

Programsko inženjerstvo

Ak. god. 2021./2022.

True Blood

Dokumentacija, Rev. 1

Grupa: *Illidimus Digitus*

Voditelj: *David Kerman*

Datum predaje: *19.11.2021.*

Nastavnik: *Ivan Lovrić*

Sadržaj

1 Dnevnik promjena dokumentacije	2
2 Opis projektnog zadatka	4
3 Specifikacija programske potpore	8
3.1 Funkcionalni zahtjevi	8
3.1.1 Obrasci uporabe	10
3.1.2 Sekvencijski dijagrami	24
3.2 Ostali zahtjevi	28
4 Arhitektura i dizajn sustava	29
4.1 Baza podataka	33
4.1.1 Opis tablica	34
4.1.2 Dijagram baze podataka	38
4.2 Dijagram razreda	39
4.3 Dijagram stanja	45
4.4 Dijagram aktivnosti	46
4.5 Dijagram komponenti	48
5 Implementacija i korisničko sučelje	49
5.1 Korištene tehnologije i alati	49
5.2 Ispitivanje programskog rješenja	50
5.2.1 Ispitivanje komponenti	50
5.2.2 Ispitivanje sustava	55
5.3 Dijagram razmještaja	61
5.4 Upute za puštanje u pogon	62
5.4.1 Instalacija poslužitelja baze podataka	62
5.4.2 Instalacija Node.js runtime-a	62
5.4.3 Instalacija Jave	63
5.4.4 Instalacija Maven alata i postavljanje PATH varijable okruženja	63

5.4.5	Podešavanje i pokretanje backend dio aplikacije te inicijalno punjenje baze	68
5.4.6	Podešavanje i pokretanje frontend dijela aplikacije	72
5.4.7	Napomene	73
6	Zaključak i budući rad	74
7	Popis literature	76
	Popis literature	76
	Indeks slika i dijagrama	77
	Dodatak: Prikaz aktivnosti grupe	78

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Kerman	28.10.2021.
0.2	Dodani funkcionalni zahtjevi	Kerman, Jurinić	28.10.2021.
0.3	Dodani opisi UC11 - UC19	Kerman	25.10.2021.
0.4	Dodani opisi UC1 - UC10	Jurinić	29.10.2021
0.5	Sekvencijski dijagrami	Hudiček	31.10.2021
0.6	Opis projektnog zadatka	Šlezak	01.11.2021
0.7	Dodan opis baze	Vugrinec	03.11.2021
0.6	Dodani dijagrami obrazaca uporabe	Kerman	03.11.2021
0.7	Dodani nefunkcionalni zahtjevi	Okreša	04.11.2021
0.8	Dodani opisi varijabli u opisu baze	Vugrinec	05.11.2021
0.9	Definirana arhitektura sustava	Kerman	10.11.2021.
0.10	Osvježeni obrasci upotrebe	Kerman	12.11.2021.
0.11	Popravljen opis baze	Vugrinec	15.11.2021.
0.12	Popravljen pravopis	Kerman	15.11.2021.
0.13	Popravljeni obrasci uporabe	Kerman, Jurinić	17.11.2021.
0.14	Dodani opisi i dijagrami razreda	Kerman, Hudiček	17.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Kerman	19.11.2021.
1.1	Dodani opisi dijagrama (stanja, aktivnosti, komponenti, razmještaja)	Kerman	12.12.2021.
1.2	Dodani dijagrami (stanja, aktivnosti, komponenti, razmještaja)	Kerman	13.12.2021.
1.3	Popravljeni opisi obrazaca uporabe	Kerman	28.12.2021.
1.4	Popravljeni dijagrami obrazaca uporabe i aktivnosti	Kerman	08.01.2022.
1.5	Dodani unit testovi	Kerman	10.01.2022.
1.6	Dodane korištene tehnologije i alati i Zaključak	Kerman	11.01.2022.
1.7	Dodani selenium testovi	Pardon	12.01.2022.
1.8	Popravljen dijagram baze	Vugrinec	12.01.2022.
1.9	Popravljeni nefunkcionalni zahtjevi	Okreša	13.01.2022.
1.10	Popravljeni obrasci uporabe	Jurinić	13.01.2022.
1.11	Popravljeni sekvenijski dijagram	Hudiček	13.01.2022.
1.12	Ažuriran dijagram razreda	Hudiček	13.01.2022.
1.13	Dodano poglavlje "Puštanje u pogon"	Šlezak	13.01.2022.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje responzivne web aplikacije "True Blood" koja omogućuje prikupljanje i objavljivanje podataka o prikupljenim dozama darivane krvi te općenito vođenje evidencije podataka za banke krvi. Ovaj projekt omogućio bi smanjenje vremena i znatno olakšanje obavljanje administracijskih poslova pri samoj djelatnosti prikupljanja krvi. Vođenje evidencije 'klasičnim' načinom u današnje vrijeme je skupo, sporo i neefikasno, a ovim projektom nestaje potreba za standardnom hrpom papira, što ujedno i smanjuje kompliciranost, potrebu za dodatnom radnom snagom te dodatne troškove, a da ne govorimo o mogućoj dodatnoj redundanciji podataka i njihovoj mogućoj većoj opsežnosti.

Neregistriranom/neprijavljenom korisniku se na javim web stranicama prikazuje trenutno stanje zaliha različitih krvnih grupa te mogućnost logina i registracije. U sustavu postoje 3 vrste korisnika:

- administrator
- djelatnik banke
- donor

Djelatnik banke i donor na svoju email adresu dobivaju link za aktivaciju korisničkog računa, privremenu lozinku te djelatnici dobivaju svoje korisničko ime, a donor svoj donorId koji će koristiti kao korisničko ime. Prilikom aktivacije korisničkog računa korisnik odabire lozinku koju će koristiti. Administrator sustava administrira korisničke račune. On kreira nove korisničke račune za ulogu djelatnika banke te može u bilo kojem trenutku deaktivirati bilo koji korisnički račun. Administrator isto tako definira gornju i donju granicu optimalne količine krvi za svaku krvnu grupu, kako bi sustav dojavio upozorenja u slučaju prekoračenja gornje ili donje granice. Djelatnik banke krvi evidentira podatke o donoru kada potencijalni donor pristupa darivanju krvi. Ukoliko donor još nije evidentiran u sustavu, djelatnik banke kreira njegov korisnički profil te popunjava sve potrebne podatke:

- matični podaci
- kontakt podaci

- zdravstveni podaci

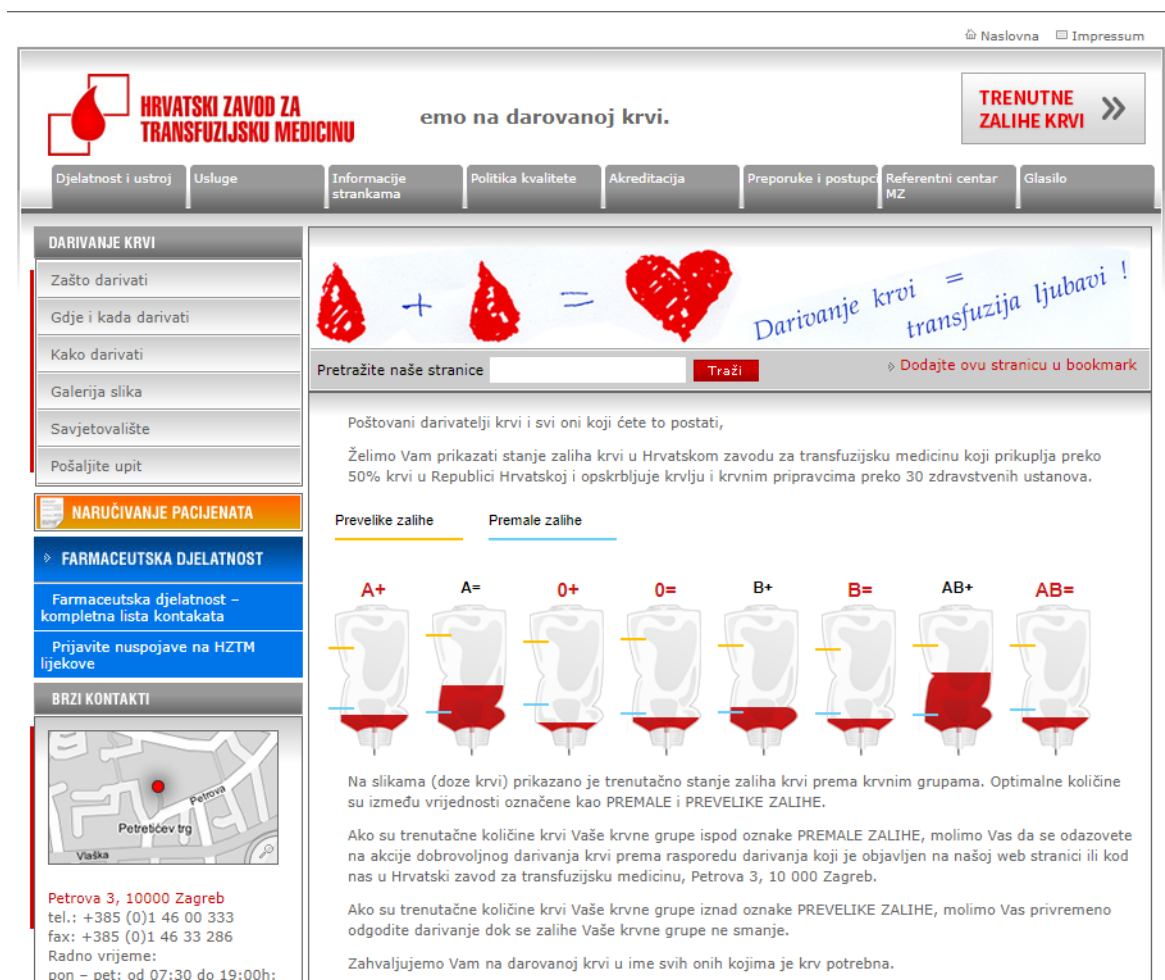
Ukoliko je donor već koristio usluge ustanove, povlače se zadnji aktualni podaci koje djelatnik banke po potrebi nadopunjuje. Djelatnik banke u sustav evidentira svaki pokušaj doniranja. Prije nego donor pristupi darivanju krvi, djelatnik banke provjerava njegovo zdravstveno stanje, te prema tome može prihvatiti te privremeno ili trajno odbiti donora. Djelatnik banke evidentira uspješno doniranje krvi, ali i potrošnju (tj. slanje određenog broja jedinica krvi van banke). Time se povećava tj. smanjuje zaliha određene krvne grupe što je odmah vidljivo na javim web stranicama, a ukoliko se prekorači gornja ili donja granica zalihe za neku krvnu grupu, djelatnici dobivaju notifikaciju putem emaila. Donori imaju mogućnost sami se registrirati na web stranici te ažurirati svoje matične i kontakt podatke te pregledavati zapisane zdravstvene podatke i povijest svojih (uspješnih i odbijenih) doniranja. Svakom evidencijom uspješnog darivanja krvi, donor na svoj email dobiva poruku s potvrdom o pristupanju darivanju krvi u PDF formatu koju može i sam podići naknadno iz aplikacije. Prilikom svakog logiranja donora u sustav (koji nema trajnu zabranu darivanja krvi), aplikacija će donoru prikazati poruku u ovisnosti o trenutnom stanju zaliha krvi za njegovu krvnu grupu:

- stanje zaliha je ispod optimalne granice
- stanje zaliha je optimalno
- stanje zaliha je iznad gornje optimalne granice

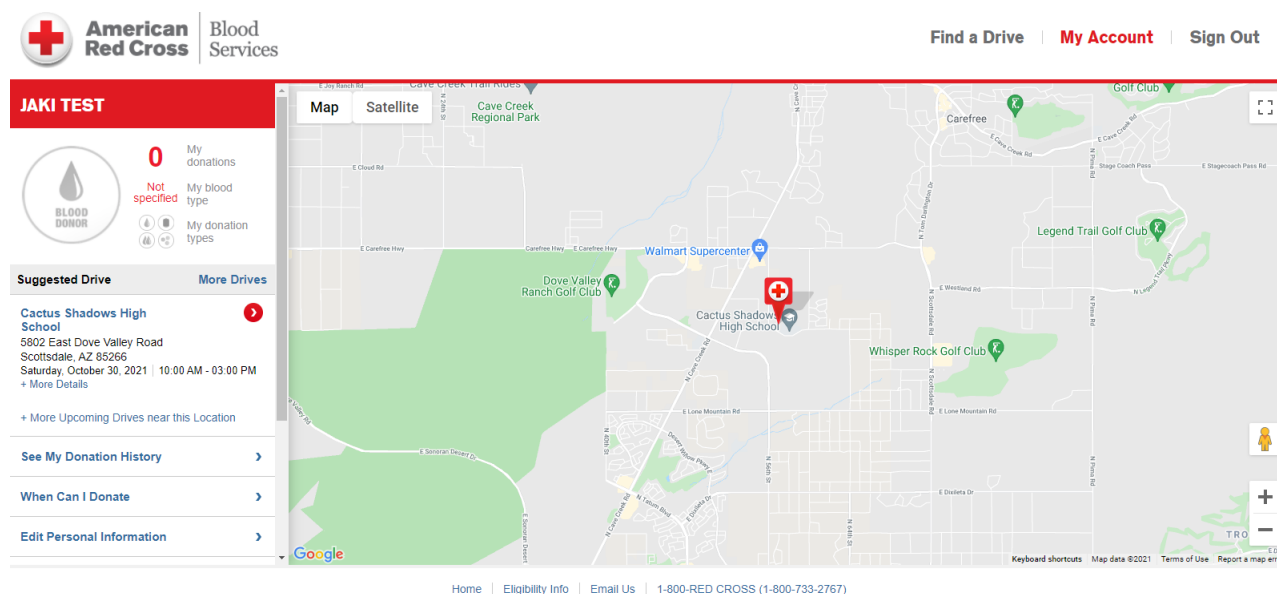
Donori dobivaju notifikacije od sustava nakon što istekne dopušteni period od zadnjeg darivanja i ukoliko zaliha njihove krvne grupe padne ispod minimalne granice.

Ovo rješenje bi mogle koristiti jednako manje ali i veće banke krvi koje još nemaju informacijski sustav za prikupljanje i objavljivanje podataka o prikupljenim dozama krvi. Pošto se radi o web aplikaciji, ona se lako može implementirati u postojeći sustav web stranica koju pojedini potencijalni korisnik rješenja ima, te se, ukoliko je to zbilja potrebno, i stil same web aplikacije može prilagoditi da korespondira sa stilom postojećih web stranica. Isto tako, ukoliko potencijalni korisnik već posjeduje neku vlastitu bazu podataka koju koristi za neku funkcionalnost logiranja i/ili evidenciju nekih relativnih podataka te bi ju htio nastaviti koristiti, rješenje je moguće prilagoditi da se koristi zahtijevanim podacima.

Trenutno u hrvatskoj postoje implementirani dijelovi ovih rješenja, točnije, postoje javne stranice gdje se prikazuje trenutno stanje zaliha različitih krvnih grupa (npr. slika 2.1). Ako pogledamo malo šire po svijetu, možemo uočiti da ima očito i nekih razrađenih sistema za evidenciju sa loginom donora (npr. slika 2.2), no sva ta rješenja nemaju kompaktno razrađen sustav kod kojih bi prijašnji donori dobivali notifikacije s obzirom na stanje banke krvi i pojedinačnih krvnih grupa, koje je javno i na jednostavan (grafički) način dostupno svima.



