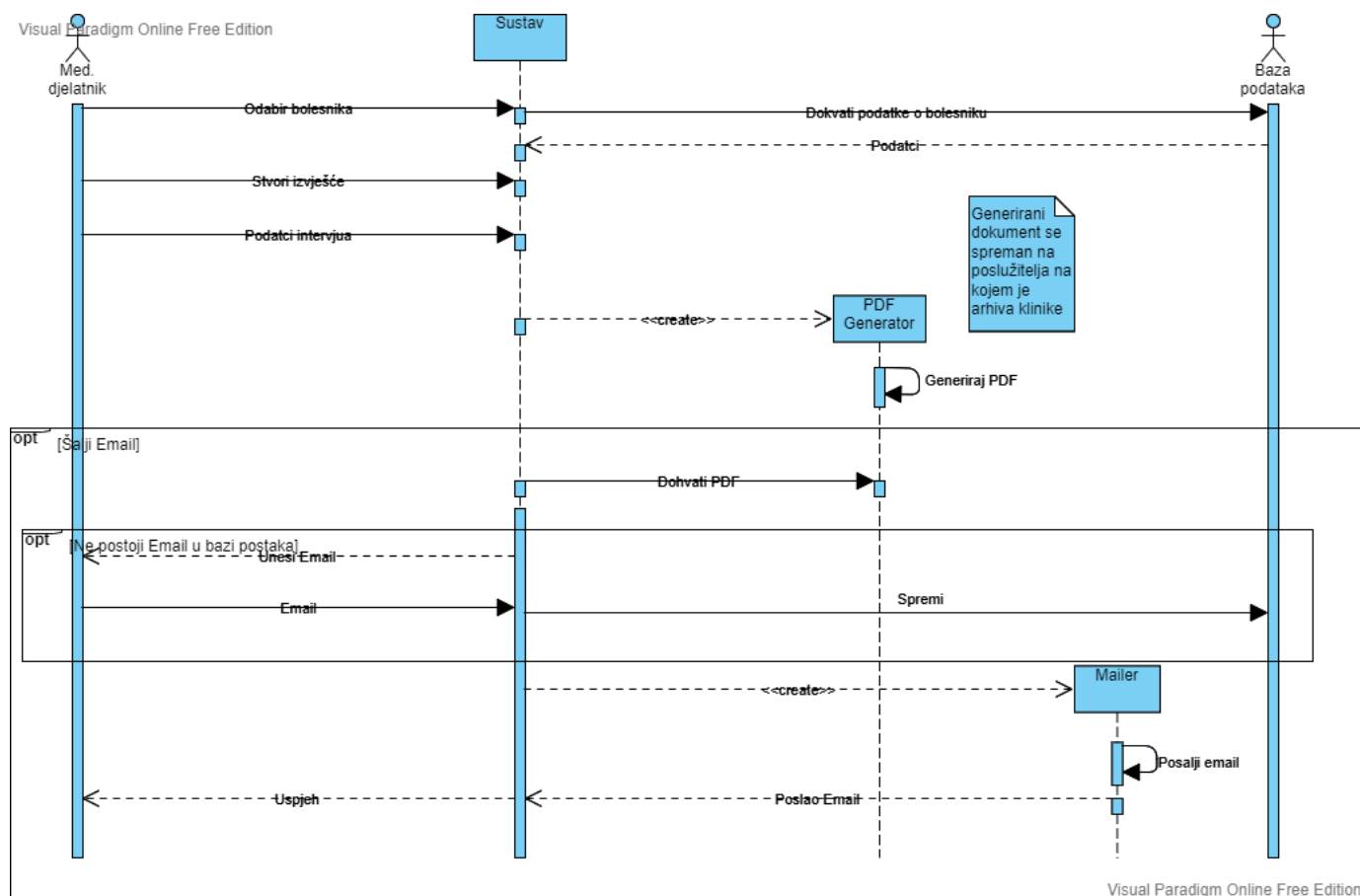
Medicinski djelatnik započinje izradu rasporeda odabirom opcije za izradu rasporeda iz korisničkog sučelja. Sustav očekuje podatke o tretmanima i terminima. Po primitku podataka, sustav u petlji stvara raspored 10 termina tretmana i provjerava jesu li termini unesenih tretmana zauzeti. Ako su termini zauzeti sustav šalje poruku medicinskom djelatniku o zauzetom terminu te ga vraća na sučelje za izradu rasporeda. Ako su svi termini slobodni, termini se spremaju u bazu podataka te sustav dojavljuje poruku o uspješnom stvaranju rasporeda.



Slika 3.5: Sekvencijski dijagram UC6

Opis sekvencijskog dijagrama UC8

Kada medicinski djelatnik želi izraditi izvješće za završetak bolesnikova tretmana, odabire bolesnika za kojeg želi izraditi izvješće. Sustav dohvaća podatke o bolesniku te, ako je bolesnik završio terapiju, djelatnik ima omogućen odabir funkcionalnosti izrade izvješća iz korisničkog sučelja. Sustav traži podatke od djelatnika te nakon što dobije podatke stvara objekt koji generira dokument i pohranjuje ga na poslužitelja koji ustanova koristi za arhivu. Nakon što objekt uspješno generira dokument, sustav o tome obavještava djelatnika i nudi mu opciju slanja dokumenta na adresu elektroničke pošte bolesnika. U slučaju da zapis o adresi elektroničke pošte bolesnika ne postoji, sustav od djelatnika traži da ju unese te ga sprema u bazu i stvara objekt zadužen za slanje dokumenta. Ako zapis postoji, sustav stvara objekt zadužen za slanje dokumenta.



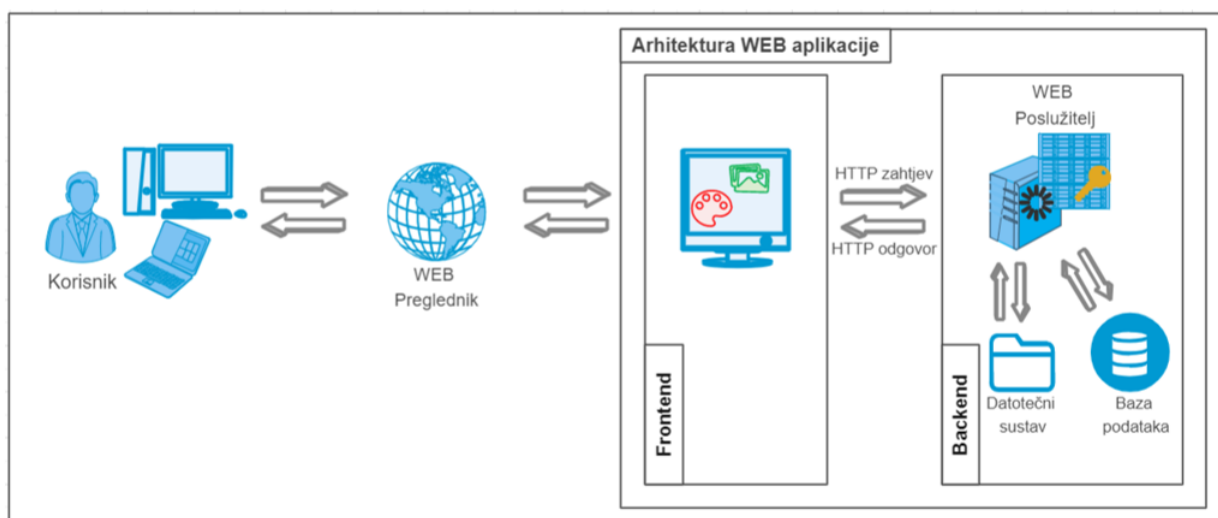
Slika 3.6: Sekvencijski dijagram UC8

3.2 Ostali zahtjevi

1. Dijagnoza se odabire iz liste dijagnoza bolesti prema važećoj MKB (Međunarodna klasifikacija bolesti i srodnih zdravstvenih problema)
2. Radno vrijeme ustanove je od 9:00 do 17:00 od ponedjeljka do petka, termini tretmana određeni ovim vremenom
3. Ograničen broj uređaja
4. Omogućen unos hrvatskih dijakritičkih znakova (UTF-8 encoding)
5. Sustav izveden u objektnoj paradigmi

4. Arhitektura i dizajn sustava

Arhitekturu sustava definiramo kao struktura sustava koja sadrži elemente programa i njihova izvana vidljiva obilježja te odnose među njima. Arhitekturu sustava dijelimo u nekoliko podsustava i prikazujemo njihovu međusobnu komunikaciju.



Slika 4.1: Arhitektura sustava

Arhitektura našeg sustava odnosno web aplikacije sastoji se od dva osnovna dijela:

- Frontend
- Backend

Frontend predstavlja sve što krajnji korisnik vidi (prezentacija sadržaja, uređenje te raspored elemenata web aplikacije) i sve čime vrši interakciju (unos, odabir, obrazac, gumbi, i dr.).

Backend predstavlja logiku web aplikacije odnosno kako web aplikacija funkcionira te kako pristupa podacima iz baze podataka i datotečnog sustava.

Frontend komunicira s backendom preko API-a. API (engl. *Application Programming Interface*) predstavlja način na koji dva ili više računalnih programa međusobno komuniciraju. Kod web aplikacija se ta komunikacija temelji na HTTP zahtjevima i HTTP odgovorima.

U backendu dijelu razlikujemo dva podsustava: web poslužitelj i baza podataka.

Web poslužitelj je temeljna komponenta za rad web aplikacije. Omogućuje komunikaciju klijenta (web preglednika) s web aplikacijom. Komunikacija je ostvarena već spomenutim HTTP (engl. *Hyper Text Transfer Protocol*) protokolom.

Web preglednik je posrednik u komunikaciji između korisnika i web poslužitelja. Omogućuje korisniku prikaz web stranice sa svim njenim elementima.

Baza podataka je skup međusobno povezanih podataka koji su tako organizirani da omoguće lakši pristup, pohranu i izmjenu podataka. Web poslužitelj komunicira s bazom podataka tako što šalje zahtjeve za podacima i dobiva podatke ukoliko postoje u bazi podataka.

Frontend je ostvaren u programskom jeziku JavaScript u radnom okviru React, backend u programskom jeziku PHP u radnom okviru Symfony, a baza podataka pomoću Heroku PSQl baze. Razvojna okruženja koja koristimo su Visual Studio Code i IntelliJ IDEA.

Oblikovanju arhitekture pristupili smo oblikovanjem od vrha prema dnu (engl. *Top-down design*). Krenuli smo od osnovnih koncepata i funkcionalnosti web aplikacije do krajnjih detalja izvedbe.