Slika 2.1: Primjer javno dostupnih podataka o stanju banke krvi Hrvatskog zavoda za transfuzijsku medicinu



Slika 2.2: Primjer web sučelja logiranog korisnika Američkog crvenog križa

Ovo rješenje ima i nešto mjesta za kasniju nadogradnju. Jedna od mogućnosti je ugradnja podrške za notifikacije putem SMS poruka. Moguće je kasnije dodavanje i nešto poput nagradnog sistema za donore gdje bi donori svakim darivanjem krvi skupljali bodove koje bi mogli iskoristiti npr. za neke popuste kod nekih sponzora, besplatan ručak, ulaznicu u kino itd. Naravno, kasnijih nadogradnji (i samih prilagodbi) može biti još, ovisno o željama i kasnijim potrebama korisnika rješenja, ukoliko su izvediva u sklopu ovog projekta.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Korisnici
2. Javnost
3. Razvojni tim
4. Naručitelj

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik(inicijator) može:
 - (a) pregledati trenutno stanje zaliha
 - (b) se registrirati u sustav, stvoriti korisnički račun za koji su mu potrebni matični i kontakt podaci
2. Donor (inicijator) može:
 - (a) pregledavati i mijenjati osobne podatke
 - (b) pregledavati povijest svojih doniranja
 - (c) iz aplikacije dobiti PDF potvrdu
 - (d) pregledati poruku u ovisnosti o trenutnom stanju zaliha krvi
 - (e) aktivirati račun aktivacijskim linkom i odabrati lozinku
3. Djelatnik banke (inicijator) može:
 - (a) kreirati korisnički profil donora
 - (b) evidentirati svaki pokušaj doniranja (uspješan / neuspješan)
 - (c) evidentirati privremeno ili trajno odbijanje
 - (d) evidentirati potrošnju krvi
 - (e) aktivirati račun aktivacijskim linkom i odabrati lozinku
 - (f) vidjeti popis registriranih donora

4. Administrator (inicijator) može:

- (a) definirati gornju i donju granicu optimalne količine krvi
- (b) kreirati nove korisničke račune za ulogu djelatnika banke
- (c) deaktivirati korisnički račun djelatnika banke ili donora
- (d) vidjeti popis registriranih svih korisnika i njihovih osobnih podataka

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje trenutno stanje količine krvi, te donju i gornju granicu optimalne količine krvi
- (c) pohranjuje sve podatke o donacijama krvi

6. Sustav za automatske poslove(inicijator) može:

- (a) slati email notifikacije donoru nakon 3 mjeseca od uspješnog darivanja ako je muškarac ili nakon 4 mjeseca ako je žensko

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregledaj količine krvi

- **Glavni sudionik:** Neregistrirani/neprijavljeni korisnik
- **Cilj:** Prikazati trenutnu količinu krvi u banci
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Količine krvi su prikazane otvaranjem aplikacije, te njihove gornje i donje granice

UC2 - Registriraj korisnika

- **Glavni sudionik:** Neregistrirani/Neprijavljeni korisnik, djelatnik banke
- **Cilj:** Stvoriti korisnički račun donora za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** Djelatnik banke prijavljen
- **Opis osnovnog tijeka:**
 1. Neregistriran/neprijavljen korisnik/djelatnik banke odabire opciju za registraciju novog donora
 2. Sustav otvara obrazac za registraciju
 3. Neregistriran/neprijavljen korisnik/djelatnik banke unosi potrebne podatke i odabire gumb za završetak registracije
 4. Sustav validira unesene podatke i sprema ih u bazu podataka
 5. Sustav generira email poruku s podacima o korisničkom računu i aktivacijskim linkom i šalje na mail adresu korisnika
- **Opis mogućih odstupanja:**
 - 2.a Unos podataka u nedozvoljenom formatu ili unos neispravnog e-maila
 1. Korisnik dobiva obavijest o neuspjelom upisu i vraća ga na korak 2 osnovnog tijeka

UC3 - Prijavi se u sustav

- **Glavni sudionik:** Donor
- **Cilj:** Prijava u sustav i dobivanje dodatnih mogućnosti aplikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Donor unosi e-mail i lozinku u login formu te odabire akciju "Prijava"
 2. Sustav validira unesene podatke, provodi autentikaciju i autorizaciju korisnika
 3. Sustav otvara početni ekran aplikacije te korisniku
 4. Pregledavanje ispisane obavijesti na ekranu
- **Opis mogućih odstupanja:**
 - 2.a Neispravan e-mail i/ili lozinka
 1. Sustav obavještava korisnika o neuspjeloj prijavi i vraća ga u korak 1. osnovnog tijeka
 - 3.a Prijeđena je donja optimalna granica ili je prošlo dovoljno vremena od prijašnje donacije
 1. Sustav prikazuje obavijest da ima novih poruka u pretincu

UC4 - Pregledaj osobne podatke

- **Glavni sudionik:** Donor, Djelatnik
- **Cilj:** Prikazati osobne podatke donora
- **Sudionici:** Baza podataka
- **Preduvjet:** Donor/Djelatnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Donor odabire opciju "Osobni podaci" ili Djelatnik odabere prikaz podataka donora
 2. Sustav dohvaća osobne podatke donora iz baze podataka te ih prikazuje

UC5 - Promjeni osobne podatke

- **Glavni sudionik:** Donor, Djelatnik
- **Cilj:** Promjeniti matične i/ili kontakt podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Donor/Djelatnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Donor/Djelatnik odabire opciju "Promijeni osobne podatke" kod prikaza osobnih podataka donora ili Djelatnik odabire opciju "Promjeni osobne podatke" kod prikaza svojih podataka
 2. Sustav otvara obrazac za promjenu osobnih podataka
 3. Donor/Djelatnik mijenja podatke i odabire opciju "Spremi promjene"
 4. Sustav ažurira bazu podataka

UC6 - Pregledaj povijest darivanja

- **Glavni sudionik:** Donor
- **Cilj:** Pregledati svoju povijest darivanja krvi
- **Sudionici:** Baza podataka
- **Preduvjet:** Donor je prijavljen
- **Opis osnovnog tijeka:**
 1. Donor odabire opciju "Povijest donacija"
 2. Sustav dohvaća povijest darivanja iz baze podataka te ju prikazuje

UC7 - Izvadi PDF potvrde

- **Glavni sudionik:** Donor
- **Cilj:** Dobiti PDF potvrdu o doniranju
- **Sudionici:** Baza podataka
- **Preduvjet:** Donor je prijavljen i barem jedanput uspješno darovao krv
- **Opis osnovnog tijeka:**
 1. Donor odabire opciju "Povijest donacija"
 2. Sustav dohvaća povijest darivanja iz baze podataka te ju prikazuje
 3. Donor odabire opciju "Generiraj PDF" koja se nalazi uz svaki uspješan zapis doniranja krvi
 4. Sustav donoru šalje potvrdu na mail

UC8 - Pregledaj poruke

- **Glavni sudionik:** Donor
- **Cilj:** pregledati dospjelu poruku
- **Sudionici:** Baza podataka
- **Preduvjet:** Donor je prijavljen i nema trajnu/privremenu zabranu darivanja krvi
- **Opis osnovnog tijeka:**
 1. Donor odabire opciju "Pretinac"
 2. Sustav dohvaća poruke iz baze podataka i prikazuje ih
 3. Donor pregledava prikazane poruke

UC9 - Smanji količinu krvi

- **Glavni sudionik:** Djelatnik banke
- **Cilj:** slanje određenog broja jedinica krvi u vanjsku instituciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Djelatnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Djelatnik odabire opciju "Evidencija krvi"
 2. Sustav dohvaća podatke za svaku vrstu krvi
 3. Djelatnik odabire krvnu grupu čiju zalihu želi smanjiti
 4. Djelatnik upisuje količinu i naziv institucije u koju je potrebno poslati krv
 5. Djelatnik odabire akciju "Smanji zalihu"

6. Sustav provjerava je li moguće provesti smanjivanje
7. Sustav provodi smanjivanje u bazi podataka
- **Opis mogućih odstupanja:**
 - 4.a Djelatnik je upisao ili preveliku količinu krvi ili manju od 1
 1. Sustav obavještava djelatnika o neispravnom unosu te ga vraća na korak 2 osnovnog tijeka
 - 5.a Smanjivanjem se prešla donja granica optimalne količine krvi
 1. Sustav šalje notifikaciju na mail adresu djelatniku banke i donoru(donoru se također šalje poruka u pretinac poruke)

UC10 - Pregledaj popis donora

- **Glavni sudionik:** Djelatnik banke, administrator
- **Cilj:** Prikazati popis svih registriranih donora
- **Sudionici:** Baza podataka
- **Preduvjet:** Djelatnik je prijavljen ili administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Djelatnik/administrator odabire opciju "Popis donora"
 2. Sustav djelatniku/administratoru dohvaća iz baze popis registriranih donora te ga prikazuje

UC11 -Aktiviraj račun

- **Glavni sudionik:** Neregistrirani/neprijavljeni korisnik, djelatnik banke
- **Cilj:** Aktivirati račun
- **Sudionici:**Baza podataka
- **Preduvjet:**Registracija
- **Opis osnovnog tijeka:**
 1. Neregistrirani/neprijavljeni korisnik ili djelatnik banke nakon klika na link u mailu vodi u aplikaciju
 2. Sustav ponovno renderira login stranicu gdje se traži da se unese nova proizvoljna lozinka
 3. Neregistrirani/neprijavljeni korisnik ili djelatnik banke se prijavljuje odabranom lozinkom

UC12 -Evidentiraj pokušaj doniranja

- **Glavni sudionik:** Djelatnik banke
- **Cilj:** Evidentirati pokušaj doniranja
- **Sudionici:**Baza podataka
- **Preduvjet:**Djelatnik banke je prijavljen i donor je registriran
- **Opis osnovnog tijeka:**
 1. Djelatnik odabere opciju "Obavi donaciju" u prikazu popisa donira
 2. Sustav prikazuje upitnik kojeg ispunjava djelatnik banke
 3. Djelatnik unosi mjesto doniranja
 4. Djelatnik banke sprema promjene
 5. Baza podataka se ažurira
 6. Sustav vraća uspješnost doniranja
 7. Sustav šalje email poruku s PDF potvrdom na mail donora
- **Opis mogućih odstupanja:**
 - 5.a Donacija je uspješna
 1. povećava se količina određene vrste krvi
 - 5.b Prešle su se gornje optimalne granice
 1. šalju se notifikacije na mail djelatnika
 - 6.a
 - Donor je bolovao ili boluje od teških kroničnih bolesti dišnog i probavnog sustava
 - Donor boluje od bolesti srca i krvnih žila, zloćudnih bolesti, bolesti jetre, AIDS-a, šećerne bolesti
 - Donor je ovisnik o alkoholu ili drogama
 - Donor ima spolne odnose s drugim muškarcima
 - Donor često mijenja seksualne partnere
 - Donor je uzimao drogu intravenskim putem
 - Donor je liječen zbog spolno prenosivih bolesti
 - Donor je HIV pozitivan
 - Donor je seksualni partner gore navedenih osoba
 1. Donor se trajno odbija
 2. U bazu se zapisuje trajno odbijanje za određenog donora

6.b

- Tjelesna težina je ispod 55kg
- Tjelesna temperatura je iznad 37°C
- Krvni tlak sistolički je ispod 100, a dijastolički ispod 60
- Puls je ispod 50 ili iznad 100 otkucaja u minuti
- Hemoglobin muškaraca ispod 135g/L, a žena ispod 125g/L
- Donor trenutno uzima antibiotike ili druge lijekove
- Donor je konzumirao alkohol unutar 8 sati
- Donor ima akutno bolesno stanje
- Donor je žena za vrijeme menstruacije ili trudnoće
- Donor je obavljao opasne poslove tog dana
- Donor je muškarac i nije prošlo 3 mjeseca od zadnjeg doniranja
- Donor je žena i nije prošlo 4 mjeseca od zadnjeg doniranja

1. Donor se privremeno odbija
2. U bazu se zapisuje privremeno odbijanje za određenog donora

UC13 -Obriši donora

- **Glavni sudionik:** Administrator
- **Cilj:** deaktivirati kor. račun donora
- **Sudionici:**Baza podataka
- **Preduvjet:**Administrator prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Popis donora"
 2. Sustav prikazuje popis registriranih donora
 3. Administrator odabire opciju "Obriši donora"
 4. Sustav briše donora, ali ostaje zapis o njegovim donacijama
 5. Baza se ažurira

UC14 -Definiraj optimalne granice

- **Glavni sudionik:** Administrator
- **Cilj:** Definirati gornju i donju granicu optimalnih količina krvi za pojedinu grupu
- **Sudionici:**Baza podataka
- **Preduvjet:**Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Definiraj granice"
 2. Sustav prikazuje trenutno određene granice za pojedinu vrstu krvi
 3. Administrator definira granice za pojedinu vrstu krvi
 4. Administrator sprema promjene
 5. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 4.b Prešle su se gornje optimalne granice
 1. Sustav šalje notifikacije na mail djelatnika
 - 4.c Prešle su se donje optimalne granice
 1. Sustav šalje notifikacije na mail djelatnika i donora ako se radi o njegovoj krvnoj grupi (također donoru se šalje poruka u pretinac za poruke)

UC15 -Pregledaj popis djelatnika

- **Glavni sudionik:** Administrator
- **Cilj:** Prikazati popis svih djelatnika banke
- **Sudionici:**Baza podataka
- **Preduvjet:**Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Popis djelatnika"
 2. Sustav prikazuje popis svih djelatnika banke

UC16 -Obriši djelatnika banke

- **Glavni sudionik:** Administrator
- **Cilj:** Deaktivirati kor. račun djelatnika banke
- **Sudionici:**Baza podataka
- **Preduvjet:**Administrator prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Popis djelatnika"
 2. Sustav vraća popis djelatnika banke
 3. Administrator odabire opciju "Obriši djelatnika banke"
 4. Sustav briše djelatnika, ali ostaju zapisi o njegovim provedenim donacija i evidentiranju smanjenja količina krvi
 5. Baza se ažurira

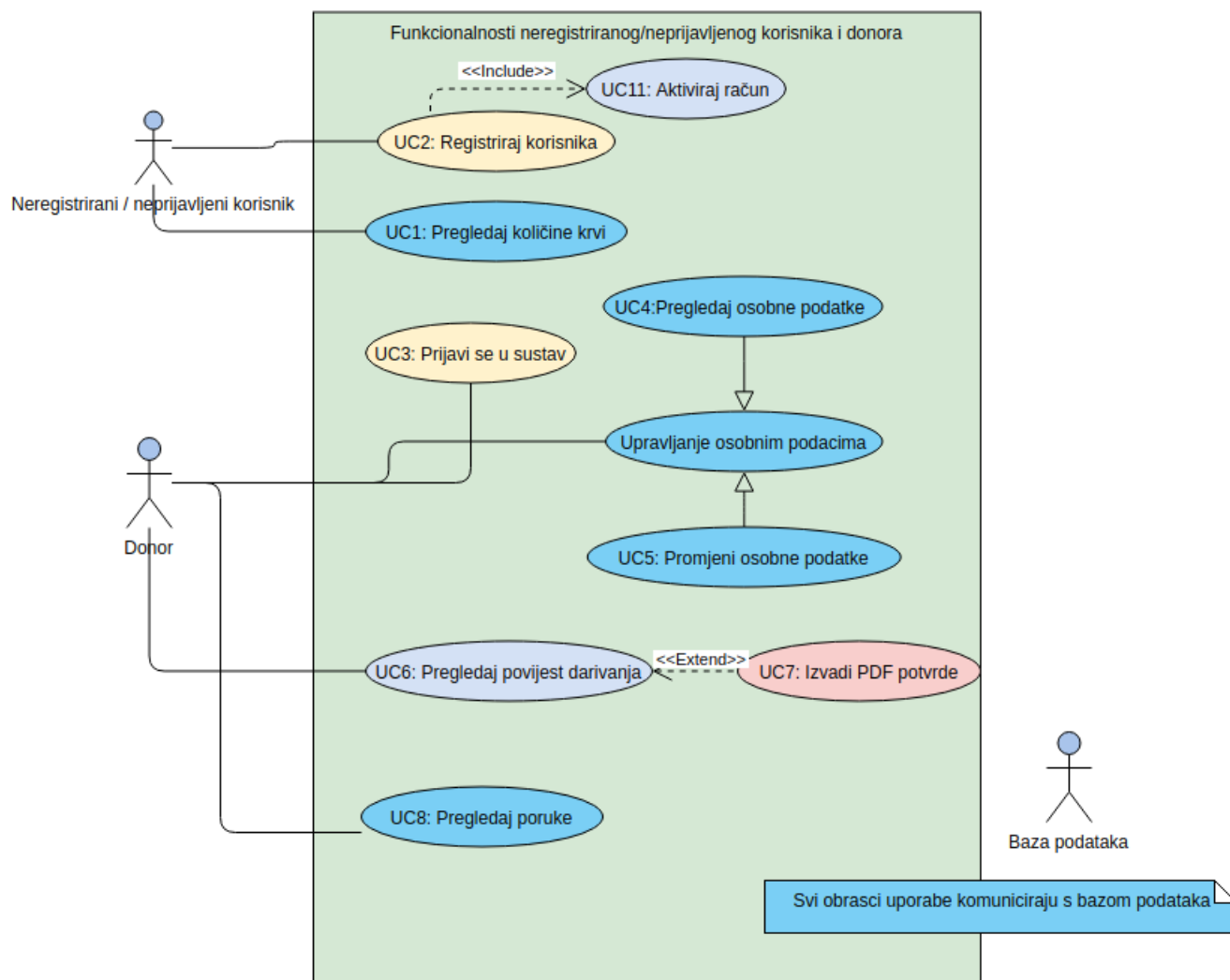
UC17 -Kreiraj kor. računa djelatnika banke

- **Glavni sudionik:** Administrator
- **Cilj:** Kreirati računa djelatnika banke
- **Sudionici:**Baza podataka
- **Preduvjet:**Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju "Dodaj djelatnika"
 2. Sustav prikazuje formu za kreiranje računa koju popunjava administrator
 3. Odabire gumb "Registriraj"
 4. Baza podataka se ažurira
 5. Sustav šalje aktivacijski link na email djelatnika banke
- **Opis mogućih odstupanja:**
 - 2.a Sva polja nisu ispunjena
 1. Sustav obavještava administratora da unese sve podatke

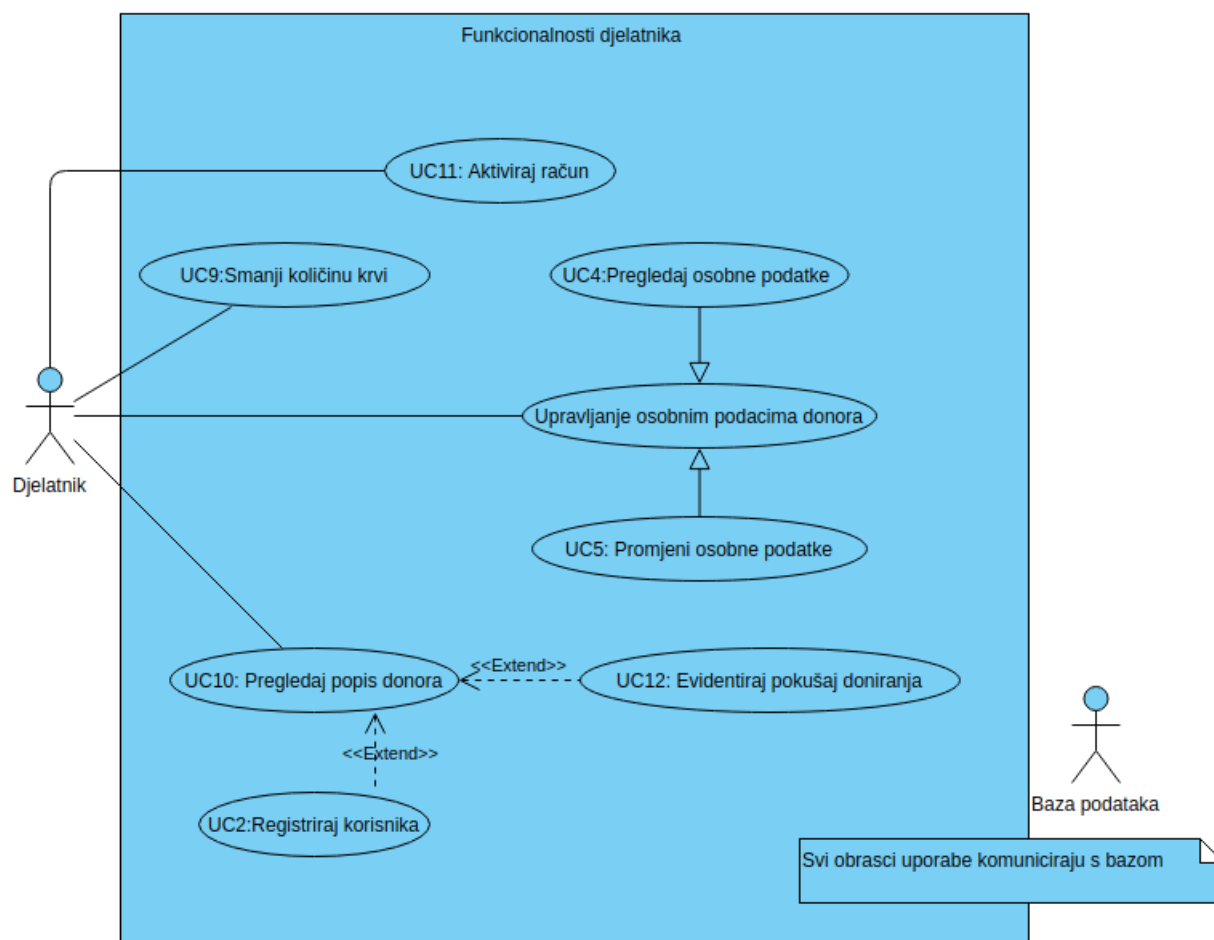
UC18 -Pošalji notifikacije nakon 3/4 mjeseca