4.1 Baza podataka

Baza podataka je skup međusobno povezanih podataka koji su pohranjeni i precizno organizirani kako bi omogućili lakše upravljanje podacima (pohrana, dohvat i izmjena podataka). Ona predstavlja sliku stvarnog organizacijskog sustava te strukturom olakšava modeliranje stvarnog svijeta. Relacija (tablica) je objekt baze podataka. Definiramo ju kao imenovanu dvodimenzionalnu tablicu čije imenovane stupce nazivamo atributi, a retke nazivamo n-torkama. U ovoj web aplikaciji izrađena je relacijska baza podataka pomoću Heroku PSQl baze i sastoji se od sljedećih entiteta:

- User
- Staff
- Patient
- Appointment
- Treatment

4.1.1 Opis tablica

User Entitet sadrži osnovne informacije o korisniku web aplikacije. Atribute koje koristi su: id, name, surname, password, phone_number, email i roles. Entitet je u vezi *One-to-Zero-or-One*(1:0..1) s entitetom Staff pomoću atributa id te je u vezi *One-to-Zero-or-One*(1:0..1) s entitetom Patient pomoću atributa id.

user		
id	INT	Jedinstveni identifikator korisnika
name	VARCHAR	Ime korisnika
surname	VARCHAR	Prezime korisnika
password	VARCHAR	Lozinka korisnika
phone_number	VARCHAR	Broj telefona korisnika
email	VARCHAR	Adresa elektroničke pošte
roles	JSON	Uloga/razina ovlasti korisnika

Staff Entitet sadrži informacije o medicinskim djelatnicima zdravstvene ustanove. Attribute koje koristi su: id, user_id, qualifications, specialization i availability. Entitet je u vezi *Zero-or-One-to-One*(0..1:1) s entitetom User pomoću atributa user_id, u vezi je *One-or-Many-to-One-or-Many*(1..N:1..N) s entitetom Patient pomoću atributa id te je u vezi *One-to-One-or-Many*(1:1..N) s entitetom Appointment pomoću atributa id.

staff		
id	INT	Jedinstveni identifikator djelatnika
user_id	INT	Jedinstveni identifikator korisnika
qualifications	VARCHAR	Stručna sprema djelatnika
specialization	VARCHAR	Specijalizacija djelatnika
availability	BOOLEAN	Dostupnost/nedostupnost djelatnika

Patient Entitet sadrži informacije o bolesnicima, tj. korisnicima usluga zdravstvene ustanove. Attribute koje koristi su: id, user_id, doctor_id, diagnosis, insurance_id, is_insured i services. Entitet je u vezi *Zero-or-One-to-One*(0..1:1) s entitetom User pomoću atributa user_id, u vezi je *One-or-Many-to-One-or-Many*(1..N:1..N) s entitetom Staff pomoću atributa doctor_id te je u vezi *One-to-One-or-Many*(1:1..N) s entitetom Appointment pomoću atributa id.

patient		
id	INT	Jedinstveni identifikator pacijenta
user_id	INT	Jedinstveni identifikator korisnika
doctor_id	INT	Jedinstveni identifikator medicinskog djelatnika
diagnosis	VARCHAR	Dijagnoza (povijest bolesti) pacijenta
insurance_id	VARCHAR	MBO (matični broj osiguranika)
is_insured	BOOLEAN	Posjeduje li pacijent dodatno osiguranje
services	JSON	Tretmani na koje je pacijent upućen

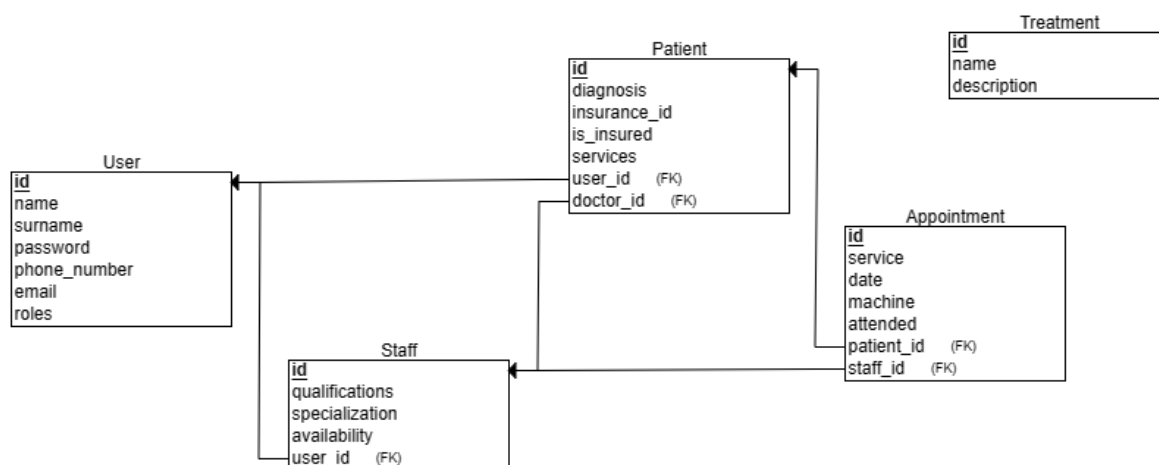
Appointment Entitet sadrži informacije o terminu tretmana za pacijenta. Attribute koje koristi su: id, patient_id, staff_id, service, date, attended i machine. Entitet je u vezi *One-or-Many-to-One*(1..N:1) s entitetom Patient pomoću atributa patient_id te je u vezi *One-or-Many-to-One*(1..N:1) s entitetom Staff pomoću atributa staff_id.

appointment		
id	INT	Jedinstveni identifikator termina tretmana
patient_id	INT	Jedinstveni identifikator pacijenta
staff_id	INT	Jedinstveni identifikator djelatnika
service	VARCHAR	Usluge/tretmani pacijenta
date	TIMESTAMP	Datum i vrijeme termina tretmana
attended	BOOLEAN	Prisutnost/dolaznost pacijenta
machine	VARCHAR	Uređaj potreban za tretman

Treatment Entitet sadrži informacije o nazivu i opisu usluge/tretmana zdravstvene ustanove. Attribute koje koristi su: id, name i description.

treatment		
id	INT	Jedinstveni identifikator tretmana/usluge
name	VARCHAR	Naziv tretmana/usluge
description	VARCHAR	Opis tretmana/usluge

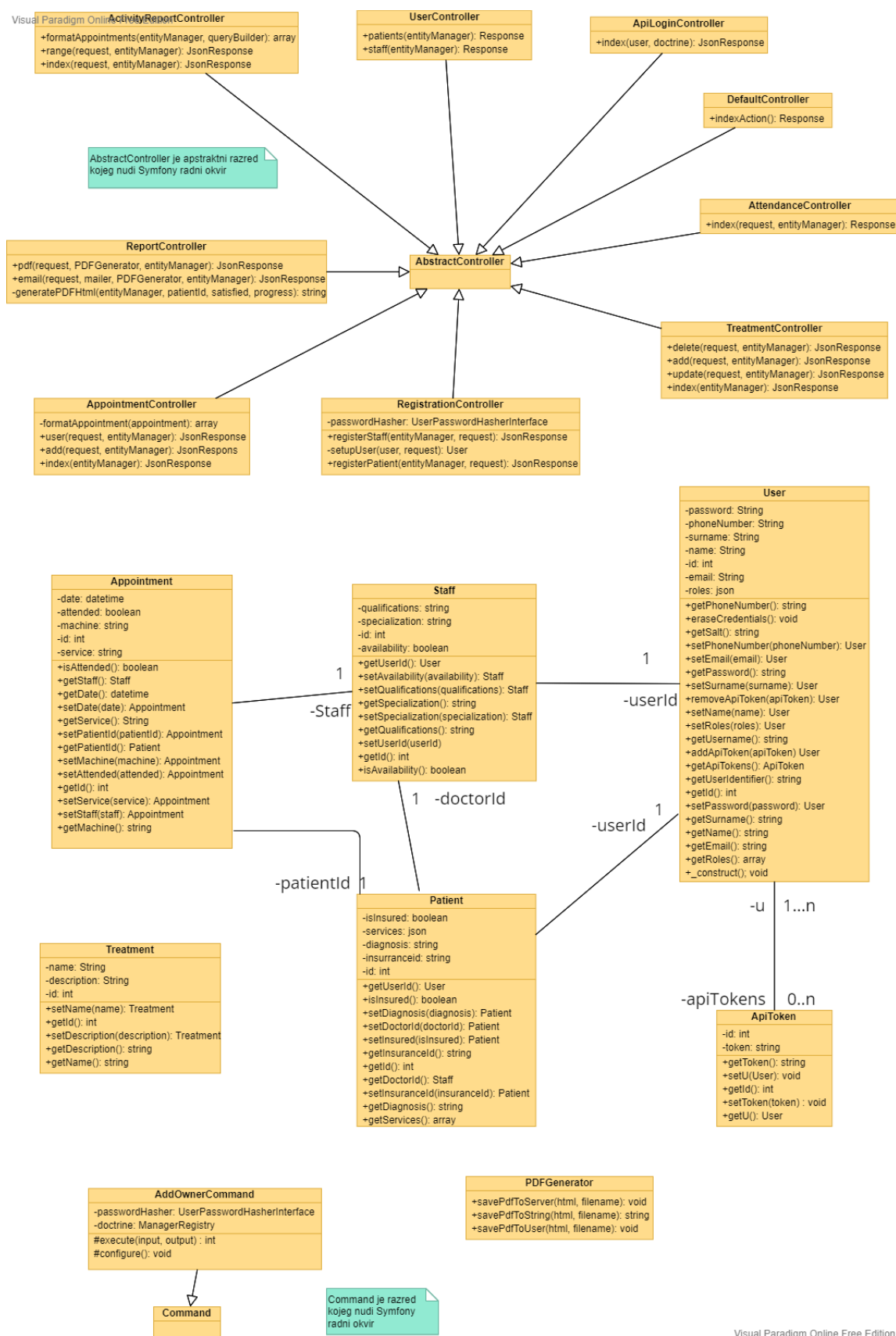
4.1.2 Dijagram baze podataka



Slika 4.2: E-R dijagram baze podataka

4.2 Dijagram razreda

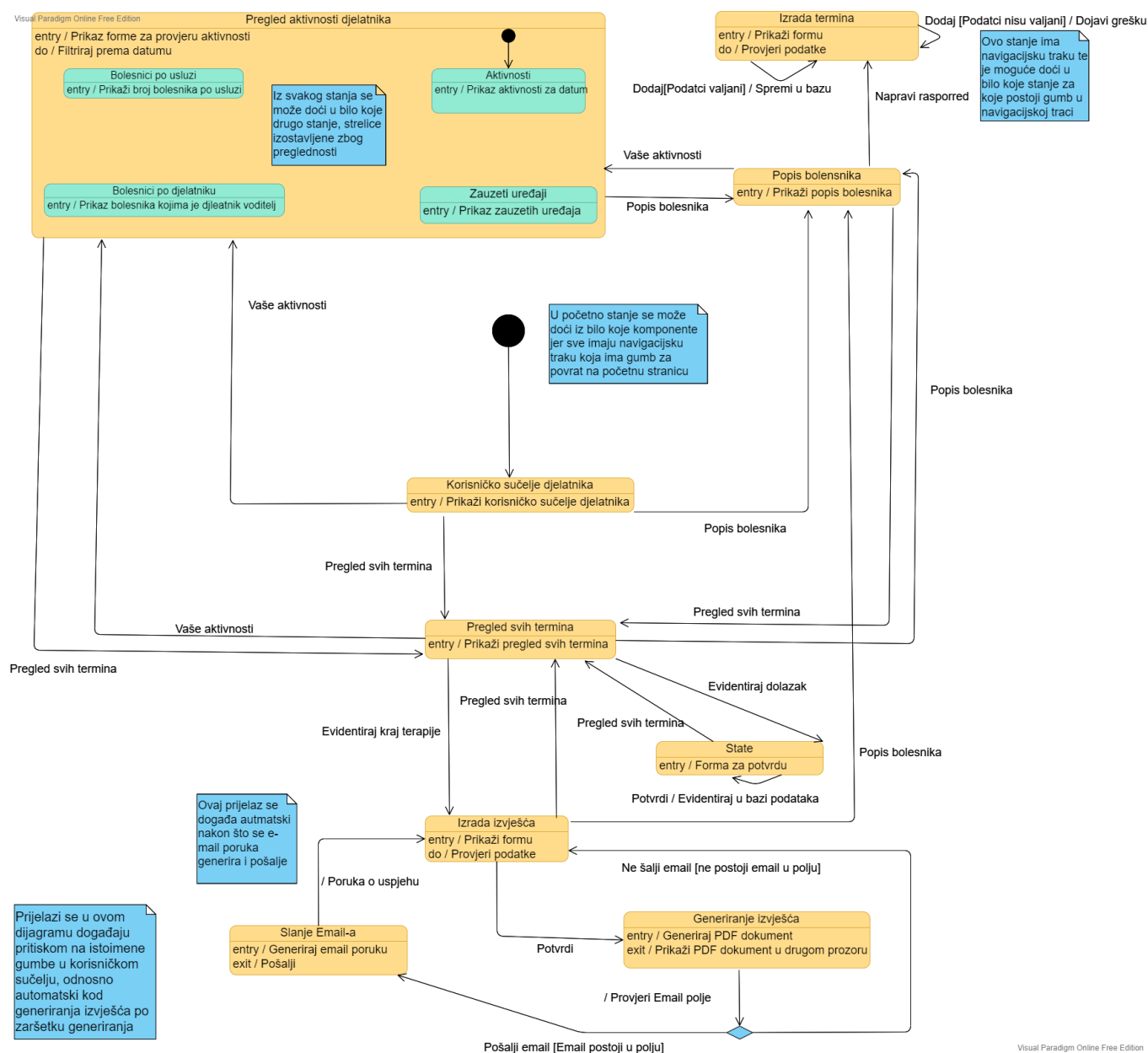
Svi razredi koji imaju sufiks "Controller" nasljeđuju razred `AbstractController` koji je dio Symfony radnog okvira te modelira kontroler komponente MVC arhitekture. Svaki od ovih razreda se bavi manipulacijom podataka te slanjem podataka na frontend kako bi se pogledi mogli ažurirati. `Appointment` razred modelira termin terapije te ima reference na djelatnika koje je voditelj te pacijenta kojemu je termin namijenjen. `Staff` predstavlja djelatnika ustanove. On ima referencu na razred `User`. Razred `User` predstavlja generaliziranog korisnika te služi za pridjeljivanje uloga korisnicima te čuva podatke potrebne za login. On ima referencu na 0 ili više objekata `ApiToken`. `ApiToken` služi autentifikaciju pri loginu. Razred `Patient` predstavlja pacijenta te ima referencu na `User` objekt te također na djelatnika koji ga je registrirao. `Treatment` enkapsulira informacije o pojedinoj usluzi koju ustanova nudi. `PDFGenerator` je razred koji služi za generiranje PDF dokumenta.



Slika 4.3: Dijagram razreda

4.3 Dijagram stanja

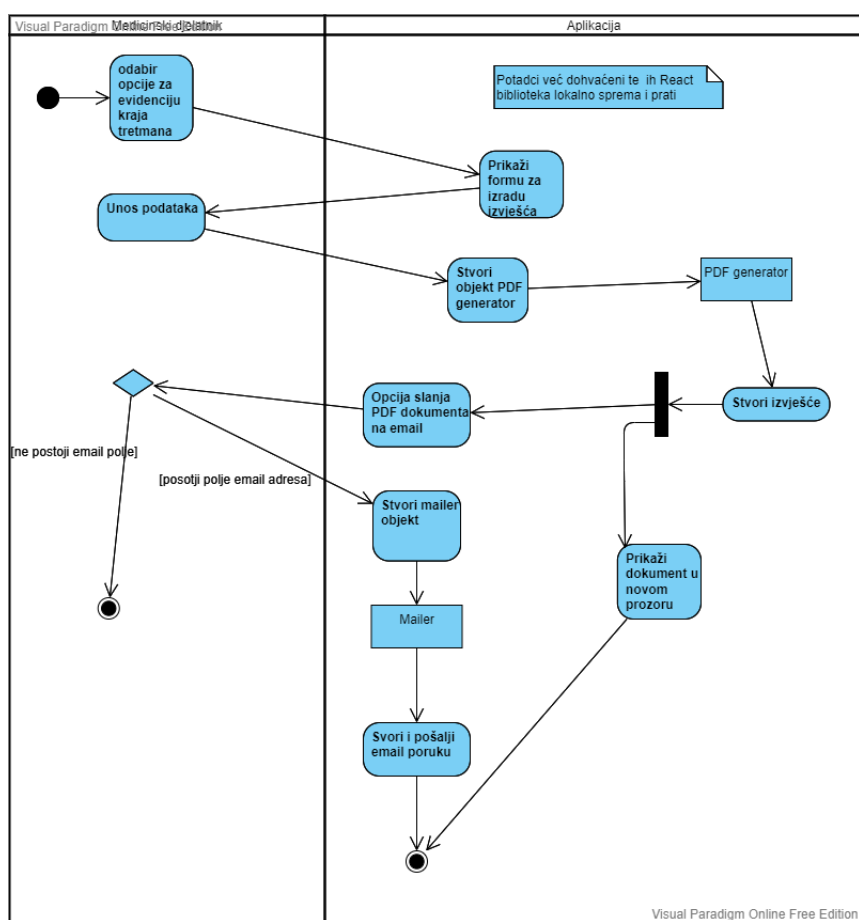
Dijagram stanja prikazuje način korištenja aplikacije od strane medicinskog djelatnika odnosno navigaciju korisničkim sučeljem specifičnim za medicinskog djelatnika. Prijelazi između stanja događaju se pritiskom na određeni gumb u sučelju. Nakon autentifikacije, sustav generira početnu stranicu koja je u ovom slučaju početno stanje. Iz ovog stanja pomoću opcija navigacijske trake djelatnik može prijeći u neko novo stanje, odnosno neki drugi pogled. Ulaskom u stanje "Pregled svih aktivnosti", sustav se prvo nalazi u podstanju "Aktivnosti". Ovisno o odabranoj opciji, filtrira i prikazuje podatke vezane uz odabranu opciju, odnosno podstanje u kojem se sustav trenutno nalazi. U stanju "Popis bolesnika" djelatnik odabire opciju "Napravi raspored" te sustav prelazi u novo stanje u kojem iscrta formu za izradu rasporeda. Pritiskom gumba "Dodaj" sustav dojavljuje poruku o uspjehu, a u suprotnom dojavljuje grešku korisniku te ostaje u istom stanju u oba slučaja. U stanju "Pregled svih termina" korisnik pritiskom na gumb "Evidentiraj dolazak" odlazi na formu za potvrdu evidencije dolaska bolesnika taj dan te se informacija o dolasku sprema u bazu podataka. U ovom stanju djelatnik također ima opciju evidentirati kraj terapije te čijim odabirom sustav prelazi u stanje "Izrada izvješća". U tom stanju sustav iscrta formu za unos podataka izvješća te provjerava podatke. Pritiskom na gumb potvrde sustav prelazi u stanje "Generiranje izvješća". U ovom stanju sustav stvara objekt PDF generator koji generira PDF dokument izvješća te ga prikazuje u novom prozoru. Sustav nakon toga izlazi iz trenutnog stanja te provjerava je li polje Email bilo popunjeno. Ako postoji email u polju, sustav prelazi u stanje "Slanje Email-a" u kojem objekt Mailer generira i šalje poruku koja sadrži izvješće te prelazi u stanje "Izrada izvješća" i dojavljuje poruku od uspjehu. Ako ne postoji e-mail adresa, prelazi se u stanje "Izrada izvješća".



Slika 4.4: Dijagram stanja

4.4 Dijagram aktivnosti

Djelatnik odabire opciju za evidenciju kraja tretmana bolesniku. Sustavi interno već pamti podatke tako da ih nije potrebno dohvaćati. Aplikaciju prikazuje formu za izradu izvješća te ju djelatnik popunjava. Aplikacija stvara objekt PDF generator te mu prosljeđuje podatke o pacijentu i unesene podatke. PDF generator stvara izvješće te otvara izvješće u PDF obliku u novom prozoru. Aplikacija tada provjerava postoji li email u email polju forme te ako postoji šalje dokument na tu adresu i obavještava korisnika o uspjehu te proces završava. Ako je polje email prazno, proces se završava