- **Glavni sudionik:** Sustav za automatske poslove
- **Cilj:** Poslati notifikaciju donoru
- **Sudionici:**Baza podataka, donor
- **Preduvjet:**Prošlo 3 ili 4 mjeseca od uspješnog darivanja
- **Opis osnovnog tijeka:**
 1. Sustav svakodnevno provjerava uspješna doniranja od prije 3 mjeseca za muškarce i od prije 4 mjeseca za žene
 2. Sustav svim donorima koji zadovoljavaju uvjet šalje email notifikacije da mogu doći ponovno donirati krv
 3. Sustav svim donorima koji zadovoljavaju uvjet šalje poruke u njihov pretinac poruka da mogu doći ponovno donirati krv

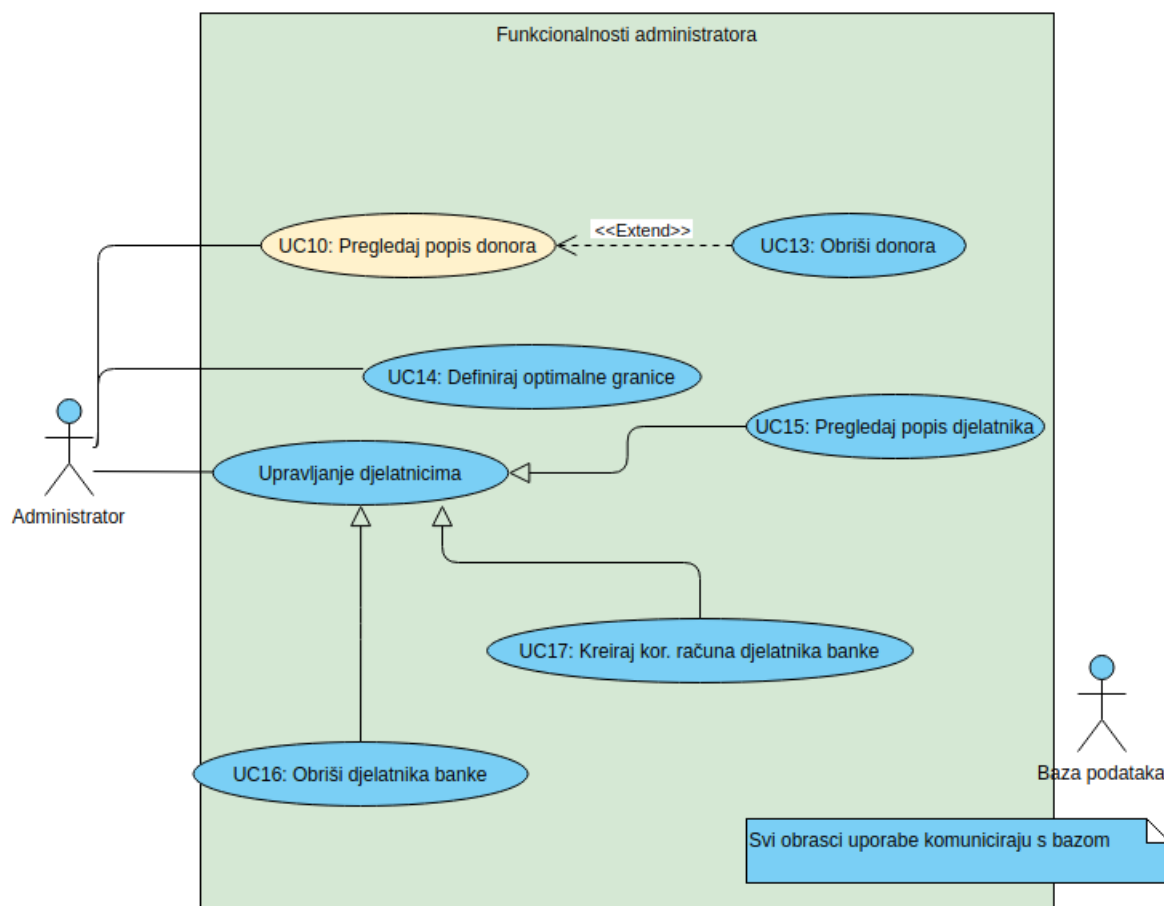
Dijagrami obrazaca uporabe



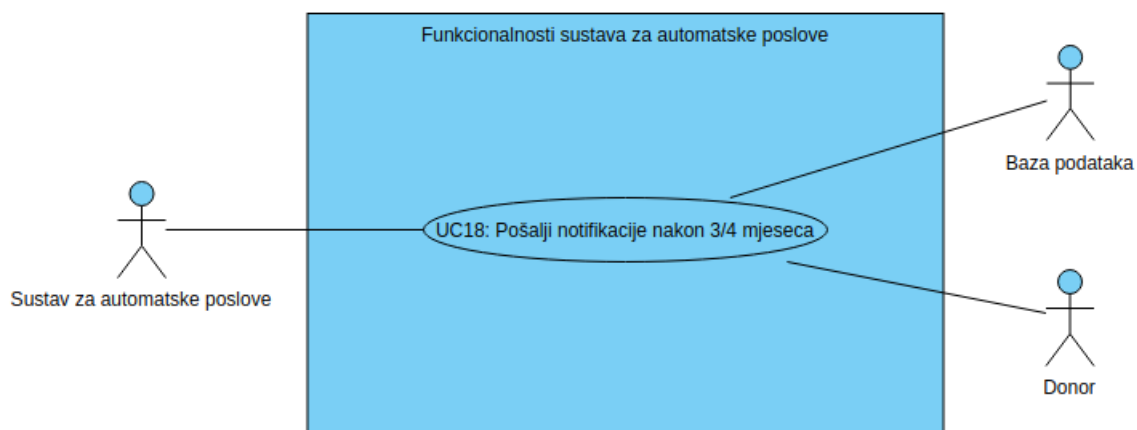
Slika 3.1: Dijagram obrasca uporabe, funkcionalnost neregistriranog / neprijavljenog korisnika i donora



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost djelatnika



Slika 3.3: Dijagram obrasca uporabe, funkcionalnost administratora

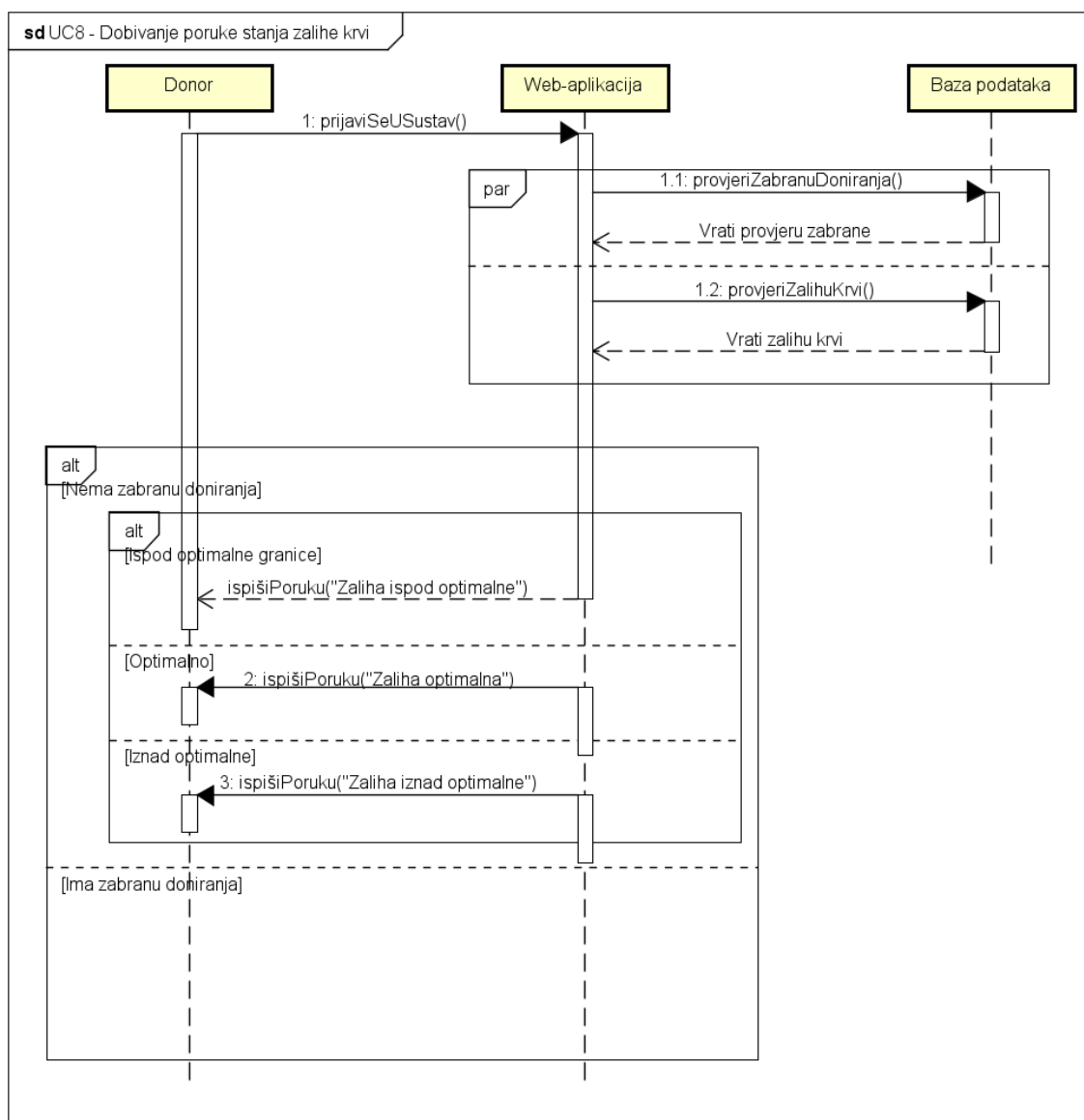


Slika 3.4: Dijagram obrasca uporabe, funkcionalnost sustava za automatske poslove

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC8 - Dobivanje poruke stanja zalihe krvi

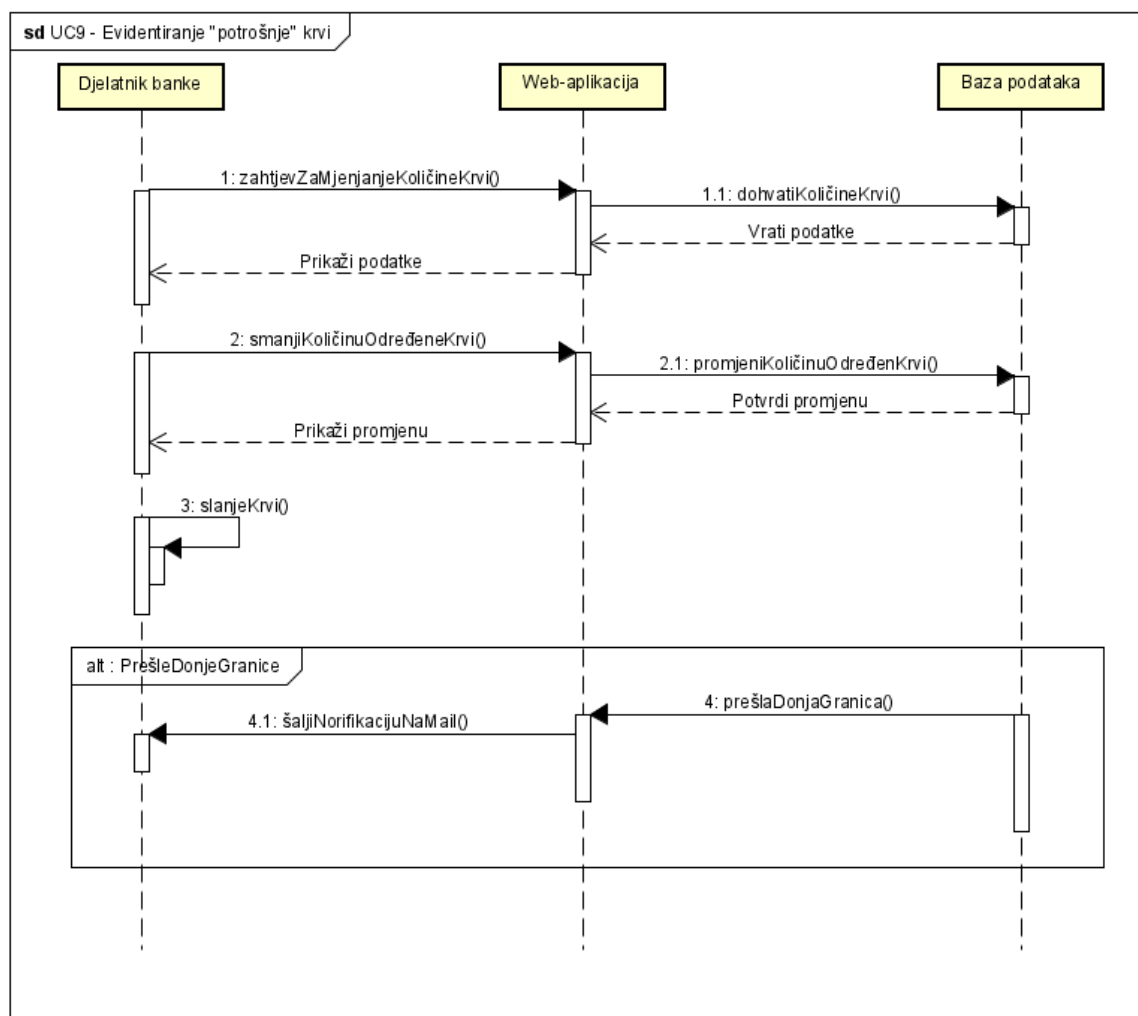
Donor, koji nema trajnu zabranu darivanja krvi, kod svakog spajanja u sustav dobiva poruku u ovisnosti o trenutnom stanju zaliha krvi. Poslužitelj dohvaća podatke iz baze podataka i prema zadanim granicama vraća jednu od tri poruke. Poruke mogu biti: stanje zaliha ispod optimalne granice; stanje zaliha je optimalno; stanje zaliha je iznad gornje optimalne granice.



Slika 3.5: Sekvencijski dijagram za UC8

Obrazac uporabe UC9 - Evidentiranje "potrošnje" krvi

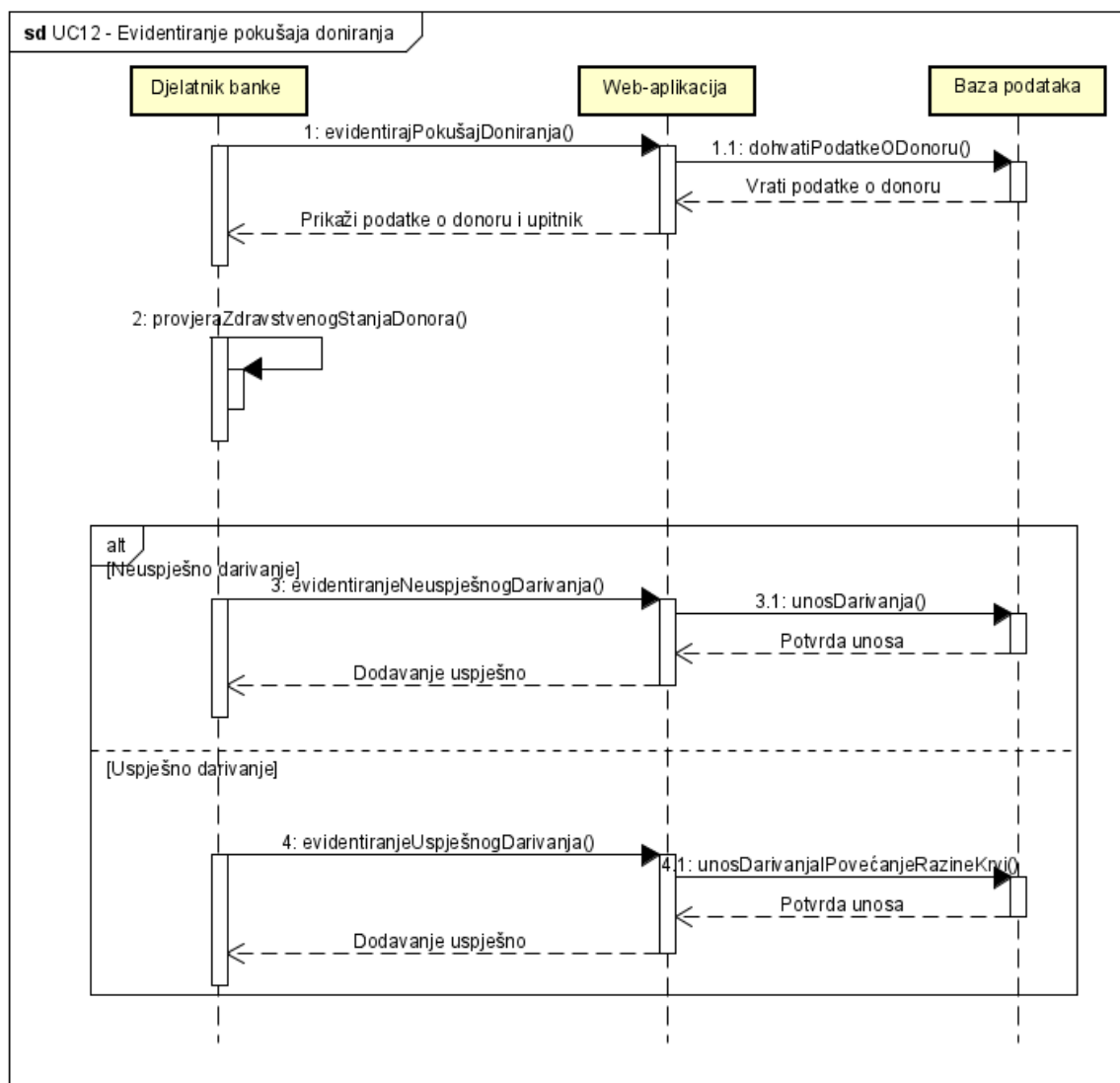
Djelatnik banke može evidentirati "potrošnju" krvi odnosno slanje određenog broja jedinica krvi u vanjsku instituciju. U bazi podataka se smanjuje količina određene vrste krvi i ako se prešla donja granica šalje se mail djelatniku.



Slika 3.6: Sekvencijski dijagram za UC9

Obrazac uporabe UC12 - Evidentiranje pokušaja doniranja

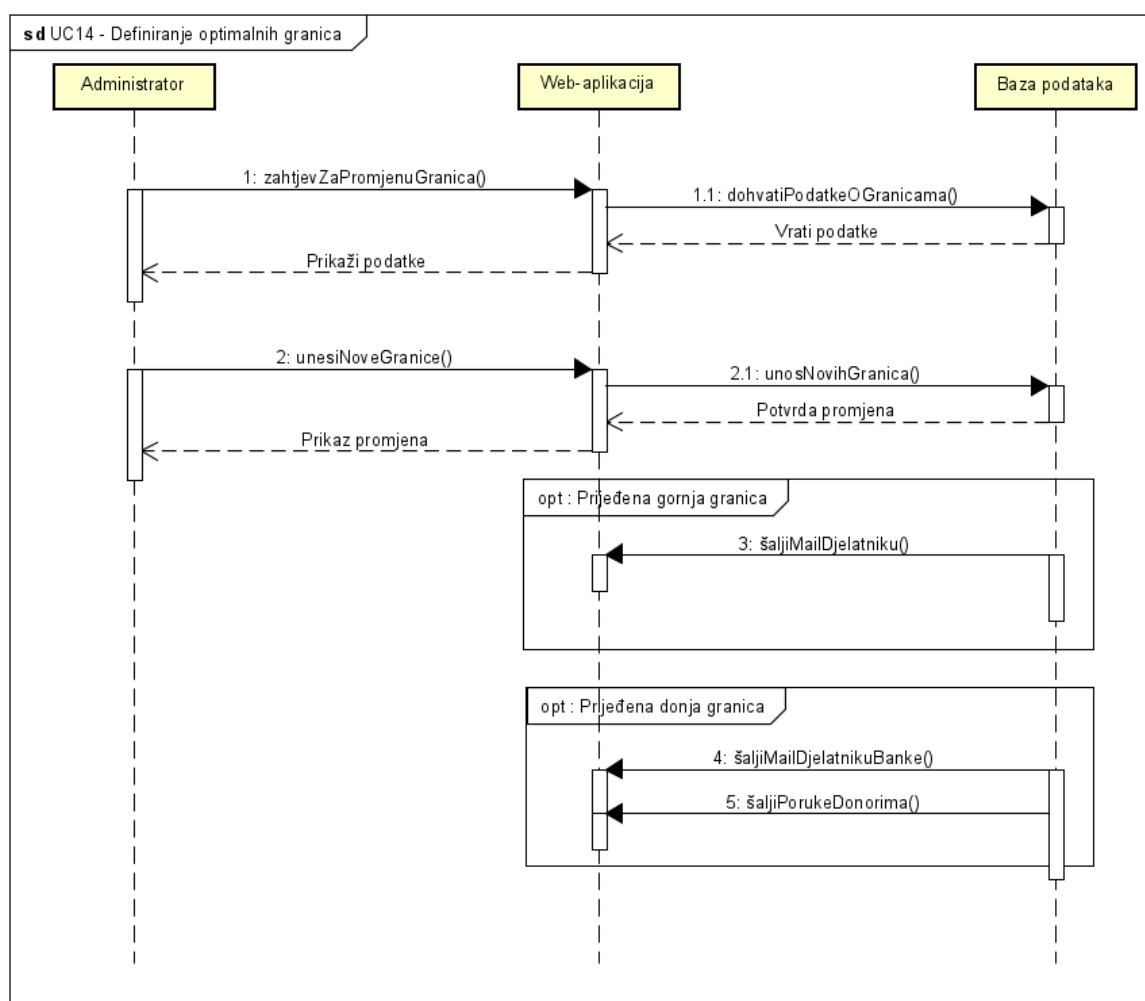
Djelatnik banke evidentira i uspješne i neuspješne pokušaje doniranja. Na temelju zdravstvenog stanja donor može pristupiti doniranju, ali može i biti privremeno ili trajno odbijen. Baza podataka se ažurira, zapisuje se pokušaj doniranja krvi i ako je uspješan poveća se razina određene vrste krvi.



Slika 3.7: Sekvencijski dijagram za UC12

Obrazac uporabe UC14 - Definiranje optimalnih granica

Za svaku krvnu grupu administrator sustava definira gornju i donju granicu optimalne količine. Administratoru se prikazuju trenutne granice za pojedinu vrstu krvi. Zatim on ih može mijenjati i promjene se spremaju i bazu podataka. Ako je prijeđena gornja granica šalje se mail djelatniku. Ako je prijeđena donja granica šalje se mail djelatniku i poruke donorima.



Slika 3.8: Sekvencijski dijagram za UC14

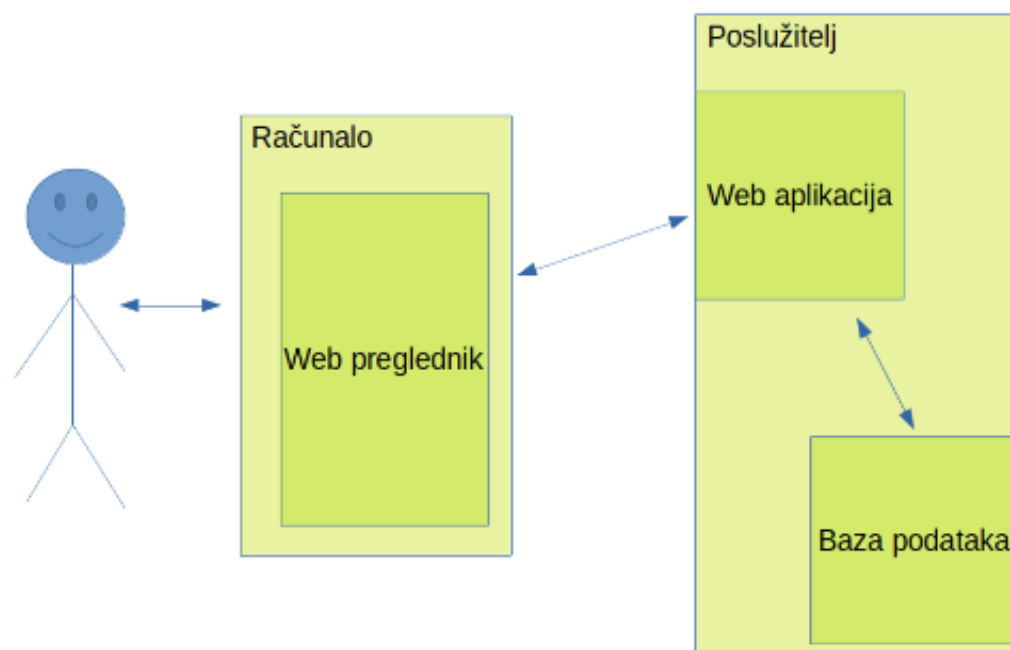
3.2 Ostali zahtjevi

- Sustav treba biti implementiran kao web aplikacija koja je prilagođena različitim veličinama ekrana. (responsive web design)
- Sustavu trebaju moći pristupati tri vrste korisnika (administrator, djelatnik banke i donor). Svaki korisnik se treba ovjeravati korisničkim imenom i lozinkom.
- Donori trebaju imati mogućnost sami kreirati svoj korisnički račun kao i djelatnici banke u slučaju da donor sam nije napravio korisnički račun prije prvog darivanja.
- Administratori trebaju imati mogućnost kreiranja korisničkog računa za djelatnike banke.
- Novi korisnici trebaju dobiti aktivacijski link na svoj mail, donori uz aktivacijski link trebaju dobiti i donorId koji će koristiti kao korisničko ime.
- Prilikom aktivacije korisničkog računa, korisnici trebaju imati mogućnost odabrati svoju lozinku.
- Sustav treba omogućiti rad više korisnika u stvarnom vremenu.
- Neispravno korištenje sustava ne smije narušiti rad sustava.
- Korisničko sučelje treba biti intuitivno, tako da se korisnici mogu koristiti sustavom bez dodatnih uputa.
- Sustav treba podržavati hrvatsku abecedu pri unosu i prikazu tekstualnog sadržaja.
- Veza s bazom podataka mora biti zaštićena.

4. Arhitektura i dizajn sustava

Arhitekturu našeg sustava možemo podijeliti na tri podsustava:

- *Web preglednik*
- *Baza podataka*
- *Web poslužitelj*



Slika 4.1: Arhitektura sustava

Web preglednik je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Korisnik putem web preglednika šalje zahtjeve na obradu poslužitelju.

Baza podataka je zbirka zapisa pohranjenih u računalu na sustavan način, tako da joj se računalni program može obratiti prilikom odgovaranja na problem. Web poslužitelj komunicira s bazom podataka te povlači potrebne zapise iz nje.

Web poslužitelj je srce našeg sustava. Njegova zadaća je komunikacija klijenta s aplikacijom te bazom podataka. Pri komunikaciji koristi HTTP protokol. Korisnik i aplikacija razmjenjuju HTTP zahtjeve (eng. HTTP request) i HTTP odgovore (HTTP response). Radi jednostavnosti, baza podataka je također smještena na poslužitelju.

Korisnik kroz grafičko sučelje, odnosno frontend, šalje zahtjeve na REST pristupne točke backenda. Tada backend procesira zahtjev i ako je potrebno komunicira s bazom podataka. Nakon konstrukcije, backend šalje odgovor frontendu u obliku JSON objekta, a frontend procesira odgovor i promjene prikazuje korisniku u obliku HTML stranice.

Programski jezik kojeg smo odabrali za izradu naše web aplikacije je Java zajedno s Spring radnim okvirom te programski jezik JavaScript u sklopu React-a. Odabrana razvojna okruženja su Eclipse i IntelliJ IDEA. Za aplikaciju je odabrana višeslojna arhitektura temeljena na **MVC** (Model - View - Controller) arhitekturnom stilu te uslužnoj arhitekturi. Podjela slojeva možemo napraviti na idući način:



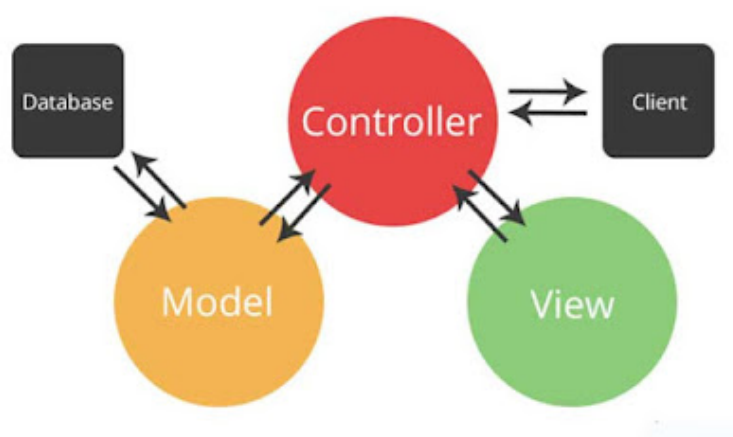
Slika 4.2: Podjela slojeva

- *sloj korisničke strane* - korisničko sučelje
- *sloj nadglednika* - REST nadglednici
- *sloj domene* - model podataka iz domene primjene
- *sloj za pristup podacima* - posrednik između sloja domene i baze podataka
- *sloj baze podataka* - pohrana podataka

Ovakva arhitektura odabrana je zbog poželjnih svojstava MVC arhitekturnog stila i višeslojne arhitekture: razvoj pojedinih slojeva je jednostavniji i u velikom stupnju nezavisan od razvoja drugih slojeva. Također komunikacija frontenda i backenda ostvarena je primjenom REST arhitekturnog stila. Zbog toga su i frontend i backend neovisni o jeziku implementacije, što potiče ponovnu uporabu.

Model-View-Controller se sastoji od:

- **Model** je centralni dio aplikacije, koja obuhvaća promjenljivu (dinamičku) strukturu podataka, direktno upravljanje podacima, logikom i pravilima aplikacije
- **View** je bilo koji izlazni prikaz podataka u korisničkom okruženju, pri čemu se isti podaci mogu prikazati na više načina
- **Controller** ulazne podatke pretvara u komande koje upravljaju modelom ili prikazom podataka u korisničkom okruženju



Slika 4.3: MVC model

4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih entiteta:

- korisnikAplikacije
- uloge
- pokusajDoniranja
- krvnaVrsta
- potrosnjaKrv
- zdravstveniPodaci
- doniranjeZdravljeOdgovori

4.1.1 Opis tablica

korisnikAplikacije • Ovaj entitet predstavlja sve korisnike aplikacije (admin, djelatnik, donor) i diferencira ih pomoću ulogaId što je referenca na tablicu uloga. Za admina i djelatnika postoje atributi : ime, prezime, oib, email dok su ostale vrijednosti null. Kod stvaranja atributa trajnoOdbijanjeDarivanja je false, a razlogOdbijanja je null ,ako se korisniku odbije darivanje zauvijek onda se ti atributi mijenjaju. One-to-many veza s pokusajDoniranja preko korisnikID dvaput, One-to-many veza s potrosnjaKrv preko korisnikId, many-to-one veza s uloge preko ulogaId i many-to-one veza s krvnaVrsta preko krvId.

korisnikAplikacije		
korisnikId	VARCHAR	id pomoću kojeg se korisnik prijavljuje u sustav (jednako donorId za donore)
lozinka	VARCHAR	hash korisničke lozinke
ime	VARCHAR	ime korisnika
prezime	VARCHAR	prezime korisnika
mjestoRođenja	VARCHAR	mjesto rođenja (nullable)
oib	VARCHAR	oib korisnika
adresaStanovanja	VARCHAR	adresa stanovanja (nullable)
mjestoZaposlenja	VARCHAR	firma u kojoj je korisnik zaposlen (nullable)
email	VARCHAR	email na koji korisniku dolaze korisne informacije (nullable)
brojMobitelaPrivatni	VARCHAR	privatni broj korisnika (nullable)
brojMobitelaPoslovni	VARCHAR	broj na koji korisniku dolaze korisne infomacije poslovni(nullable)
datumRođenja	DATE	datum rođenja
krvId	INT	vrsta krvi donora

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

korisnikAplikacije		
trajnoOdbijenoDarivanje	BOOLEAN	true ako je korisniku zabranjeno trajno darivanje, false ako nije
ulogaId	BIGINT	oznaka uloge korisnika
aktivacijskiKljuc	VARCHAR(10)	aktivacijski ključ koji služi za aktivaciju računa

uloge • entitet koji sadrži dva atributa, id za oznaku rednog broja uloge i ulogaName za ime uloge One-to-many veza s korisnikAplikacije preko ulogaId.

Uloge		
ulogaId	BIGINT	označava id uloge
ulogaName	VARCHAR	naziv uloge

pokusajDonacije• -entitet koji predstavlja pokušaj doniranja, sadrži podatke o datumu, mjestu, darivatelju, djelatniku, uspješnosti i razlogu odbijanja ako je uspješnost false. Many-to-one veza preko korisnikIdDjelatnika s korisnikAplikacije, Many-to-one veza preko korisnikIdDonora s korisnikAplikacije, one-to-many veza s doniranjeZdravljeOdgovori reko brDoniranja.

pokusajDonacije		
brDoniranja	BIGINT	redni broj donacije
datum	DATE	datum donacije
mjestoDarivanja	VARCHAR	opisno mjesto donacije
korisnikIdDjelatnika	VARCHAR	identifikator djelatnika
korisnikId	VARCHAR	identifikator donora
uspjeh	BOOLEAN	oznaka uspješnosti darivanja

krvnaVrsta•

- entitet čuva podatke o zalihi i predviđenim gornjim i donjim granicama za sve krvne grupe. one-to-many veza s potrosnjaKrv preko krvId i one-to-many veza s korisnikAplikacije preko krvId.

krvnaVrsta		
krvId	SERIAL	identifikator krvne grupe
imeKrvneGrupe	VARCHAR	naziv krvne grupe
gornjaGranica	INT	gornja granica dopuštene količine krvi u jedinicama
donjaGranica	INT	donja granica dopuštene količine krvi u jedinicama
trenutnaZaliha	INT	trenutna zaliha konkretne krvne grupe u jedinicama

potrosnjaKrv

entitet koji prati isporuke krvi . Many-to-one veza s korisnikAplikacije preko korisnikIdDjelatnika i Many-to-one veza s krvnaVrsta preko krvId.

potrosnjaKrv		
idPotrosnje	SERIAL	redni broj isporuke krvi
timestampPotrosnje	TIMESTAMP	timestamp potrošnje
krvId	INT	id krvne grupe
količinaJedinica	INT	broj jedinica koje su se potrošili
korisnikIdDjelatnika	VARCHAR	identifikator djelatnika koji je inicirao slanje krvi bolnici
lokacijaPotrosnje	VARCHAR	opisna lokacija kamo ide isporuka krvi

zdravstveniPodaci

entitet koji sprema sve moguće zdravstvene podatke koji se pitaju korisnika u upitniku. Oni sadrže svoj id, koji se sam generira u tablici, opis zdravstvenog podatka i težinu kriterija odnosno, 0 ako se na temelju toga podatka trajno odbija darivanje i 1 ako se privremeno odbija, inače null. One-to-many veza s doniranjeZdravljeOdgovori preko idZdravstvenih.