Slika 4.5: Dijagram stanja

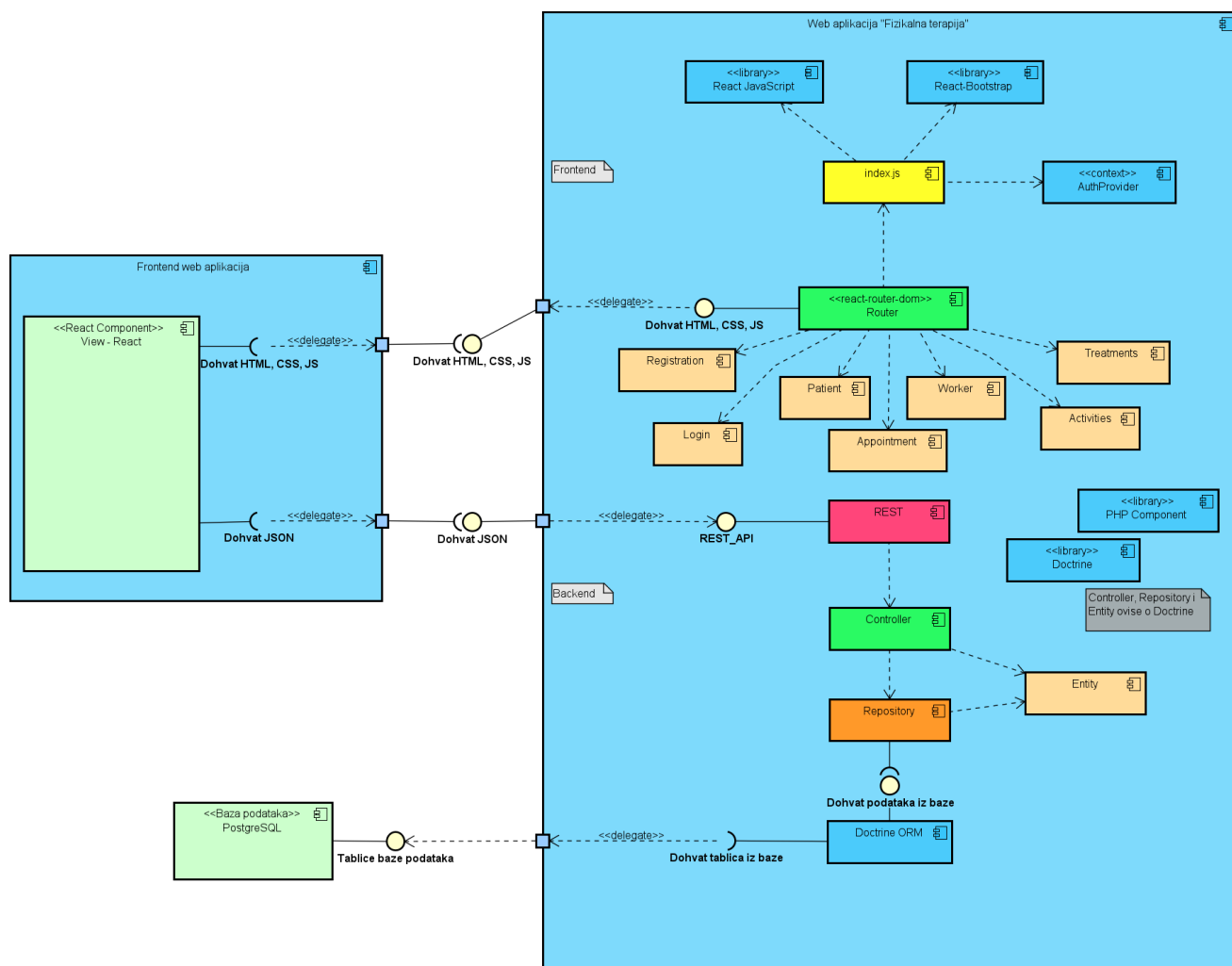
4.5 Dijagram komponenti

Dijagram komponenti predstavlja UML dijagram koji omogućuje vizualizaciju organizacije implementiranih komponenti te njihove međusobne ovisnosti. Također prikazuje odnos programske potpore prema okolini. U našem komponentnom dijagramu prikazanom na slici 4.6 razlikujemo 3 osnovne komponente, a to su frontend web aplikacija, web aplikacija (sustav) i baza podataka.

Prvo sučelje je sučelje za dohvat HTML, CSS i JS datoteka. Preko njega se poslužuju datoteke koje pripadaju frontend dijelu aplikacije. Router je glavna komponenta unutar sustava koja upravlja datotekama te određuje koja će se datoteka poslužiti na sučelje. Frontend dio sustava obuhvaća nekoliko JavaScript (JS) komponenti podijeljenih po načelu aktora i osnovnih aktivnosti koje pruža web aplikacija. Sve JS komponente ovise o React-ovim bibliotekama.

Drugo sučelje je sučelje za dohvat JSON podataka. Preko njega se pristupa REST API komponenti koja je zadužena za posluživanje datoteka sa backend dijela aplikacije. Backend dio sustava je ostvaren kroz 3 međusobno povezane komponente Controller, Repository te Entity.

Pomoću Doctrine ORM dohvaćaju se podaci iz baze podataka te poslužuju dalje na korištenje komponentama backend dijela aplikacije. React view preko navedenih sučelja komunicira sa web aplikacijom (sustavom) te ovisno o korisnikovim akcijama i zahtjevima osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.6: Dijagram komponenti web aplikacije

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za izradu dokumentacije korišten je alat LaTeX(<https://www.latex-project.org/>) u radnom okruženju TeXstudio(<https://www.texstudio.org/>) koristeći TeX Live distribuciju(<https://www.tug.org/texlive/>) LaTeX-a. LaTeX je korišten jer je defacto industrijski standard za izradu tehničke dokumentacije i znanstvenih radova. Za izradu dijagrama korišten je besplatni alat za izradu UML dijagrama Visual Paradigm Online(<https://online.visual-paradigm.com/features/>) jer nudi široku selekciju paketa komponenti za velik broj UML dijagrama i jednostavan je za korištenje. Za izradu relacijske baze podataka korišten je PostgreSQL(<https://www.postgresql.org/>) sustav za upravljanje bazama podataka. PostgreSQL poštuje ACID principe pri izvođenju transakcija te se pridržava većine SQL2011 standarda. Frontend aplikacije izrađen je u programskom jeziku JavaScript(<https://en.wikipedia.org/wiki/JavaScript>, <https://www.w3schools.com/js/>) koristeći besplatnu biblioteku otvorenog koda React(<https://reactjs.org/>) koja nudi komponente korisničkog sučelja koje se učinkovito ažuriraju i iscrtavaju te omogućuju stvaranje interaktivnih korisničkih sučelja njihovim grupiranjem. Također, VS-Code je korišten kao radno okruženje zbog svoje male veličine i široke potpore za JavaScript. Backend funkcionalnosti implementirane su u programskom jeziku PHP(<https://www.php.net/>, <https://www.w3schools.com/php/>), specifično u radnom okviru otvorenog koda Symfony(<https://symfony.com/>) koji je skup biblioteka i ponovno iskoristivih PHP komponenti koji nudi dugoročnu potporu i skalabilnost sustava. Izvorni kodovi te resursi i dokumentacija su bili pohranjeni u GitLab(<https://about.gitlab.com/>) repozitorij.

5.2 Ispitivanje programskog rješenja

Ispitivanje programske potpore je postupak otkrivanja informacija o ispravnosti i kvaliteti programske potpore te omogućuje ispravak i poboljšanje ispitivane programske potpore pronalaženjem kvarova. Ispitni slučaj je osnovna jedinica ispitivanja. Predstavljen je kao uređeni par (ulaz, izlaz), gdje je ulaz ulazni podatak, a izlaz unaprijed zabilježen očekivani izlazni podataka.

Za potrebe našeg projekta izvršili smo dvije vrste ispitivanja: ispitivanje komponenti i ispitivanje sustava.

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti vrši postupak verifikacije rada programskih dijelova koje je moguće zasebno ispitivati (funkcije, razrede sa više atributa i metoda te neke složenije komponente). Ispitivanje smo proveli koristeći **JUnit** radni okvir, koji olakšava izradu testova i ubrzava provjeru ispravnosti koda.

5.2.2 Ispitivanje sustava

Ispitivanje sustava je postupak ispitivanja završne i potpuno integrirane inačice programske potpore namijenjene distribuciji korisniku. Ovim načinom ispitivanja ispitujemo potpuni sustav u cjelini. Za ispitivanje sustava izradili smo testove pomoću **Selenium WebDriver**, radnog okvira koji omogućuje ispitivanje cijelog sustava programskom interakcijom sa web preglednicima. Radi jednostavnijeg ispitivanja izveli smo Selenium testove unutar JUnit testova.

Programski testovi su izrađeni za sljedećih 4 funkcionalnosti:

- Prijava korisnika u sustav
- Registracija medicinskog djelatnika
- Upis novih tretmana/usluga
- Evidencija dolaska bolesnika na terapiju

Prijava korisnika u sustav (*loginDriver*)

Test ispituje može li se korisnik prijaviti u sustav sa dodijeljenom e-mail adresom i lozinkom.

```
/**
 * Logs in the driver.
 */
private void loginDriver(String email, String password) {
    // Go to staff login.
    driver.get(BASE_URL + "login_osoblje");

    // Fill the form.
    WebElement emailInput = driver.findElement(By.id("email"));
    emailInput.sendKeys(email);

    WebElement passwordInput = driver.findElement(By.id("lozinka"));
    passwordInput.sendKeys(password);

    // Submit the form.
    driver.findElement(By.cssSelector("input[type='submit']")).click();

    // Wait for page to load.
    try {
        Thread.sleep(2000);
    } catch (InterruptedException ignorable) {}
}

@BeforeAll
/**
 * Initializes driver.
 */
public static void setup() {
    System.setProperty("webdriver.chrome.driver", "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    driver = new ChromeDriver();
}
```

Slika 5.1: Selenium - test 1

Ispitni slučajevi koje ispituje u testu 1 su ispravan unos podataka, tj. postojeći korisnik *testLogin* i neispravan unos, tj. nepostojeći korisnik *testLoginFail*. Sustav u drugom slučaju treba dojaviti pogrešku.

```
@Test
/**
 * Test login owner.
 */
public void testLogin() {
    // Login as owner.
    loginDriver("martin.bakaccasdsasdsdfghjasdasd@test.com", "asdfghjk");

    // Redirect to base URL means success.
    assertEquals(BASE_URL, driver.getCurrentUrl());
}
```

Slika 5.2: Ispitni slučaj - Uspješna prijava

```
@Test
/**
 * Test login fail
 */
public void testLoginFail() {
    loginDriver("NULL", "NULL");

    assertEquals(BASE_URL + "login_osoblje", driver.getCurrentUrl());
}
```

Slika 5.3: Ispitni slučaj - Neuspješna prijava

Registracija medicinskog djelatnika (*registerWorker*)

Test ispituje mogućnost registracije medicinskog djelatnika u sustav od strane vođitelja ustanove ili vlasnika sustava.

```
/**
 * Attempts to register a worker as an owner.
 *
 * @param email of the worker.
 * @param password of the worker.
 * @param alertId id of the alert in the HTML.
 * @return the text of the alert.
 */
private String registerWorker(String email, String password, String alertId) {
    // Login as owner.
    loginDriver("martin.bakaccsdssasdsdfghjasdasd@test.com", "asdfghjk");

    // Go to adding the worker location.
    driver.get(BASE_URL + "add_worker");

    // Fill the form.
    driver.findElement(By.id("email")).sendKeys(email);
    driver.findElement(By.id("password")).sendKeys(password);
    driver.findElement(By.id("name")).sendKeys(String.format("Zlatko%d", ACCOUNT_NUMBER));
    driver.findElement(By.id("surname")).sendKeys(String.format("Horvat%d", ACCOUNT_NUMBER));
    driver.findElement(By.id("phoneNumber")).sendKeys(String.format("091472398$d", ACCOUNT_NUMBER));
    driver.findElement(By.id("qualifications")).sendKeys("Doktor");
    driver.findElement(By.id("specialization")).sendKeys("Otorinolaringolog");
    Select dropDown = new Select(driver.findElement(By.id("availability")));
    dropDown.selectByVisibleText("Dostupan");

    // Send the form.
    driver.findElement(By.id("test-register")).click();
    try {
        Thread.sleep(2000);
    } catch (InterruptedException ignorable) {}
    String res = driver.findElement(By.id(alertId)).getAttribute("innerHTML");

    return res;
}
```

Slika 5.4: Selenium - test 2

Ispitni slučajevi koje ispituje u testu 2 su uspješna registracija medicinskog djelatnika *registerWorkerSuccess* i neuspješna registracija medicinskog djelatnika, tj. pokušaj registracije postojećeg djelatnika *registerWorkerFail*. Sustav u drugom slučaju treba dojaviti pogrešku.

```
@Test
/**
 * Test registering a worker.
 */
public void registerWorkerSuccess() {
    assertEquals("Djelatnik uspješno registriran", registerWorker(String.format("selenium%d@test.com", ACCOUNT_NUMBER), "password", "alert-check"))
}
```

Slika 5.5: Ispitni slučaj - Uspješna registracija

```
@Test
/**
 * Test adding the same worker from the previous test.
 * It should fail
 */
public void registerWorkerFail() {
    assertEquals("E-mail adresa se već koristi", registerWorker("dev@dev.com", "password", "alert-fail"));
}
```

Slika 5.6: Ispitni slučaj - Neuspješna registracija

Upis novih tretmana/usluga (*addService*)