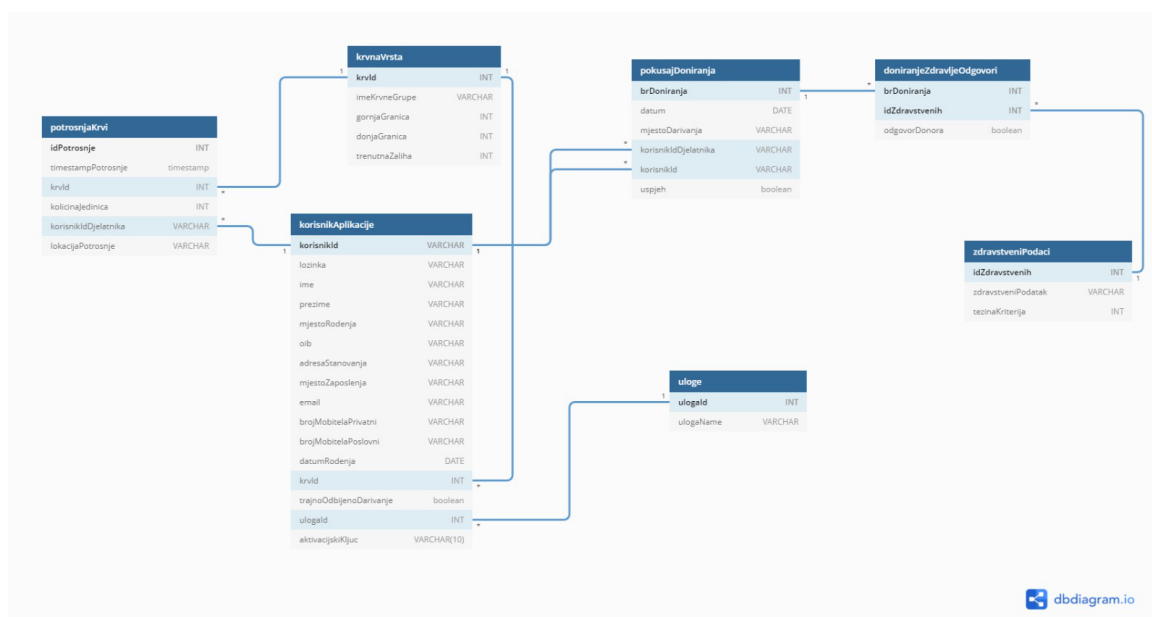
zdravstveniPodaci		
idZdravstvenih	SERIAL	id zdravstvenog podatka
zdravstveniPodatak	VARCHAR	opis zdravstvenog podatka koji se ispituje npr. osoba teži manje od 55 kg
tezinaKriterija	INT	tezina kriterija (0 za trajno, 1 za privremeno)

doniranjeZdravljeOdgovori•

entitet koji služi kao spremište odgovora korisnika koji su došli donirati krv. Many-to-one veza s zdravstveniPodaci preko idZdravstvenih. Many-to-one veza s pokusajDoniranja preko brDoniranja.

doniranjeZdravljeOdgovori		
brDoniranja	INT	brDoniranja iz tablice pokusajDoniranja
idZdravstvenih	INT	idZdravstvenih iz tablice zdravstveni podaci
odgovorDonora	BOOLEAN	odgovor donora na zdravstveni podatak pod idZdravstvenih

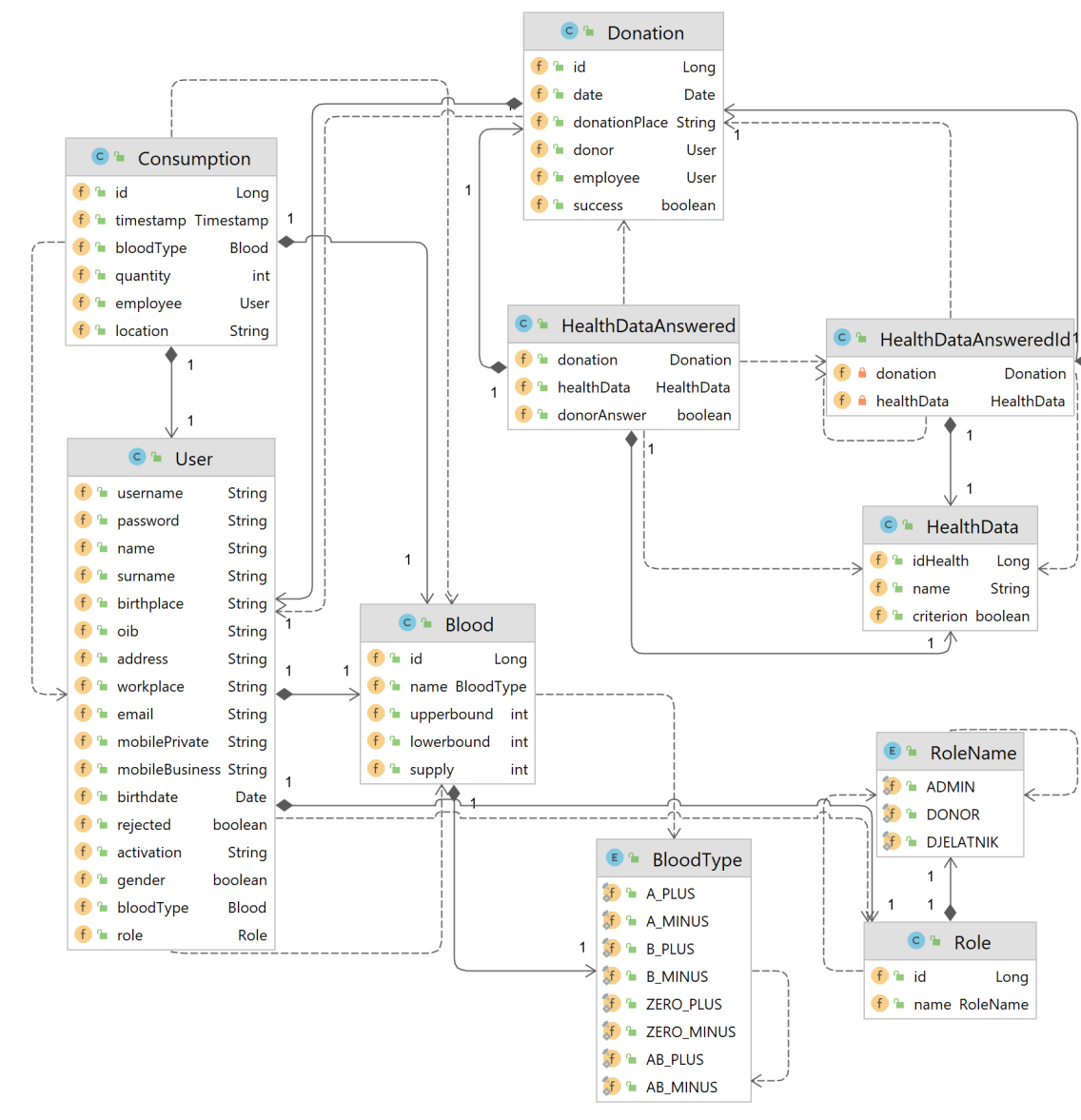
4.1.2 Dijagram baze podataka



Slika 4.4: relacijski model baze podataka

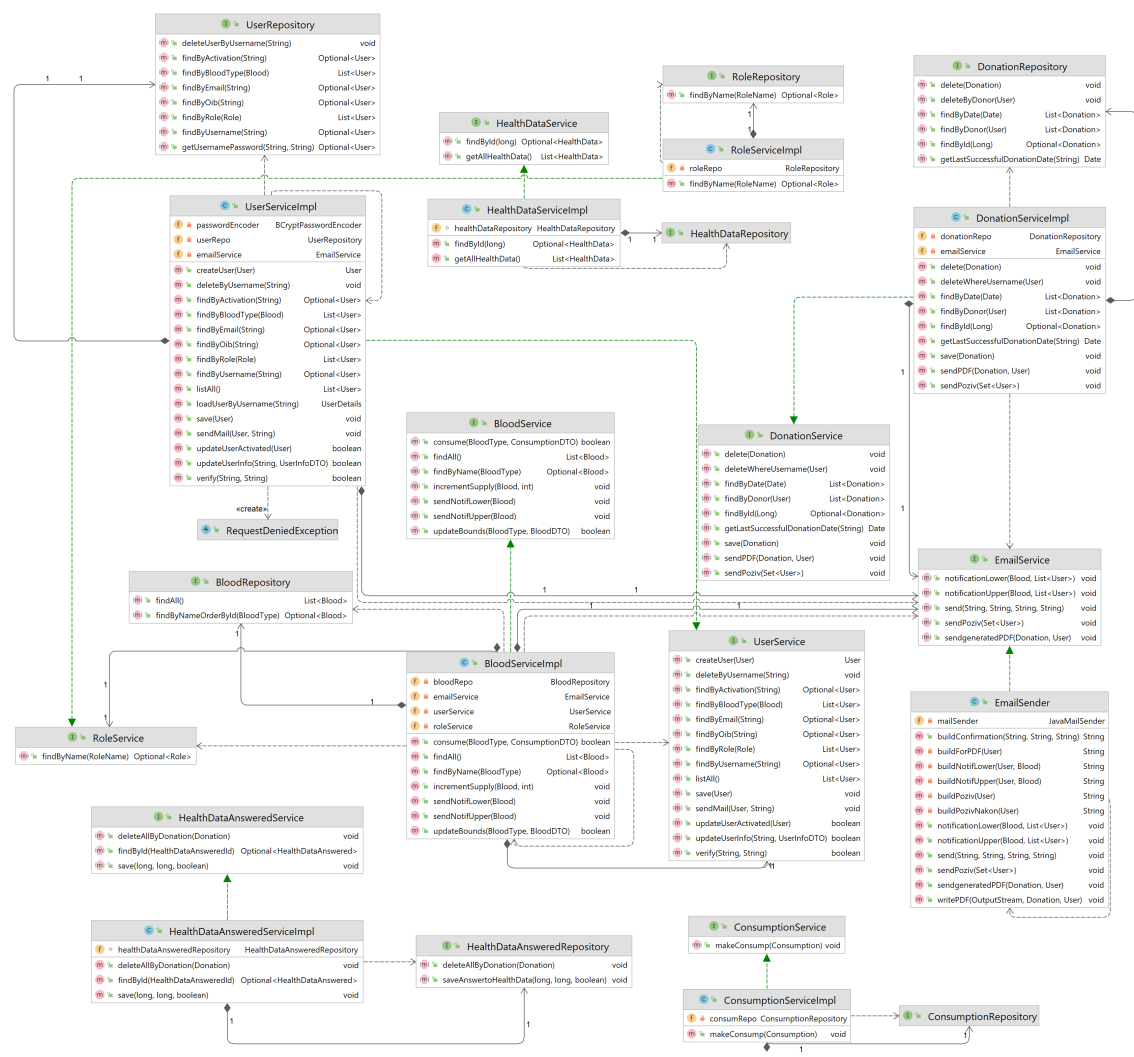
4.2 Dijagram razreda

iiiiiii HEAD Dijagram razreda pokazuje odnose između različitih objekata, te njihove attribute i operacije kojima vladaju. Na slikama 4.5., 4.6, 4.7. prikazani su razredi koji pripadaju *backend* dijelu naše arhitekture. Radi jednostavnosti, dijagram razreda je podijeljen u više slika, no bez obzira na to, prikazani razredi na neki način komuniciraju. Na idućoj slici prikazan je model podataka kojima backend rukuje. Korisnik aplikacije (administartor, djelatnik banke, donor) modeliran je razredom *User*. Razred *Blood* modelira krvne grupe. Razred *Consumption* modelira potrošnju krvi. Razred *Donation* modelira pokušaj donacije krvi, dok razred *Role* modelira 3 moguće uloge u aplikaciji. Razred *HealthData* modelira zdravstvena pitanja, a *HealthDataAnswered* modelira odgovore na određena zdravstvena pitanja. Također navedene su enumeracije za nazive krvnih grupa i nazive uloga. ===== Dijagram razreda pokazuje odnose između različitih objekata, te njihove attribute i operacije kojima vladaju. Na slikama 4.5., 4.6, 4.7., 4.8 prikazani su razredi koji pripadaju *backend* dijelu naše arhitekture. Radi jednostavnosti, dijagram razreda je podijeljen u više slika, no bez obzira na to, prikazani razredi na neki način komuniciraju. Na idućoj slici prikazan je model podataka kojima backend rukuje. Korisnik aplikacije (administartor, djelatnik banke, donor) modeliran je razredom *User*. Razred *Blood* modelira krvne grupe. Razred *Consumption* modelira potrošnju krvi. Razred *Donation* modelira pokušaj donacije krvi, dok razred *Role* modelira 3 moguće uloge u aplikaciji. Razredi *HealthData*, *HealthDataAnswered* i *HealthDataAnsweredId* modeliraju zdravstvene podatke potrebne za donacije. Također navedene su enumeracije za nazive krvnih grupa i nazive uloga. ~~~~~ devdoc



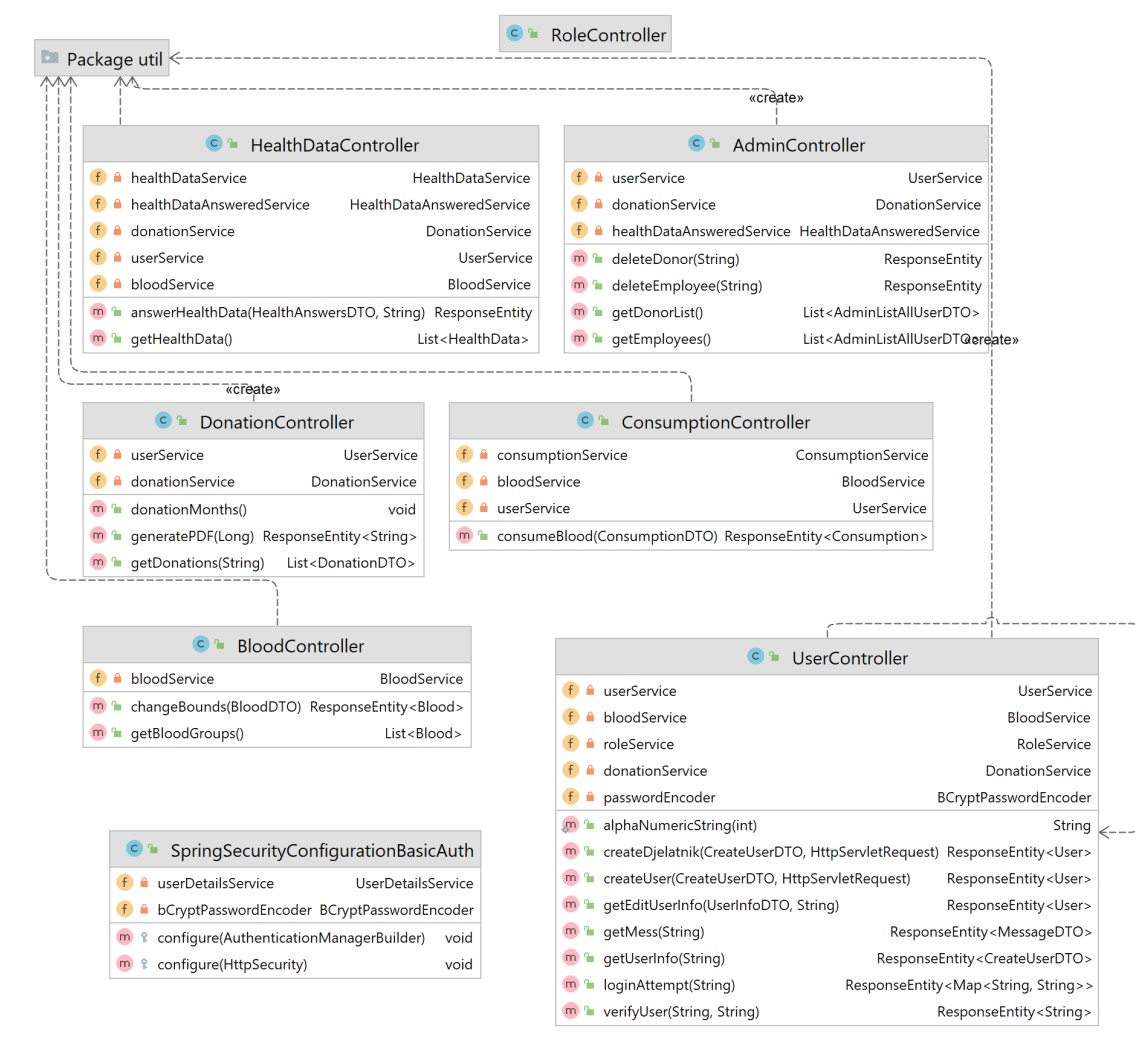
Slika 4.5: Dijagram razreda koji opisuje model

iiiiiii HEAD Na idućoj slici prikazana je sloj *backenda* koji je odgovoran za neposrednu komunikaciju s poslužiteljem baze podataka. Kao glavna komponenta na slici prikazano je sučelje JpaRepository, koje predstavlja apstraktni repozitorij podataka. Iz tog sučelja, izvedena su sučelja UserRepository, RoleRepository, BloodRepository, ConsumptionRepository, DonationRepository, HealthDataRepository i HealthDataAnsweredRepository. Ta sučelja predstavljaju repozitorij podataka za prije navedene razrede modela, tj. oni predstavljaju poveznicu s bazom ili DAO (eng. Data Access Object). Također koriste se različite ServiceJpa klase koje koriste te repozitorije, te one implementiraju sučelje imeKlaseService. ===== Na idućoj slici prikazana je sloj *backenda* koji je odgovoran za neposrednu komunikaciju s poslužiteljem baze podataka. Imamo sučelja UserRepository, RoleRepository, BloodRepository, ConsumptionRepository, DonationRepository, HealthDataRepository te HealthDataAnsweredRepository koja predstavljaju apstraktni repozitorij podataka. Ta sučelja predstavljaju repozitorij podataka za prije navedene razrede modela, tj. oni predstavljaju poveznicu s bazom ili DAO (eng. Data Access Object). Također koriste se različite Service klase koje koriste te repozitorije, te one implementiraju sučelje imeKlaseService. ?????? devdoc

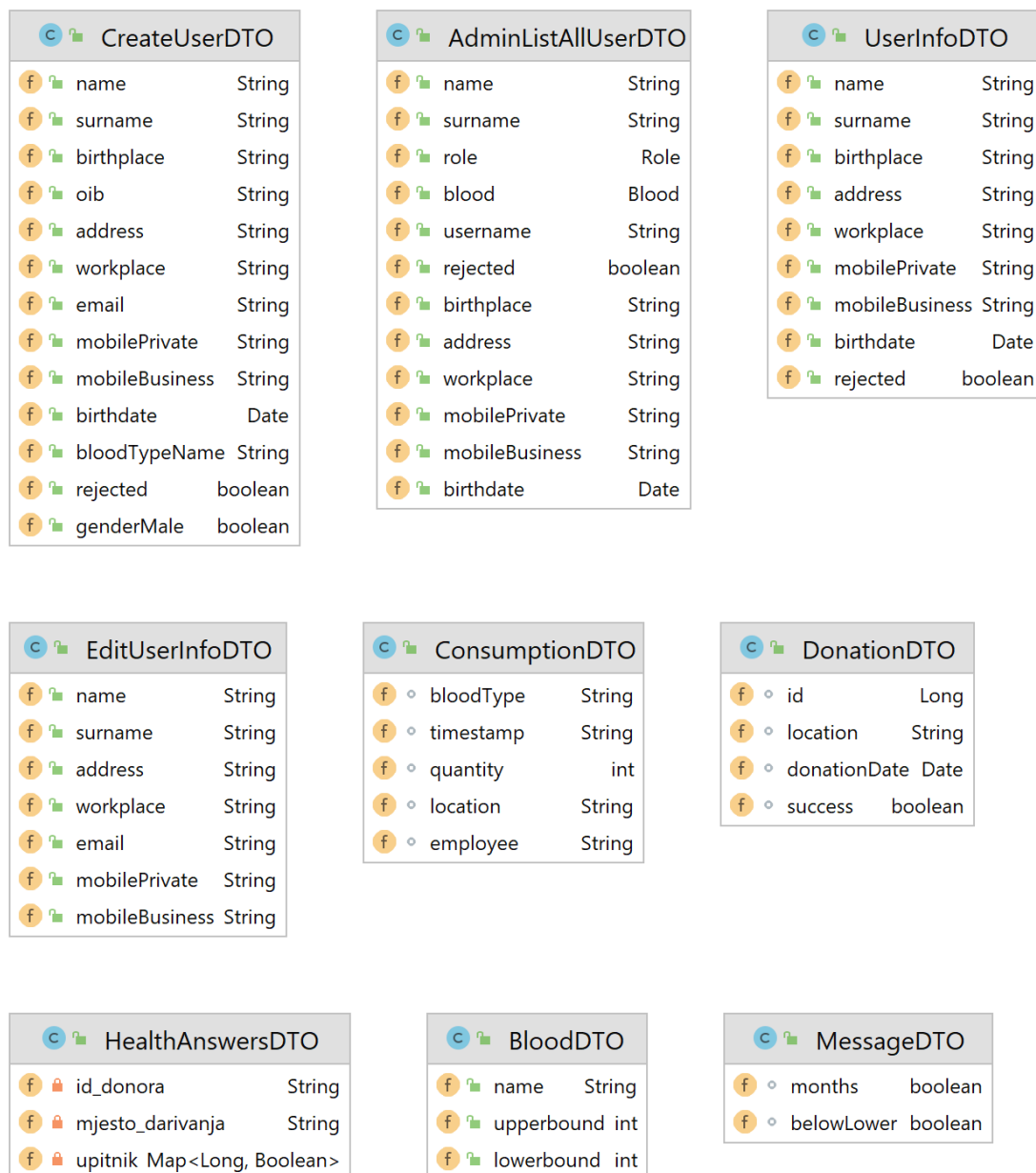


Slika 4.6: Dijagram razreda koji opisuju repozitorije

Na zadnje dvije slike prikazani su *Controlleri* koji komuniciraju s vanjskim svijetom odnosno *frontendom* te upravljaju modelom podataka i DTO-ovi (eng. Data Transfer Object) koje oni koriste za prijenos podataka. Ovdje vidimo razrede *UserController*, *RoleController*, *BloodController*, *ConsumptionController*, *DonationController*, *AdminController* i *HealthDataController*. Svi ti razredi koriste Java anotaciju *RestController* koji predstavlja REST endpoint i DTO-ove koji su u paketu *Util*. Ti razredi su oni koji dobivaju zahtjeve iz vanjskog svijeta, a odgovaraju HTTP odgovorima i JSON objektima.



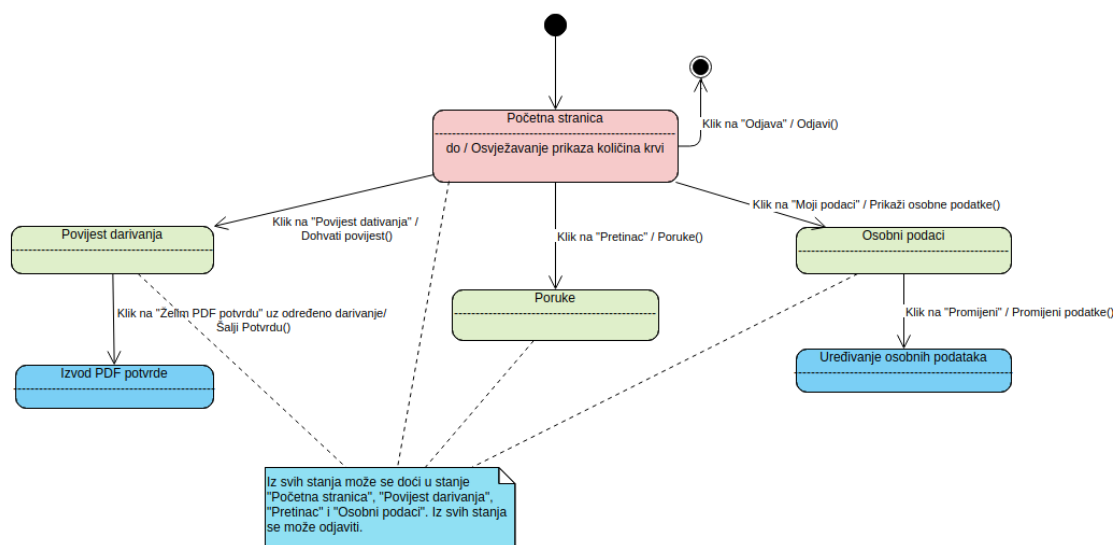
Slika 4.7: Dijagram razreda koji opisuje kontrolere



Slika 4.8: Dijagram razreda koji opisuju DTO-ove

4.3 Dijagram stanja

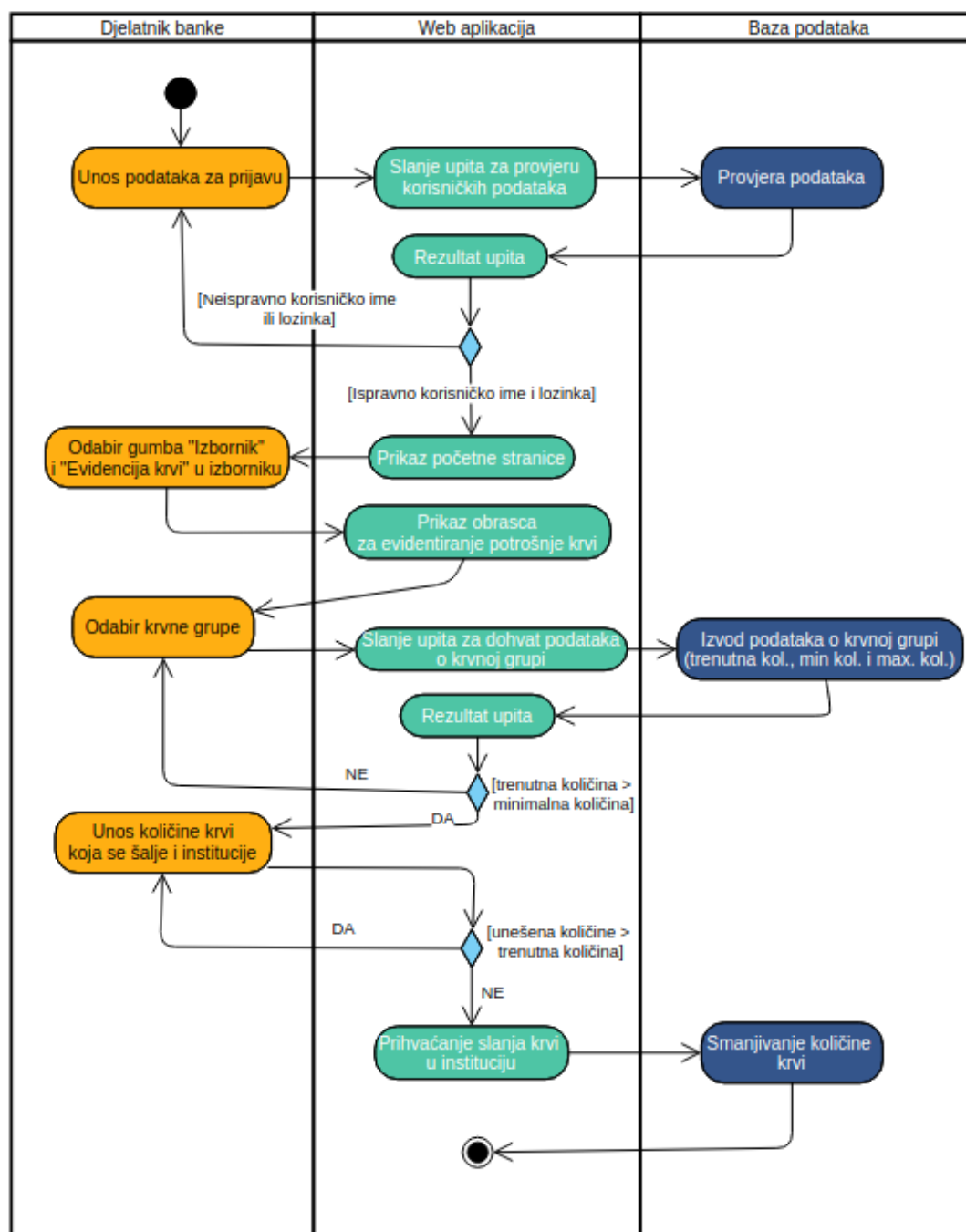
Dijagram stanja primijenjuje se za opis stanja objekta i za opisivanje prijelaza iz jednog u drugo stanje. Priložena slika prikazuje stanjadijagram stanja za registriranog donora. Početno stanje je početna stranica aplikacije. U izborniku aplikacije donor može odabrati jednu od mogućnosti. Klikom na "Moji podaci" prikazuju mu se njegovi podaci, koje može urediti. Odabirom opcije "Poruke" prikazuju se poruke donoru koje mu javlja sustav. Klikom na "Povijest darivanja" prikazuju se prijašnja darivanja za koje, ako je darivanje uspješno, može izvaditi PDF potvrdu.



Slika 4.9: Dijagram stanja

4.4 Dijagram aktivnosti

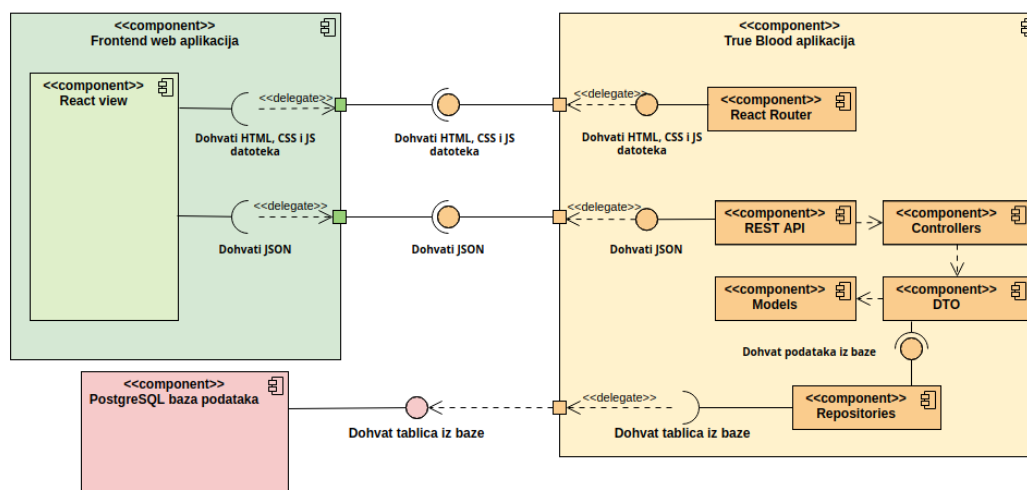
Dijagram aktivnosti prikazuje povezane aktivnosti na visokoj apstrakcijskoj razini, odnosno intuitivno prikazuje kako podaci teku kroz aplikaciju i kako se kontrola nad podacima mijenja. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog. Idući dijagram prikazuje proces smanjivanja količine krvi slanjem u zdravstvenu ustanovu. Ovaj proces može provesti djelatnik banke. Proces započinje prijavom, djelatnik banke odabere krvnu grupu na prikazu količina krvi i unese količinu krvi koju želi poslati zdravstvenoj ustanovi. Kada transakcija završi prikazuje mu se potvrda o smanjenju krvi.



Slika 4.10: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti omogućuje nam pogled na sustav s visoke razine apstrakcije. Web aplikacija komunicira sa sustavom preko dva sučelja. Prvo sučelje služi za dohvat samih stranica, dakle HTML, CSS i JS datoteka. Drugo sučelje služi za komunikaciju između aplikacije i baze podataka. Dakle, web aplikacija šalje zahtjev REST API-u, koji preko kontrolera komunicira s repozitorijima podataka, koji predstavljaju sloj povezanosti između baze podataka i kontrolera. Podaci koji su pristigli iz baze se šalju dalje MVC arhitekturi u obliku DTO (Data transfer object). React view komponenta preko dostupnih sučelja komunicira s web aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.11: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem platforme Whatsapp¹. Za izradu UML dijagrama korišten je alat Astah Professional² i Visual Paradigm³, a kao sustav za upravljanje izvornim kodom Git⁴. Udaljeni repozitorij projekta je dostupan na web platformi GitLab⁵.

Kao razvojno okruženje korišten je IntelliJ IDEA⁶ - integrirano razvojno okruženje (IDE) tvrtke JetBrains i Eclipse⁷.

Aplikacija je napisana u Javi⁸ koristeći ekosustav Java Spring⁹ za izradu backenda te React¹⁰ i jezik JavaScript¹¹ za izradu frontenda.

Baza podataka koju smo koristili je (PostgreSQL¹²) i ona se nalazi na pgAdmin¹³.

¹<https://www.whatsapp.com/>

²<https://astah.net/products/astah-professional/>

³<https://online.visual-paradigm.com/>

⁴<https://git-scm.com/>

⁵<https://gitlab.com/>

⁶<https://www.jetbrains.com/idea/>

⁷<https://www.eclipse.org/ide/>

⁸<https://www.java.com/en/>

⁹<https://spring.io/>

¹⁰<https://reactjs.org/>

¹¹<https://www.javascript.com/>

¹²<https://www.postgresql.org/>

¹³<https://www.pgadmin.org/>

5.2 Ispitivanje programskog rješenja

Nakon što smo završili s izradom testirali smo rad aplikacije koristeći JUnit tehnologiju i Selenium WebDriver. Testovi su dali zadovoljavajuće rezultate te možemo zaključiti da smo uspješno implementirati zadane funkcionalnosti.

5.2.1 Ispitivanje komponenti

Za ispitivanje komponenti koristili smo JUnit tehnologiju. JUnit okvir je za testiranje komponenti sustava programiranog u Javi. Ispitali smo funkcionalnosti ažuriranja osobnih podataka donora, promjenu optimalnih granica krvi i evidencije krvi slanjem u instituciju. Koriste se objekti tipa UserController, BloodController i ConsumptionController na kojima se pozivaju ispitivane funkcionalnosti te UserService, BloodService i RoleService koji su potrebni za dohvaćanje podataka korištenih u funkcijama.