Test ispituje mogućnost upisa novih tretmana odnosno usluga u sustav.

```
/**
 * Attempts to add a service with a given name and a description.
 *
 * @param name of the service.
 * @param description of the service.
 * @return the content of the response.
 */
private String addService(String name, String description) {
    // Login as owner.
    loginDriver("martin.bakaccasdsasdsdfghjasdasd@test.com", "asdfghjk");

    // Go to adding services.
    driver.get(BASE_URL + "add_treatmen");

    // Select the add action and fill the form.
    driver.findElement(By.id("treatment-add")).click();
    driver.findElement(By.id("ime")).sendKeys(name);
    driver.findElement(By.id("opisNovog")).sendKeys(description);

    // Send the form.
    driver.findElement(By.id("add-modal")).click();
    try {
        Thread.sleep(2000);
    } catch (InterruptedException ignorable) {}

    // Check for success.
    WebElement el = driver.findElement(By.id("feedback-button"));
    String res = el.getAttribute("innerHTML");

    return res;
}
```

Slika 5.7: Selenium - test 3

Ispitni slučajevi koje ispitujemo u testu 3 su uspješan upis tretmana *testAddService* i neuspješan upis tretmana, tj. pokušaj upisa već postojećeg tretmana odnosno tretmana istog naziva *testAddServiceFail*. Sustav u drugom slučaju treba dojaviti pogrešku.

```
@Test
/**
 * Test adding a service.
 */
public void testAddService() {
    assertEquals("Uspješno dodavanje tretmana", addService("Testiranje frontenda", "Opis testnog primjera"));
}
```

Slika 5.8: Ispitni slučaj - Uspješan unos tretmana

```
@Test
/**
 * Adding a service of the same name.
 */
public void testAddServiceFail() {
    addService("DUPLIKAT", "opis raziličit");
    assertEquals("Ime tog tretmana vec postoji", addService("DUPLIKAT", "opis isti"));
}
```

Slika 5.9: Ispitni slučaj - Neuspješan unos tretmana

Evidencija dolaska bolesnika na terapiju (*logAppointment*)

Test ispituje mogućnost evidentiranja dolaska pristiglog bolesnika na terapiju.

```
@Test
/**
 * Test logging an appointment. This test can take more time, depending on the connection, so it is slowed down.
 */
public void logAppointment() {
    // Login as a worker.
    loginDriver("dev@dev.com", "password");

    // Go to appointments.
    driver.get(BASE_URL + "appointments");
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ignorable) {}

    driver.get(BASE_URL + "appointments");
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ignorable) {}

    // Select the appropriate log appointment button.
    driver.findElement(By.cssSelector(PATIENT_ACCORDION_SELECTOR)).click();
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ignorable) {}
    driver.findElement(By.cssSelector(LOG_BUTTON_SELECTOR)).click();
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ignorable) {}

    // Press confirm.
    driver.findElement(By.id("test-button")).click();
    try {
        Thread.sleep(10000);
    } catch (InterruptedException ignorable) {}

    // Check for success.
    String res = driver.findElement(By.id("good-message")).getAttribute("innerHTML");
    assertTrue(res.contains("Evidencija uspjela."));
}
```

Slika 5.10: Selenium - test 4

Ispitni slučaj koji ispitujemo u testu 4 je uspješna evidencija dolaska bolesnika na terapiju. Ispitni slučaj je ugrađen u testu.

```
/**
 * This class runs 4 unit tests which test different uses of the system.
 *
 * @author Fran Đorđević
 */
public class SystemTest {

    /**
     * CSS selector for opening the list of appointments for a patient.
     */
    private static final String PATIENT_ACCORDION_SELECTOR = "#root > section > div > div:nth-child(10) > div > div > h2 > button";

    /**
     * CSS selector for logging an appointment. This changes so check if the button is available.
     */
    private static final String LOG_BUTTON_SELECTOR = "#root > section > div > div:nth-child(10) > div > div > div > table > tbody > tr:nth";

    /**
     * Number used to test creation of a new worker.
     */
    private static final int ACCOUNT_NUMBER = 7;

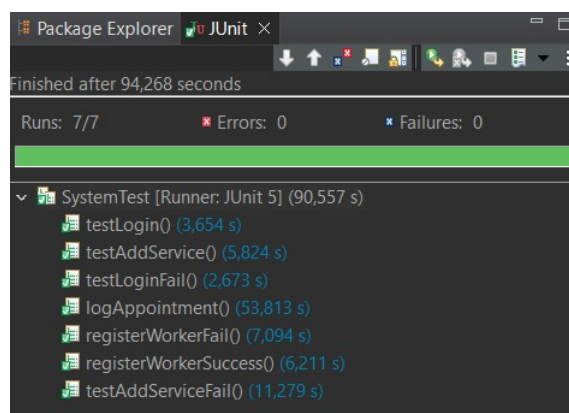
    /**
     * Base URL of the web application.
     */
    private static final String BASE_URL = "https://fctestf1.netlify.app/";

    /**
     * Driver used for the tests.
     */
    private static WebDriver driver;
```

Slika 5.11: Konstante korištene u testovima

Rezultati ispitivanja sustava

Ispitni slučajevi su izvedeni kao JUnit testovi i ostvarili su zadovoljavajuće rezultate. Nisu nastupile pogreške niti zatajenja sustava.



Slika 5.12: Rezultati ispitivanja sustava

5.3 Dijagram razmještaja

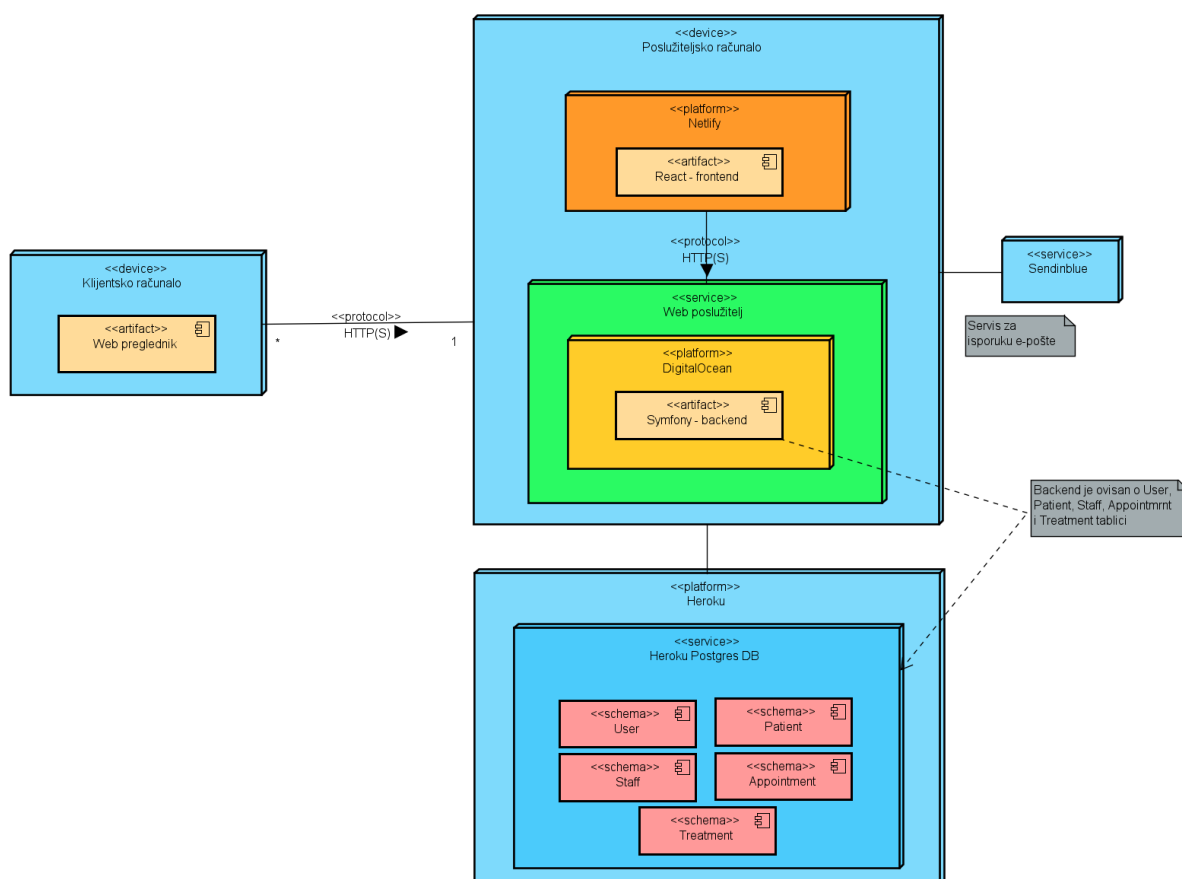
Dijagram razmještaja predstavlja UML dijagram koji opisuje topologiju sustava odnosno odnos sklopovskih i programskih dijelova. Naš dijagram razmještaja prikazan slikom 5.1 predstavlja specifikacijski dijagram razmještaja.

Klijentsko računalo peko web preglednika pristupa našoj web aplikaciji. Pritom se ostvaruje HTTP(S) veza između klijentskog računala i poslužiteljskog računala.

Poslužiteljsko računalo se sastoji od frontend dijela aplikacije koji je razmješten (engl. deploy) na platformi Netlify i web poslužitelja koji je ostvaren backend dijelom aplikacije te razmješten (engl. deploy) na platformi DigitalOcean. Frontend i backend komuniciraju također već spomenutim HTTP(S) protokolom.

Na platformi Heroku je razmještena baza podataka sa 5 entiteta. Backend i baza podataka su međusobno ovisni te razmjenjuju podatke.

Također sa poslužiteljskim računalom je u vezi i servis Sendinblue koji služi za isporuku e-pošte iz aplikacije.