Ažuriranje osobnih podataka donora

Pozivom funkcije getEditUserInfo iz klase UserController želimo donoru uspješno promijeniti prezime. Na kraju uspoređujemo je li prezime ažuriranog donora jednako željenom novom prezimenu.

```
@Test
public void editUserProfileCorrectly() {

    User user = new User("MR78912",
        "12345678",
        "Marko",
        "Radić",
        true,
        "Zagreb",
        "12345678912",
        "Radićeva 7",
        "Radnička 8",
        "markoradic@fer.hr",
        "0987412589",
        "0987412589",
        new Date(),
        roleService.findByName(RoleName.DONOR).get(),
        bloodService.findByName(BloodType.A_MINUS).get());

    userService.save(user);

    String newSurname="Radićević";
    UserInfoDTO userWithNewSurname = new UserInfoDTO(user.getName(),newSurname,
        user.getBirthplace(), user.getAddress(), user.getWorkplace(),
        user.getMobilePrivate(), user.getMobileBusiness(), user.getBirthdate(), false);

    String expected = newSurname;

    userController.getEditUserInfo(userWithNewSurname, user.getUsername());
    String result = userService.findByUsername(user.getUsername()).get().getSurname();
    Assertions.assertEquals(expected,result);
}
```

Slika 5.1: Unit test1

U idućem testu pozivom iste funkcije želimo neuspješno ažurirati polje rejected, koje se ne bi smjelo mijenjati. Na kraju uspoređujemo je li dobivena statusna poruka jednaka očekivanoj "400 BADREQUEST" što bi značilo da aplikacija na pobuđenu situaciju prikladno odgovara.

```
@Test
public void editUserProfileIncorrectly() {
    User user = new User("MR78912",
        "12345678",
        "Marko",
        "Radić",
        true,
        "Zagreb",
        "12345678912",
        "Radićeva 7",
        "Radnička 8",
        "markoradic@fer.hr",
        "0987412589",
        "0987412589",
        new Date(),
        roleService.findByName(RoleName.DONOR).get(),
        bloodService.findByName(BloodType.A_MINUS).get());

    userService.save(user);
    boolean newRejected = !user.isRejected();
    UserInfoDTO userWithNewRejected = new UserInfoDTO(user.getName(), user.getSurname(),
        user.getBirthplace(), user.getAddress(), user.getWorkplace(),
        user.getMobilePrivate(), user.getMobileBusiness(), user.getBirthdate(), newRejected);

    String expected = "400 BAD REQUEST";
    String result = userController.getEditUserInfo(userWithNewRejected, user.getUsername()).getStatusCode().toString();
    Assertions.assertEquals(expected, result);
}
```

Slika 5.2: Unit test2

Promjena optimalnih granica krvi

Pozivom funkcije changeBounds iz klase BloodController želimo uspješno promijeniti donju optimalnu granicu određene krvne grupe. Na kraju uspoređujemo je li donja optimalna granica ažurirane krvne grupe jednaka željenoj granici.

```
@Test
public void changeLowerboundCorrectly() {
    Blood blood = bloodService.findByName(BloodType.A_PLUS).get();
    int newLowerbound = 100;
    BloodDTO newBounds = new BloodDTO("A+", blood.getUpperbound(), newLowerbound);

    int expected = newLowerbound;

    bloodController.changeBounds(newBounds);
    int result = bloodService.findByName(BloodType.A_PLUS).get().getLowerbound();
    Assertions.assertEquals(expected, result);
}
```

Slika 5.3: Unit test3

U idućem testu pozivom iste funkcije želimo neuspješno ažurirati donju optimalnu granicu, koja bi uvijek morala biti pozitivna. Na kraju uspoređujemo je li dobivena statusna poruka jednaka očekivanoj "400 BADREQUEST" što bi značilo da aplikacija na pobuđenu situaciju prikladno odgovara.

```
@Test
public void changeLowerboundIncorrectly() {
    Blood blood = bloodService.findByName(BloodType.A_PLUS).get();

    int newLowerbound = -100;
    BloodDTO newBounds = new BloodDTO("A+", blood.getUpperbound(), newLowerbound);

    String expected = "400 BAD_REQUEST";

    String result = bloodController.changeBounds(newBounds).getStatusCode().toString();
    Assertions.assertEquals(expected, result);
}
```

Slika 5.4: Unit test4

Evidencija krvi slanjem u instituciju

Pozivom funkcije consumeBlood iz klase ConsumptionController želimo uspješno evidentirati slanje određene krvne grupe u neku instituciju. Na kraju uspoređujemo je li trenutna količina ažurirane krvne grupe jednaka količini umanjenjoj za količinu koja je poslana u instituciju.

```
@Test
public void makeConsumptionCorrectly() {
    Blood blood = bloodService.findByName(BloodType.A_PLUS).get();

    User employee = new User("DR78912",
        "12345678",
        "Dario",
        "Radić",
        true,
        "Zagreb",
        "12345678912",
        "Radićeva 7",
        "Radnička 8",
        "markoradic@fer.hr",
        "0987412589",
        "0987412589",
        new Date(),
        roleService.findByName(RoleName.DJELATNIK).get(),
        bloodService.findByName(BloodType.A_MINUS).get());

    userService.save(employee);

    int quantity = 25;
    ConsumptionDTO newConsump = new ConsumptionDTO(blood.getName().toString(), new Timestamp(0).toString(),
        quantity, "Zagreb", employee.getUsername());

    int expected = bloodService.findByName(blood.getName()).get().getSupply() - quantity;
    consumptionController.consumeBlood(newConsump);

    int result = bloodService.findByName(blood.getName()).get().getSupply();
    Assertions.assertEquals(expected, result);
}
```

Slika 5.5: Unit test5

U idućem testu pozivom iste funkcije želimo neuspješno evidentirati slanje krvi, čija bi količina uvijek morala biti pozitivna i veća od 0. Na kraju uspoređujemo je li dobivena statusna poruka jednaka očekivanoj "400 BADREQUEST" što bi značilo da aplikacija na pobuđenu situaciju prikladno odgovara.

```
@Test
public void makeConsumptionIncorrectly() {
    Blood blood = bloodService.findByName(BloodType.A_PLUS).get();

    User employee = new User("DR78912",
        "12345678",
        "Dario",
        "Radić",
        true,
        "Zagreb",
        "12345678912",
        "Radićeva 7",
        "Radnička 8",
        "markoradic@fer.hr",
        "0987412589",
        "0987412589",
        new Date(),
        roleService.findByName(RoleName.DJELATNIK).get(),
        bloodService.findByName(BloodType.A_MINUS).get());

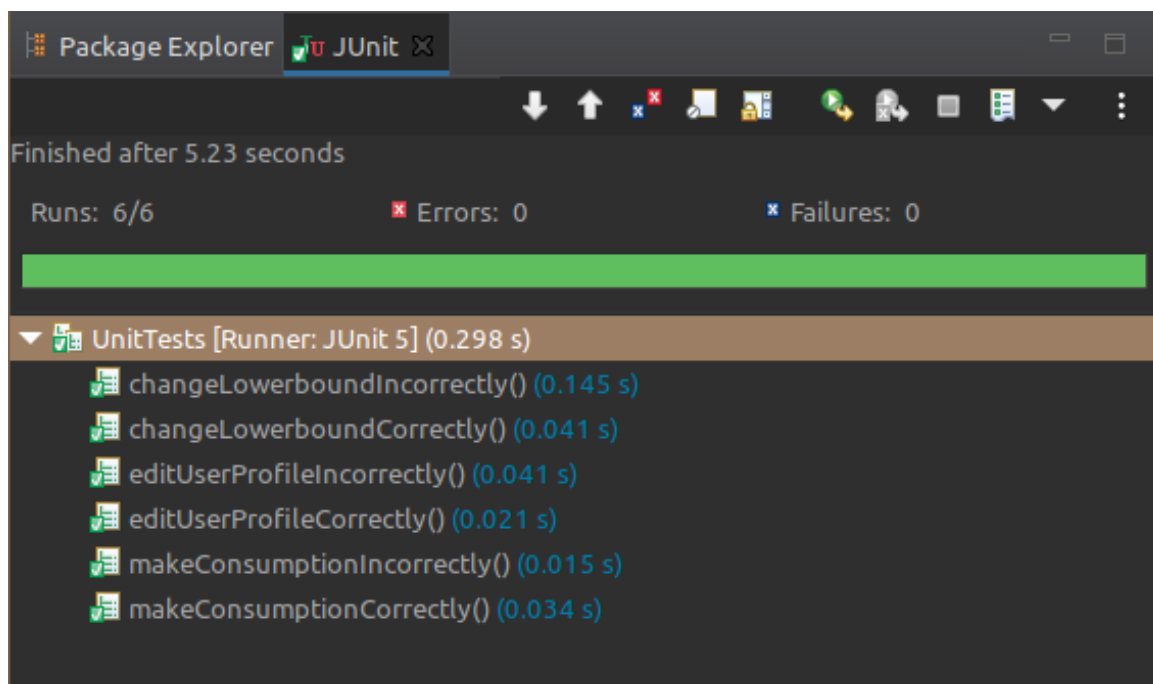
    userService.save(employee);

    int quantity = -5;
    ConsumptionDTO newConsump = new ConsumptionDTO(blood.getName().toString(), new Timestamp(0).toString(),
        quantity, "Zagreb", employee.getUsername());

    String expected = "400 BAD_REQUEST";

    String result = consumptionController.consumeBlood(newConsump).getStatusCode().toString();
    Assertions.assertEquals(expected, result);
}
```

Slika 5.6: Unit test6



Slika 5.7: Rezultati ispitivanja komponenti

5.2.2 Ispitivanje sustava

Selenium WebDriver, okvir koji omogućuje programsku interakciju sa internetskim preglednicima, je bio korišten za ispitivanje sustava. Testovi su provedeni na Google Chrome pregledniku. Za sljedeće funkcionalnosti je napravljeno programsko ispitivanje:

- prijava (za donore, djelatnike i admin-a)
- promjene optimalnih granica krvnih grupa (funkcionalnost admin-a)
- evidencija potrošnje krvi (funkcionalnost djelatnika)
- upisivanje donacije u registar (funkcionalnost djelatnika)

Korištena je dodatna funkcija za usporavanje sustava, kako se ne bi određeni dijelovi programskog koda za testiranje prerano izvršili:

```
public class SeleniumTests {  
  
    private static void waitMilliseconds(int milliseconds) {  
        try {  
            Thread.sleep(milliseconds);  
        } catch (InterruptedException e) {  
            // TODO Auto-generated catch block  
            Thread.currentThread().interrupt();  
        }  
    }  
}
```

Slika 5.8: Funkcija za usporavanje sustava

Prijava

Ovaj test ima za zadaću ispitati može li se svaka vrsta korisnika sustava ulogirati. Program učitava stranicu i pokušava se ulogirati sa zadanim username-om i password-om, te provjera je li doista ulogiran profil za zadani role. Uspjehom se smatra kada je zadani role upisan u drugo polje s desna u gornjoj traci.

```
//ispitivanje logina
public static boolean loginTest(WebDriver driver, String username, String password, String role) {

    driver.get("http://localhost:3000/login");
    // Find the username input element by its name
    WebElement element;
    element = driver.findElement(By.name("username"));
    // Enter username
    element.sendKeys(username);

    //Find password element and enter password
    element = driver.findElement(By.name("password"));
    element.sendKeys(password);

    //submit (login)
    driver.findElement(By.cssSelector("button[type='submit']")).click();

    waitMilliseconds(2000);
    element = driver.findElement(By.cssSelector("#root > div:nth-child(1) > nav > div.navbar-nav.justify-content-end.ms-auto > div:nth-child(4) > a"));

    if(element.getText().equals(role))
        return true;
    else
        return false;
}
```

Slika 5.9: Selenium test1

Promjena optimalnih granica krvnih grupa

Ovaj test ima za zadaću ispitati može li admin postaviti gornju i donju granicu količine krvi određene grupe, koje ako budu pređene dolazi do već spominjanih akcija. Test proglašavamo uspješnim ako poslije pritiska gumba brojevi prikazani u poljima "Gornja" i "Donja" odgovaraju željenima, ako su željene granice pozitivne vrijednosti. Ako pak nisu, onda je test uspješan kada se ispod gumba "Promjeni granice" prikaže poruka "Granice moraju biti pozitivne!".

```
public static boolean changeBloodLimit(WebDriver driver, int bloodIndex, String upperbound, String lowerbound) {  
  
    driver.get("http://localhost:3000/login");  
    WebElement element;  
    element = driver.findElement(By.name("username"));  
    element.sendKeys("admin");  
    element = driver.findElement(By.name("password"));  
    element.sendKeys("admin");  
  
    driver.findElement(By.cssSelector("button[type='submit']")).click();  
    waitMilliseconds(2000);  
    driver.findElement(By.cssSelector("#root > div:nth-child(1) > nav > div.navbar-nav.justify-content-end.ms-auto > div:nth-child(1) > a")).click();  
    driver.findElement(By.cssSelector("#root > div.App > div > div > div.col-md-2.mb-1.col > div > div:nth-child(3)")).click();  
  
    Select dropdown = new Select(driver.findElement(By.cssSelector("#parent > form > div:nth-child(1) > div.col-md-6 > select")));  
    waitMilliseconds(1000);  
    dropdown.selectByIndex(bloodIndex);  
  
    element = driver.findElement(By.name("upperbound"));  
    element.sendKeys(upperbound);  
    element = driver.findElement(By.name("lowerbound"));  
    element.sendKeys(lowerbound);  
  
    driver.findElement(By.cssSelector("button[type='submit']")).click();  
    waitMilliseconds(1000);  
    if (Integer.parseInt(upperbound) >= 0 && Integer.parseInt(lowerbound) >= 0) {  
        driver.switchTo().alert().accept();  
        dropdown = new Select(driver.findElement(By.cssSelector("#parent > form > div:nth-child(1) > div.col-md-6 > select")));  
        waitMilliseconds(1000);  
        dropdown.selectByIndex(bloodIndex);  
  
        String text = driver.findElement(By.cssSelector("#parent > form > div:nth-child(1) > div.col > p:nth-child(2)")).getText();  
        String gornjaGranica = text.substring(text.indexOf(":")+2);  
        text = driver.findElement(By.cssSelector("#parent > form > div:nth-child(1) > div.col > p:nth-child(3)")).getText();  
        String donjaGranica = text.substring(text.indexOf(":")+2);  
  
        return gornjaGranica.equals(upperbound) && donjaGranica.equals(lowerbound);  
    } else {  
        String textAlert = driver.findElement(By.cssSelector("#parent > form > div.mb-3.mt-3.row > div > div")).getText();  
        return textAlert.equals("Granice moraju biti pozitivne!");  
    }  
}
```

Slika 5.10: Selenium test2

Evidencija potrošnje krvi

U ovom se testu ispituje može li djelatnik evidentirati slanje krvi u određenu instituciju i odgovara li količina određene krvne grupe u banci nakon slanja očekivanoj količini. Test je uspješan ako je količina krvi određene krvne poslije slanja jednaka očekivanoj. Krv se šalje fiktivnoj lokaciji TESTNA LOKACIJA.

```
private static boolean sendBlood(WebDriver driver, int bloodIndex) {  
    driver.get("http://localhost:3000/login");  
    WebElement element;  
    element = driver.findElement(By.name("username"));  
    element.sendKeys("djelatnik");  
  
    //Find password element and enter password  
    element = driver.findElement(By.name("password"));  
    element.sendKeys("djelatnik");  
  
    //submit (login)  
    driver.findElement(By.cssSelector("button[type='submit']")).click();  
    waitMiliSeconds(2000);  
    driver.findElement(By.cssSelector("#root > div:nth-child(1) > nav > div.navbar-nav.justify-content-end.ms-auto > div:nth-child(1) > a")).click();  
    driver.findElement(By.cssSelector("#root > div.App > div > div > div.col-md-2.mb-1.col > div > div:nth-child(3)")).click();  
    //ovo je zbirnala podatki import gore  
    Select dropdown = new Select(  
        driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.container.col-md-3.border.border-danger.rounded > form > div:nth-child(1) > div > select")));  
    waitMiliSeconds(1000);  
    dropdown.selectByIndex(bloodIndex);  
  
    element = driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.container.col-md-3.border.border-danger.rounded > form > div:nth-child(2) > div:nth-child(1) > input"));  
    element.sendKeys("1");  
    element = driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.container.col-md-3.border.border-danger.rounded > form > div:nth-child(2) > div:nth-child(2) > input"));  
    element.sendKeys("IESINA_DISIUCIJA");  
  
    String text = driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.col > p:nth-child(5)")).getText();  
    String nakonSlanja = text.substring(text.indexOf(":")+2);  
  
    driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.container.col-md-3.border.border-danger.rounded > form > div.mb-3.row > div > button")).click();  
    waitMiliSeconds(1000);  
    driver.switchTo().alert().accept();  
  
    dropdown = new Select(driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.container.col-md-3.border.border-danger.rounded > form > div:nth-child(1) > div > select")));  
    waitMiliSeconds(1000);  
    dropdown.selectByIndex(bloodIndex);  
  
    text = driver.findElement(By.cssSelector("#left-tabs-example-tabpane-third > div > div > div.col > p:nth-child(5)")).getText();  
    String trenutnaZaliha = text.substring(text.indexOf(":")+2);  
    return nakonSlanja.equals(trenutnaZaliha);  
}
```