Slika 5.13: Dijagram razmještaja web aplikacije

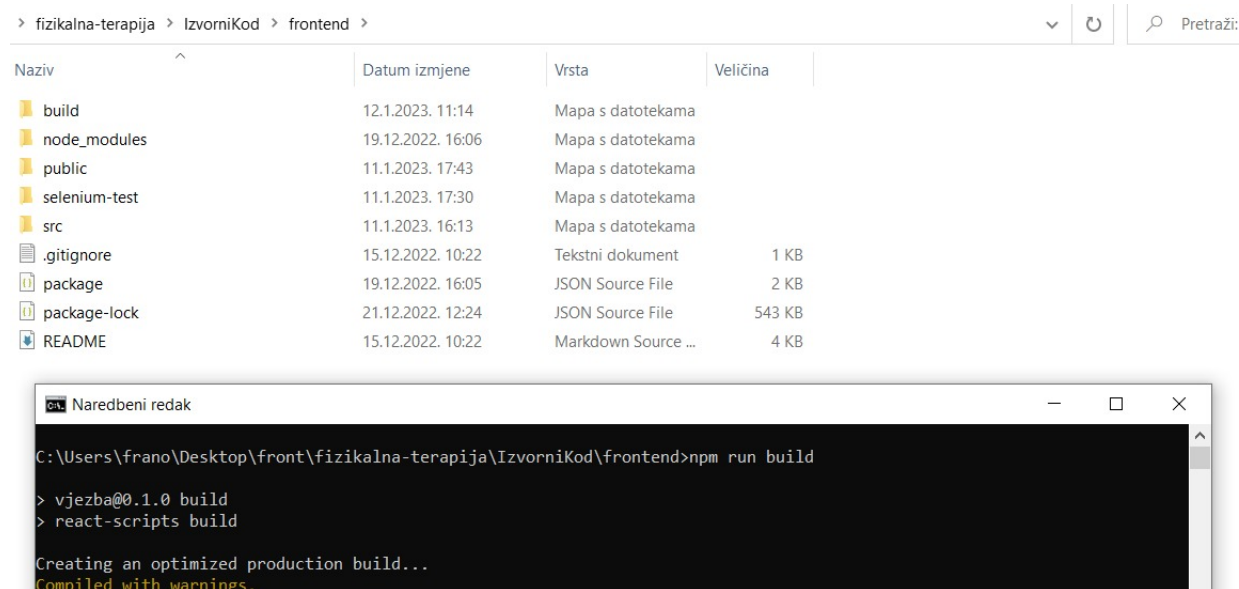
5.4 Upute za puštanje u pogon

Puštanje baze podataka u pogon:

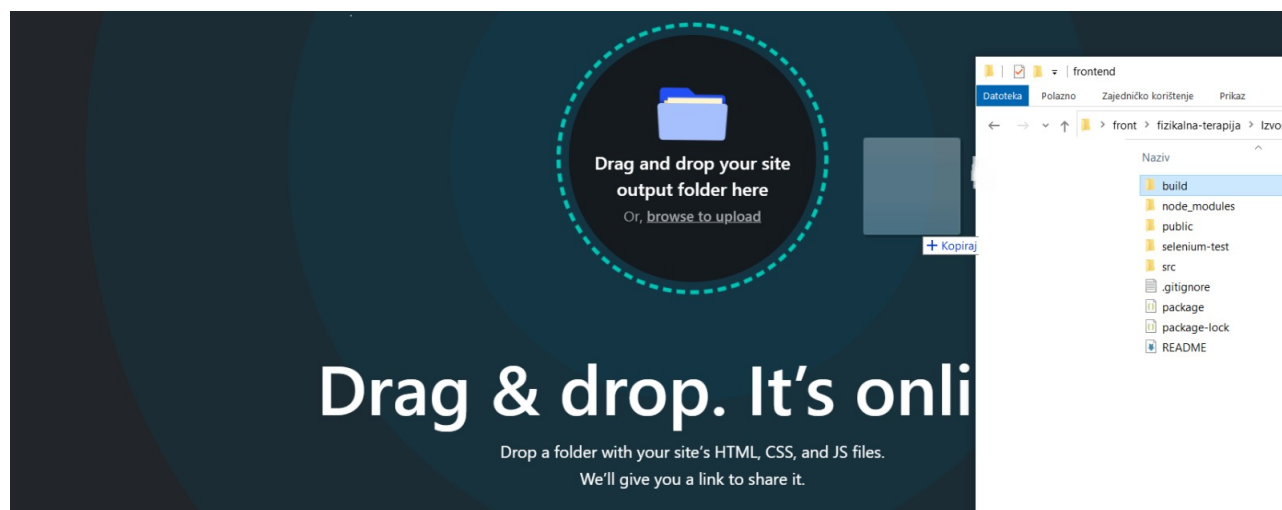
- PostgreSQL baza se postavlja na Heroku poslužitelja
- Podatci za spajanje na bazu se unose u konfiguracijsku datoteku backend-a
- Pokreće se migracijska skripta objektno-relacijskog rukovoditelja na backend-u kako bi se iz programiranih entiteta stvorile tablice relacijske baze podataka
- Ostatak posla obavlja objektno objektno-relacijskog rukovoditelja (dohvaćanje, spremanje ažuriranje podataka u bazi)

Puštanje frontend-a u pogon:

- U direktoriju gdje se nalazi frontend kod pokreće se *npm build* naredba iz naredbenog retka
- Nakon toga se na Netlify servis(<https://www.netlify.com/>) postavi kod iz direktorija u kojem je bila pokrenuta *npm build* naredba
- Frontend u sebi sadrži vezu na backend te nije potrebno nikakvo dodatno konfiguriranje



Slika 5.14: npm run naredba



Slika 5.15: Drag and drop na Netlify poslužitelj

Puštanje backend-a u pogon:

- Koristi se račun na servisu za objavu aplikacija (engl. web host), kao što je DigitalOcean (<https://www.digitalocean.com/>). Ako se koristi DigitalOcean, u njemu se stvara novi poslužitelj (engl. droplet).
- Stvara se direktorij gdje će se nalaziti backend aplikacija, npr. `/var/www/be/fizikalna-terapija/` te se u njega kopira izvorni kod backenda aplikacije pomoću naredbe `git clone https://gitlab.com/progi_tim_2022/fizikalna-terapija.git`.
- Konfiguriraju se varijable okruženja (engl. environment variables) na poslužitelju. To se radi pozicioniranjem u direktorij `/var/www/be/fizikalna-terapija/IzvorniKod/backend/` te uređivanjem datoteke `.env`. Ovdje se dodaju linije `APP_ENV=prod` te `APP_DEBUG=0`.
- Zatim se instalira `php7.2-xm1` na poslužitelju pomoću naredbe `sudo apt-get install php7.2-xm1`, kako bi se omogućilo parsiranje XML datoteka.
- Projektne se ovisnosti instaliraju pomoću naredbe `composer install` te se pokreće migracijska skripta na backend-u kako bi se iz programiranih entiteta stvorile tablice relacijske baze podataka. Migracija se pokreće naredbom `php bin/console doctrine:migrations:migrate` nakon pozicioniranja u direktorij `/var/www/be/fizikalna-terapija/IzvorniKod/backend/`. Provjera je li migracija uspješno izvršena moguća je pomoću naredbe `php bin/console doctrine:migrations:status`. Migracija je uspješno izvršena ako se u naredbenom retku ispiše `No migrations to execute`.
- Instalira se potrebna programska podrška nužna za izvedbu projekta u radnom okviru Symfony. To se može napraviti pomoću Symfonyjevog alata Sym-

fony CLI kojeg se instalira pokretanjem naredbe `wget https://get.symfony.com/cli/installer` -O - — *bash* u naredbenom retku. Nakon instalacije navedenog programa moguće je testirati jesu li potrebni programi instalirani pomoću naredbe `symfony check:requirements`.

- Poslužitelj se podešava uređivanjem datoteke `/etc/nginx/sites-available/default` te se dodaje sljedeći sadržaj:

```
# Default server configuration
#
server {
    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/be/fizikalna-terapija/IzvorniKod/backend/public;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html index.php;

    server_name martinjosipkocijan.from.hr www.martinjosipkocijan.from.hr;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
        proxy_pass https://127.0.0.1:8080;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #    include snippets/fastcgi-php.conf;
    #
    #    # With php-fpm (or other unix sockets):
    #    fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #    # With php-cgi (or other tcp sockets):
    #    fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny all;
    #}

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/martinjosipkocijan.from.hr/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/martinjosipkocijan.from.hr/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
}
```

Slika 5.16: Postavke za poslužitelj (1)

```
server {
    if ($host = www.martinjosipkocijan.from.hr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

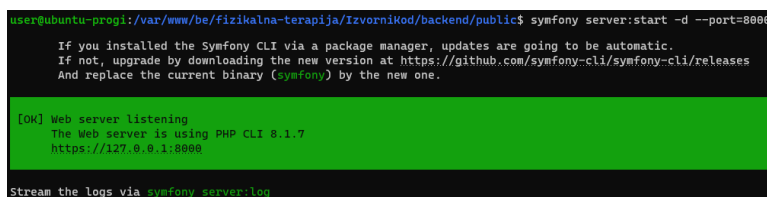
    if ($host = martinjosipkocijan.from.hr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 default_server;
    listen [::]:80 default_server;

    server_name martinjosipkocijan.from.hr www.martinjosipkocijan.from.hr;
    return 404; # managed by Certbot
}
```

Slika 5.17: Postavke za poslužitelj (2)

- Dodani se sadržaj sprema i provjerava se ispravnost konfiguracije pomoću naredbe `sudo nginx -t`. Ako je konfiguracija ispravna, ponovno se pokreće nginx poslužitelj pomoću naredbe `sudo systemctl restart nginx`.
- Kako bi HTTPS bio omogućen na poslužitelju, instalira se SSL certifikat kao što je onaj koji nudi Let's Encrypt. To se može napraviti pomoću naredbe `sudo certbot --nginx -d domena` gdje se *domena* zamjenjuje nazivom domene na kojoj će se nalaziti backend aplikacije. Nakon toga se u datoteci `/etc/nginx/sites-available/default` u bloku `server` dodaje linija `proxy_set_header X-Forwarded-Proto https;`. Nakon toga se ponovno pokreće nginx poslužitelj pomoću naredbe `sudo systemctl restart nginx`.
- Kroz vatrozid se omogućava pristup backend aplikaciji. To se može napraviti pomoću naredbe `sudo ufw allow 'Nginx HTTPS'`. Sprema se i provjerava se ispravnost konfiguracije pomoću naredbe `sudo ufw status`. Ako je konfiguracija ispravna, ponovno se pokreće vatrozid pomoću naredbe `sudo systemctl restart ufw`.
- Aplikacija se pokreće pomoću naredbe `symfony server:start -d --port=8000` u direktoriju `/var/www/be/fizikalna-terapija/IzvorniKod/backend/public`:



```
user@ubuntu-progi:/var/www/be/fizikalna-terapija/IzvorniKod/backend/public$ symfony server:start -d --port=8000
If you installed the Symfony CLI via a package manager, updates are going to be automatic.
If not, upgrade by downloading the new version at https://github.com/symfony-cli/symfony-cli/releases
And replace the current binary (symfony) by the new one.

[OK] Web server listening
The Web server is using PHP CLI 8.1.7
https://127.0.0.1:8000

Stream the logs via symfony server:log
```

Slika 5.18: Pokretanje backend poslužitelja Symfony

6. Zaključak i budući rad

Projekt izrade web aplikacije "Zdravljem do znanja" u sklopu nastavnog kolegija „Programsko inženjerstvo“ uspješno je završen. Kada kažemo uspješno završen onda ne mislimo samo na uspješan konačni produkt projekta odnosno web aplikaciju, već na cijeli životni ciklus projekta. Na projektu su radili sedam međusobno nepoznatih studenata koji nakon što su se upoznali i prilagodili rad jedni drugima, započeli su intenzivan rad na projektu. Podijelili smo posao na 3 skupine. Prvu skupinu su činili trojica studenata zadužena za frontend dio web aplikacije, drugu skupinu dvojica studenata zadužena za backend dio web aplikacije te posljednju, no ne i manje važnu skupinu dvojica studenata zadužena za izradu i vođenje dokumentacije. Projekt se odvijao kroz dvije faze.

U prvoj fazi većina posla je bila usmjerena na dokumentaciju. Razlog tome je nužnost dokumentacije kao prethodni korak za ulazak u proces implementacije zahtjeva i izrade web aplikacije. Početna verzija dokumentacije je obuhvaćala popis funkcionalnih i nefunkcionalnih zahtjeva, oblikovanje i arhitekturu web aplikacije te nezaobilazni UML dijagrami obrazaca, sekvencijski dijagrami i dijagram razreda. Uz navedeno, u prvoj fazi je programski ostvarena i generička funkcionalnost registracije i prijave korisnika u sustav.

Druga faza projekta je obuhvaćala intenzivan rad na izradi web aplikacije i ostvarenju svih dokumentiranih zahtjeva, što su usklađeno i zajedničkim snagama realizirali skupina na frontend-u i backend-u. Osnovne funkcionalnosti koje su izrađene u sklopu web aplikacije su: registracija i prijava medicinskog djelatnika u sustav, registracija i prijava bolesnika u sustav, evidencija dolazaka bolesnika, izrada individualnog rasporeda tretmana za pojedinog bolesnika, unos kraja tretmana i zapažanja, izrada pdf dokumenta, pregled i unos tretmana i usluga te pregled aktivnosti medicinskih djelatnika i zauzeće uređaja. Također i u ovoj fazi aktivno, ali manje nego u prvoj fazi sudjeluje skupina za izradu dokumentacije. Konačna verzija dokumentacije proširuje početnu dokumentaciju ugrađujući dodatno UML dijagrame stanja, aktivnosti, komponenti i razmještaja te dokumentiranje cjelo-

kupnog procesa ispitivanja programske potpore i puštanja web aplikacije u pogon.

Projekt nas je proveo kroz razne situacije, od početnog entuzijazma za rad pa blagog narušavanja komunikacije i slabljenja interesa do konačnog savladavanja svih prepreka i uzdizanja do želje da se projekt uspješno dovede do kraja. Projekt nam je donio veliko iskustvo za daljnji rad. Prikazao je stvarnu sliku grupnog rada na projektu, potaknuo nas je na učenje i upoznavanje sa novim tehnologijama - dotada nepoznatim radnim okvirima i programskim jezicima te ispitivanjem programske potpore. Upoznali smo koliko je važna dokumentacija u životnom ciklusu projekta. Bez dokumentacije projekt ne bilo moguće dovesti do uspješnog kraja. Dokumentacija nam je ponajprije olakšala i ubrzala rad na projektu.

Nakon ovakvog iskustva spremni smo se okušati u novim projektima i izazovima koji nas očekuju u budućem radu i karijeri nakon fakulteta.

Popis literature

Kontinuirano osvježavanje

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. LaTeX, <https://www.latex-project.org/>
8. TeXstudio <https://www.texstudio.org/>
9. TeX Live <https://www.tug.org/texlive/>
10. GitLab <https://about.gitlab.com/>
11. PostgreSQL <https://www.postgresql.org/>
12. JavaScript Tutorial <https://www.w3schools.com/js/>
13. React <https://reactjs.org/>
14. PHP <https://www.w3schools.com/php/>
15. Symfony <https://symfony.com/>
16. Visual Paradigm Online <https://online.visual-paradigm.com/>

Indeks slika i dijagrama

3.1	UML obrazaca korištenja UC1 do UC3	17
3.2	UML obrazaca korištenja UC4 do UC6 te UC9	18
3.3	UML obrazaca korištenja UC7 do UC8 te od UC10 do UC12	19
3.4	Sekvencijski dijagram UC5	20
3.5	Sekvencijski dijagram UC6	21
3.6	Sekvencijski dijagram UC8	22
4.1	Arhitektura sustava	24
4.2	E-R dijagram baze podataka	29
4.3	Dijagram razreda	31
4.4	Dijagram stanja	33
4.5	Dijagram stanja	34
4.6	Dijagram komponenti web aplikacije	36
5.1	Selenium - test 1	39
5.2	Ispitni slučaj - Uspješna prijava	39
5.3	Ispitni slučaj - Neuspješna prijava	39
5.4	Selenium - test 2	40
5.5	Ispitni slučaj - Uspješna registracija	40
5.6	Ispitni slučaj - Neuspješna registracija	40
5.7	Selenium - test 3	41
5.8	Ispitni slučaj - Uspješan unos tretmana	41
5.9	Ispitni slučaj - Neuspješan unos tretmana	41
5.10	Selenium - test 4	42
5.11	Konstante korištene u testovima	42
5.12	Rezultati ispitivanja sustava	43
5.13	Dijagram razmještaja web aplikacije	45
5.14	npm run naredba	46
5.15	Drag and drop na Netlify poslužitelj	47
5.16	Postavke za poslužitelj (1)	48
5.17	Postavke za poslužitelj (2)	48

5.18 Pokretanje backend poslužitelja Symfony	49
6.1 Commitovi na master granu	60
6.2 Commitovi na backend granu	60
6.3 Commitovi na frontend granu	61
6.4 Commitovi na devdoc granu	61
6.5 Commitovi na develop granu	61
6.6 Commitovi na user-controller granu	62
6.7 Commitovi na lukaUC5i12 granu	62
6.8 Commitovi na Patrik-Frontend granu	62

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

Kontinuirano osvježavanje

1. sastanak

- Datum: 20. listopada 2022.
- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, F.Hruza, M.Bakač, F.Ogrinšak
- Teme sastanka:
 - Upoznavanje članova tima
 - Raspodjela u članova u timove prema vrsti posla (frontend, backend, dokumentacija)
 - Odabir tehnologija i alata koji će se koristiti na projektu

2. sastanak

- Datum: 10. studenoga 2022.
- Prisustvovali: L.Kurtin, P.Pašić, F.Hruza, M.Bakač, F.Ogrinšak
- Teme sastanka:
 - Razrađeni daljnji planovi izrade dokumentacije
 - Raspodjela zadataka za izradu frontend dijela aplikacije
 - Inicijalizirana grana na GitLab-u za frontend

3. sastanak

- Datum: 17. studenoga 2022.
- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, F.Hruza, F.Ogrinšak, M.J.Kocijan
- Teme sastanka:
 - Pregled napravljenog posla
 - Diskusija daljnjeg razvoja aplikacije
 - Diskusija potrebnih preinaka dokumentacije kako bi bila spremna za rok predaje 1.revizije

4. sastanak

- Datum: 7. prosinca 2022.

- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, F.Hruza, F.Ogrinšak, M.J.Kocijan
- Teme sastanka:
 - Raspodjela zadataka
 - Definiranje rokova zadataka
 - Razrada ciljeva do sljedećeg sastanka

5. sastanak

- Datum: 12. prosinca 2022.
- Prisustvovali: L.Kurtin, P.Pašić, F.Ogrinšak
- Teme sastanka:
 - Diskusija napravljenog frontend-a
 - Rješavanje problema frontend-a

6. sastanak

- Datum: 14. prosinca 2022.
- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, F.Hruza, M.Bakač, M.J.Kocijan
- Teme sastanka:
 - Diskusija nejasnoća
 - Diskusija napravljenog
 - Određivanje preostalih zadataka
 - Usklađivanje dijagrama sa trenutnim stanjem i funkcionalnostima aplikacije

7. sastanak

- Datum: 19. prosinca 2022.
- Prisustvovali: L.Kurtin, P.Pašić, F.Ogrinšak
- Teme sastanka:
 - Rješavanje problema frontend-a

8. sastanak

- Datum: 20. prosinca 2022.
- Prisustvovali: L.Kurtin, P.Pašić, F.Ogrinšak
- Teme sastanka:
 - Rješavanje problema frontend-a

9. sastanak

- Datum: 21. prosinca 2022.
- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, F.Hruza, M.Bakač, M.J.Kocijan, F.Ogrinšak
- Teme sastanka:

- Rješavanje problema frontend-a
- Manje promjene backend-a
- Diskusija daljnjeg razvoja

10. sastanak

- Datum: 2. siječnja 2023.
- Prisustvovali: D.Đurinić, L.Kurtin, P.Pašić, M.Bakač, M.J.Kocijan, F.Ogrinšak
- Teme sastanka:
 - Pregled napravljenog
 - Raspoređivanje novih zadataka
 - Diskusija daljnjeg razvoja

11. sastanak

- Datum: 3. siječnja 2022.
- Prisustvovali: P.Pašić, F.Hruza, F.Ogrinšak
- Teme sastanka:
 - Planiranje zadnjeg tjedna
 - Diskusija dokumentacije
 - Diskusija frontend-a

12. sastanak

- Datum: 10. siječnja 2022.
- Prisustvovali: D.Đurinić, M.Bakač, M.J.Kocijan, F.Ogrinšak
- Teme sastanka:
 - Prepravljanje tablica baze
 - Diskusija dokumentacije
 - Zadani novi zadatci

13. sastanak

- Datum: 10. siječnja 2022.
- Prisustvovali: L.Kurtin, P.Pašić, M.J.Kocijan, F.Ogrinšak
- Teme sastanka:
 - Diskusija posljednjih detalja
 - Diskusija dokumentacije

Tablica aktivnosti

Kontinuirano osvježavanje

	Fran Ogrinšak	Dominik Đurinić	Luka Kurtin	Patrik Pašić	Fran Hruza	Martin Bakač	Martin Josip Kocijan
Upravljanje projektom	10				2		
Opis projektnog zadatka		2					
Funkcionalni zahtjevi		3					
Opis pojedinih obrazaca					4		
Dijagram obrazaca					2		
Sekvencijski dijagrami					4		
Opis ostalih zahtjeva					1		
Arhitektura i dizajn sustava		3					
Baza podataka		2					
Dijagram razreda					4		
Dijagram stanja					4		
Dijagram aktivnosti					3		
Dijagram komponenti		3					
Korištene tehnologije i alati					2		
Ispitivanje programskog rješenja	6	2	2	3		5	3
Dijagram razmještaja		2					
Upute za puštanje u pogon	3				1	5	5
Dnevnik sastajanja		2			2		

Nastavljeno na idućoj stranici

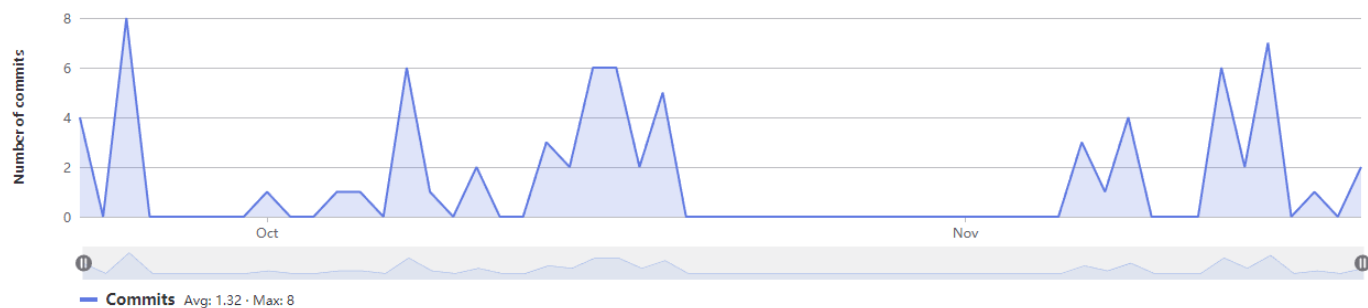
Nastavljeno od prethodne stranice

	Fran Ogrinšak	Dominik Đurinić	Luka Kurtin	Patrik Pašić	Fran Hruza	Martin Bakač	Martin Josip Kocijan
Zaključak i budući rad		1					
Popis literature					1		
<i>Frontend</i>	35		40	40		3	4
<i>back end</i>	5		2	3		40	40

Dijagrami pregleda promjena

Commits to master

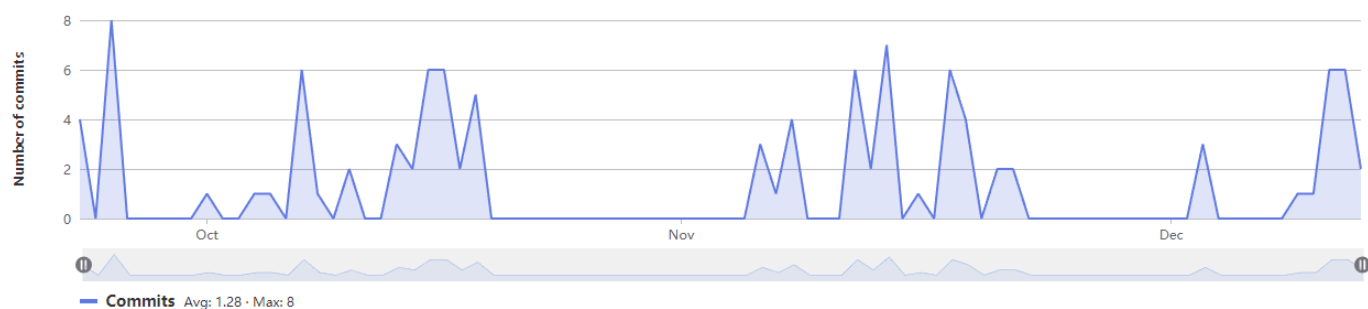
Excluding merge commits. Limited to 6,000 commits.



Slika 6.1: Commitovi na master granu

Commits to backend

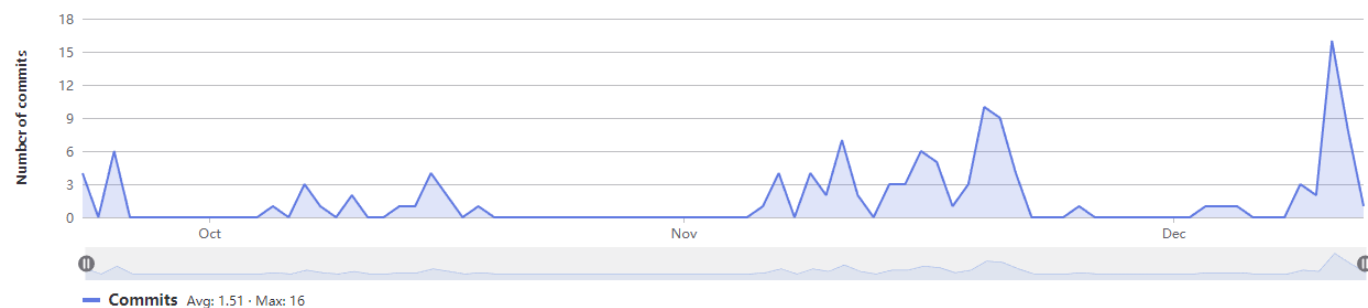
Excluding merge commits. Limited to 6,000 commits.



Slika 6.2: Commitovi na backend granu

Commits to frontend

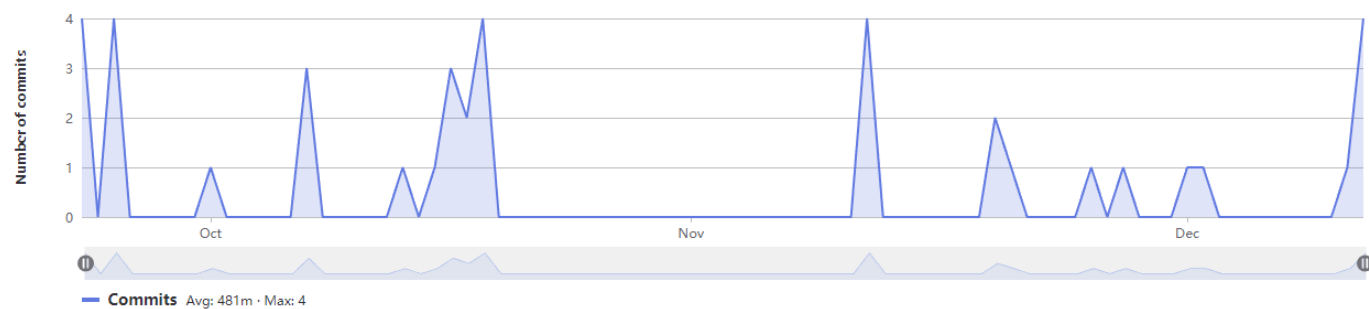
Excluding merge commits. Limited to 6,000 commits.



Slika 6.3: Commitovi na frontend granu

Commits to devdoc

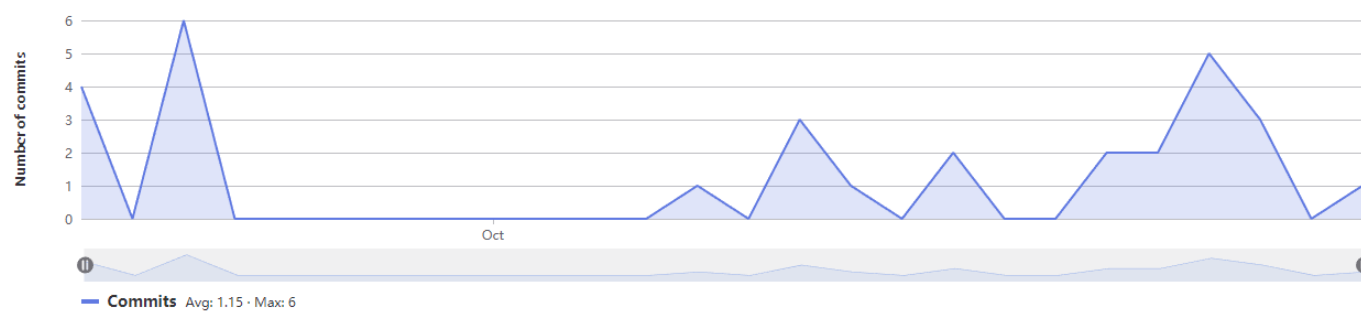
Excluding merge commits. Limited to 6,000 commits.



Slika 6.4: Commitovi na devdoc granu

Commits to develop

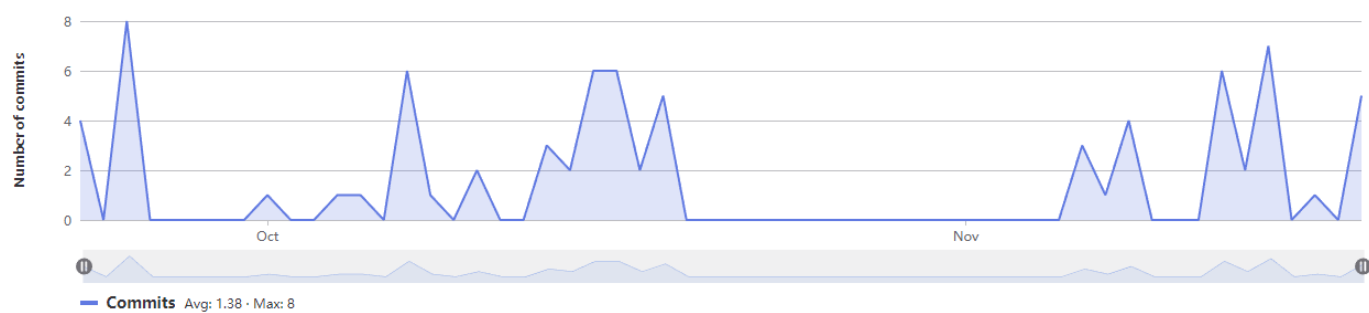
Excluding merge commits. Limited to 6,000 commits.



Slika 6.5: Commitovi na develop granu

Commits to user-controller

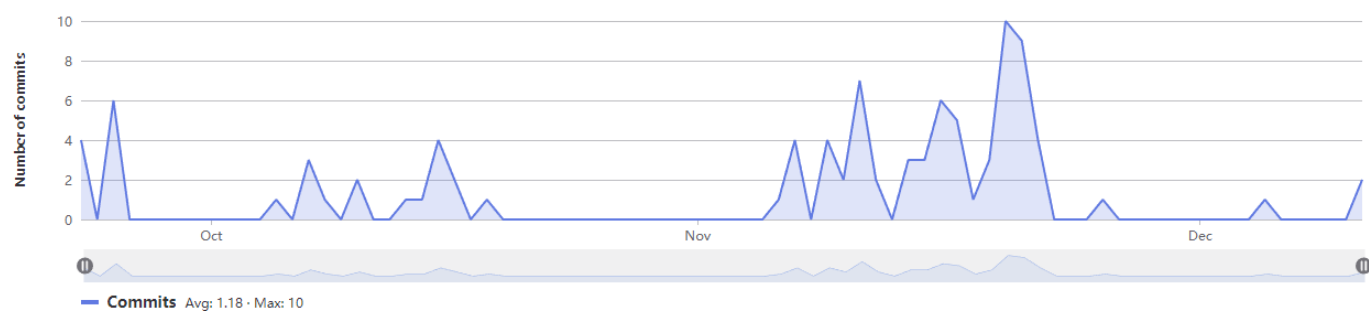
Excluding merge commits. Limited to 6,000 commits.



Slika 6.6: Commitovi na user-controller granu

Commits to lukaUC5i12

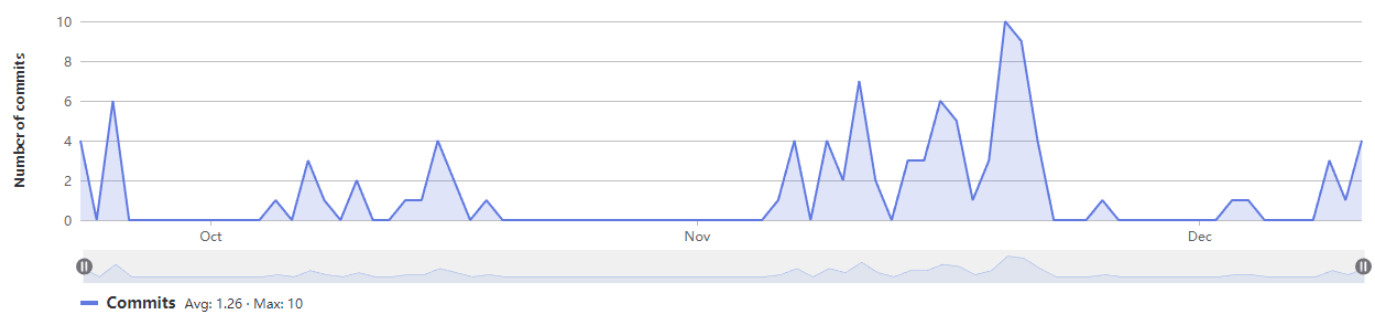
Excluding merge commits. Limited to 6,000 commits.



Slika 6.7: Commitovi na lukaUC5i12 granu

Commits to Patrik-Frontend

Excluding merge commits. Limited to 6,000 commits.



Slika 6.8: Commitovi na Patrik-Frontend granu