Slika 5.11: Selenium test3

Upisivanje donacije u registar

U ovom se testu ispituje može li djelatnik evidentirati obavljenju donaciju (koju je figurativno obavio donor imena TESTDONORNAME na lokaciji TESTLOKACIJA). Test je prolazan ako preglednik izbací poruku "Donacija evidentirana" ili "Nije moguće donirati! Nije prošlo dovoljno vremena od zadnje donacije!".

```
public static boolean makeDonation(WebDriver driver) {  
    driver.get("http://localhost:3000/login");  
    WebElement element;  
    element = driver.findElement(By.name("username"));  
    element.sendKeys("djelatnik");  
  
    //Find password element and enter password  
    element = driver.findElement(By.name("password"));  
    element.sendKeys("djelatnik");  
  
    //submit (login)  
    driver.findElement(By.cssSelector("button[type='submit']")).click();  
    waitMilliseconds(2000);  
  
    driver.findElement(By.cssSelector("#root > div:nth-child(1) > nav > div.navbar-nav.justify-content-end.ms-auto > div:nth-child(1) > a")).click();  
  
    driver.findElement(By.cssSelector("#root > div.App > div > div > div.col-md-2.mb-1.col > div > div:nth-child(2)")).click();  
    element = driver.findElement(By.name("filter"));  
    element.sendKeys("TEST_DONOR_NAME");  
  
    driver.findElement(By.cssSelector("#left-tabs-example-tabpane-second > div > div > div:nth-child(2) > table > tbody > tr:nth-child(3) > td:nth-child(6) > button")).click();  
    element = driver.findElement(By.name("mjestoDarivanja"));  
    element.sendKeys("TEST_LOKACIJA");  
  
    driver.findElement(By.cssSelector("#left-tabs-example-tabpane-second > div > div > div > div > form > div:nth-child(23) > div.col-md-6 > button")).click();  
    waitMilliseconds(1000);  
    String alertText = driver.switchTo().alert().getText();  
    driver.switchTo().alert().accept();  
  
    return alertText.equals("Donacija evidentirana") || alertText.equals("Nije moguće donirati! Nije prošlo dovoljno vremena od zadnje donacije!");  
}
```

Slika 5.12: Selenium test4

Te sve testne funkcije iskoristimo za provođenje testova u glavnoj (main) funkciji:

```
public static void main(String[] args) {  
  
    System.setProperty("webdriver.chrome.driver", "C:\\\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");  
    WebDriver driver = new ChromeDriver();  
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  
  
    boolean result;  
  
    result = loginTest(driver, "admin", "admin", "admin");  
    System.out.println("Test pass: " + result);  
  
    result = loginTest(driver, "djelatnik", "djelatnik", "djelatnik");  
    System.out.println("Test pass: " + result);  
  
    result = loginTest(driver, "donor", "donor", "donor");  
    System.out.println("Test pass: " + result);  
  
    result = sendBlood(driver, 2);  
    System.out.println("Test pass: " + result);  
  
    result = changeBloodLimit(driver, 2, "350", "150");  
    System.out.println("Test pass: " + result);  
  
    result = changeBloodLimit(driver, 2, "55", "-5");  
    System.out.println("Test pass: " + result);  
  
    result = makeDonation(driver);  
    System.out.println("Test pass: " + result);  
    driver.quit();  
}
```

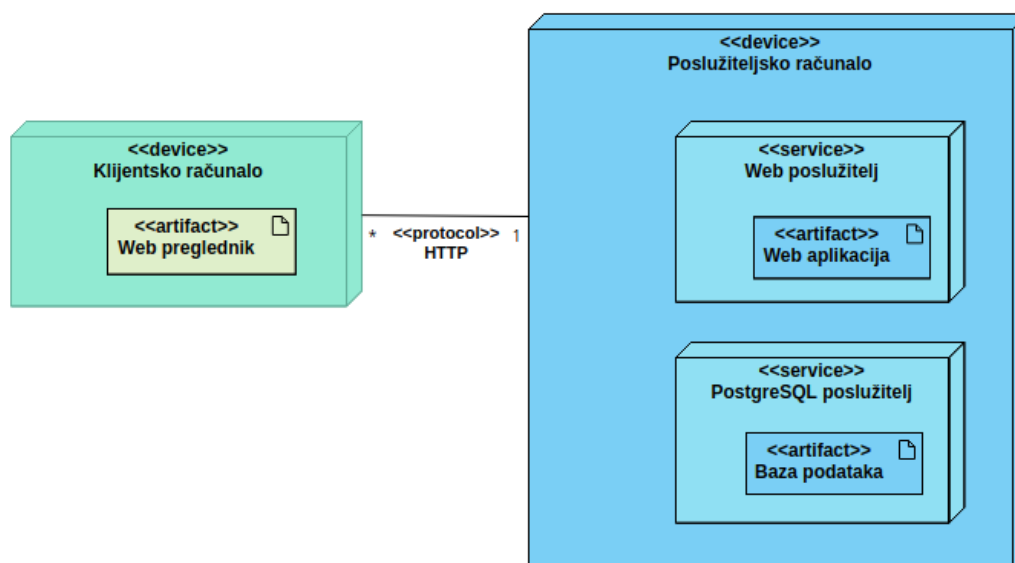
Slika 5.13: Funkcija main

```
Test pass: true  
Test pass: true  
Test pass: true  
Test pass: true  
Test pass: true  
Test pass: true  
Test pass: true
```

Slika 5.14: Rezultat ispitivanja sustava

5.3 Dijagram razmještaja

Dijagrami razmještaja prikazuju topologiju sustava i odnos sklopovskih i programskih dijelova. Olakšavaju nam vizualizaciju razmještaja fizičkog dijela sustava i sklopovlja. Sustav se sastoji od klijentskog i poslužiteljskog računala. Klijent na svojem računalu preko web preglednika pristupa aplikaciji. Klijentsko računalo komunicira s poslužiteljskim računalom preko HTTP veze. Na poslužiteljskom se računalu nalaze web poslužitelj i poslužitelj baze podataka (PostgreSQL).



Slika 5.15: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Ove upute vrijede te su rađene za Windows Server 2019 desktop experience operativni sustav. Slična (ili gotovo ista) procedura je i za ostale Windows, pa čak i Linux operativne sustave. Većina razlika je u samoj instalaciji potrebnih programskih paketa i runtime-a te postavljanje putanje, nakon čega je konfiguriranje aplikacije i njezino korištenje gotovo identično.

5.4.1 Instalacija poslužitelja baze podataka

Potrebno je preuzeti PostgreSQL bazu podataka za Windows x86-64¹⁴. Preporuča se verzija 13.x. Prilikom instalacije preporuča se ostaviti sve (po defaultu) označene komponente za instalaciju, jer će kasnije biti potrebno inicijalno popuniti bazu pomoću pgAdmin4 alata. Odaberite šifru za korisnika postgres te port na kojem će baza podataka slušati zahtjeve. Prilikom završetka instalacije nije potrebno pokretati stack builder. Nakon instalacije (i prilikom svakog restarta servera), Windows automatski pokreće servis baze podataka, te je ona dostupna na prethodno specificiranom portu.

5.4.2 Instalacija Node.js runtime-a

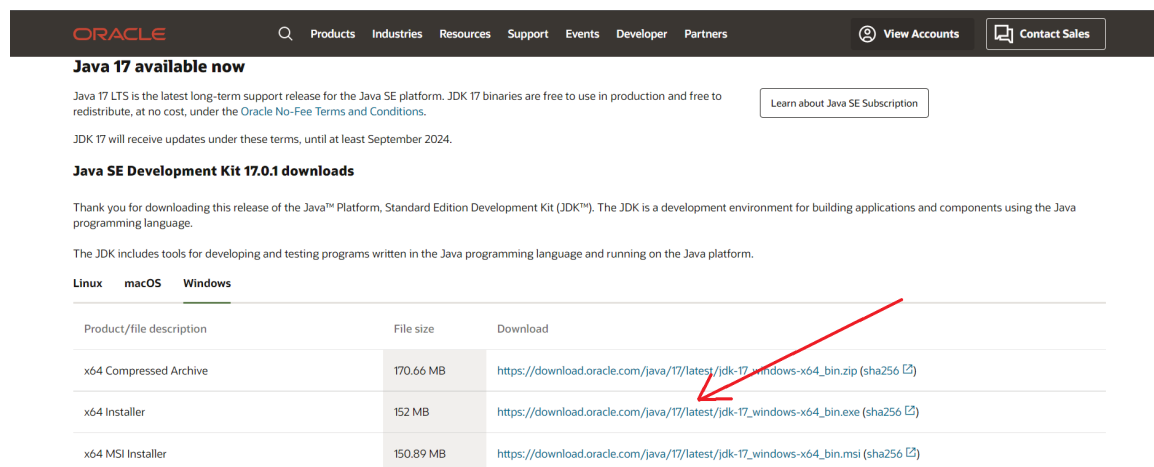
Potrebno je preuzeti i instalirati Node.js runtime¹⁵ koji će pokrenuti frontend dio aplikacije. Preporuča se trenutno dostupna LTS verzija (16.13.2 na dan pisanja). Prilikom instalacije potrebno je ostaviti sve (po defaultu) označene komponente za instalaciju, dok ponuđeni alat Chocolatey nije potrebno instalirati. Installer je automatski postavio PATH varijablu okruženja, te su *npm* i *node* naredbe postale globalno dostupne u cmd-u.

¹⁴<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads/>

¹⁵<https://nodejs.org/en/download/>

5.4.3 Instalacija Java

Potrebno je preuzeti i instalirati JDK paket¹⁶ koji će pokretati backend dio aplikacije.



ORACLE Products Industries Resources Support Events Developer Partners View Accounts Contact Sales

Java 17 available now

Java 17 LTS is the latest long-term support release for the Java SE platform. JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions](#).

JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17.0.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

Linux macOS Windows

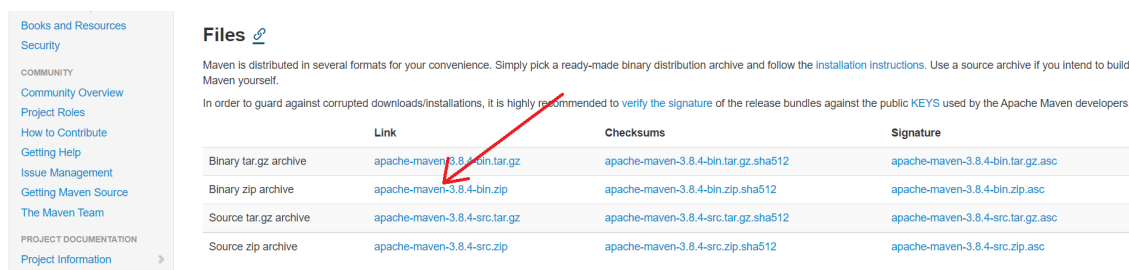
Product/file description	File size	Download
x64 Compressed Archive	170.66 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256 🔗)
x64 Installer	152 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256 🔗)
x64 MSI Installer	150.89 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256 🔗)

Slika 5.16: Download JDK17 Installera

Potrebna je trenutna 17.x.x verzija, no moguće je koristiti i neku od novijih verzija. Installer je automatski postavio PATH varijablu okruženja, te je *java* naredba postala globalno dostupna u cmd-u.

5.4.4 Instalacija Maven alata i postavljanje PATH varijable okruženja

Potrebno je preuzeti Maven alat¹⁷ koji služi za kreiranje .jar datoteke iz izvornog koda koja sadrži cijelu backend aplikaciju sa web serverom. Preporuča se 3.8.4 verzija ili novija.



Books and Resources Security

COMMUNITY Community Overview Project Roles How to Contribute Getting Help Issue Management Getting Maven Source The Maven Team

PROJECT DOCUMENTATION Project Information

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

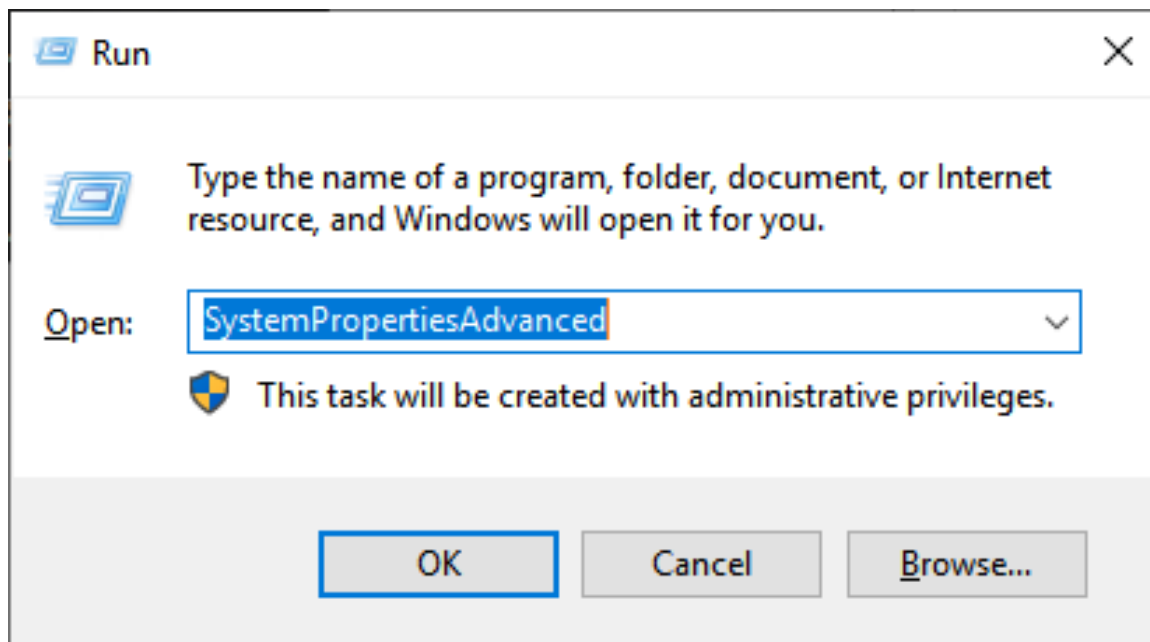
	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.8.4-bin.tar.gz	apache-maven-3.8.4-bin.tar.gz.sha512	apache-maven-3.8.4-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.4-bin.zip	apache-maven-3.8.4-bin.zip.sha512	apache-maven-3.8.4-bin.zip.asc
Source tar.gz archive	apache-maven-3.8.4-src.tar.gz	apache-maven-3.8.4-src.tar.gz.sha512	apache-maven-3.8.4-src.tar.gz.asc
Source zip archive	apache-maven-3.8.4-src.zip	apache-maven-3.8.4-src.zip.sha512	apache-maven-3.8.4-src.zip.asc

Slika 5.17: Download Maven Binary-a

¹⁶<https://www.oracle.com/java/technologies/downloads/jdk17-windows/>

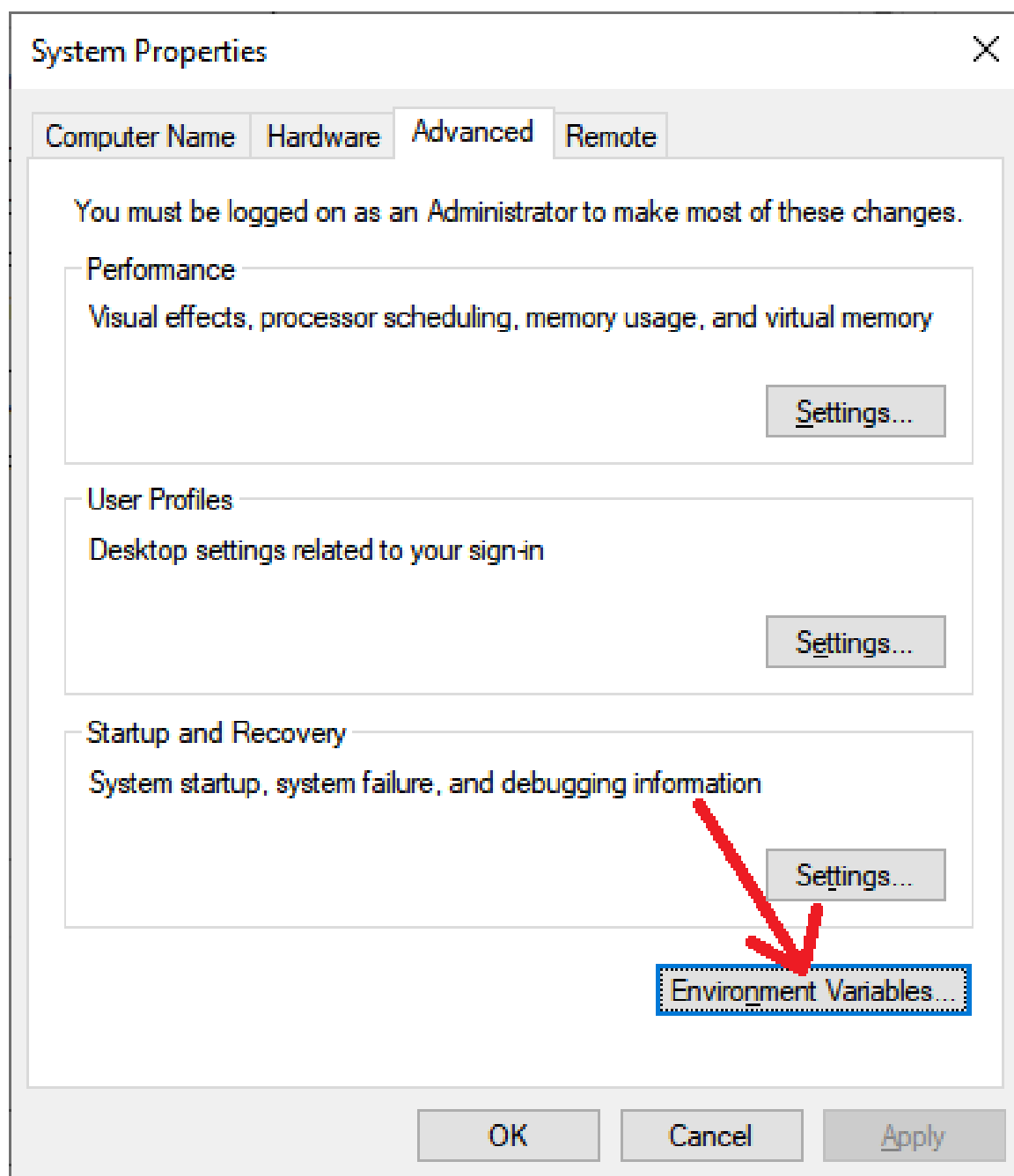
¹⁷<https://maven.apache.org/download.cgi/>

Nakon preuzimanja potrebno je raspakirati folder u proizvoljnu mapu (u ovom primjeru u *C:/Program Files/apache-maven-3.8.4-bin*) te dodati *bin* folder u PATH varijablu okruženja što se napravi na sljedeći način: Pritiskom *Windows + R* tipki otvara se prozor u kojem treba napisati *SystemPropertiesAdvanced* i pritisnuti Enter.



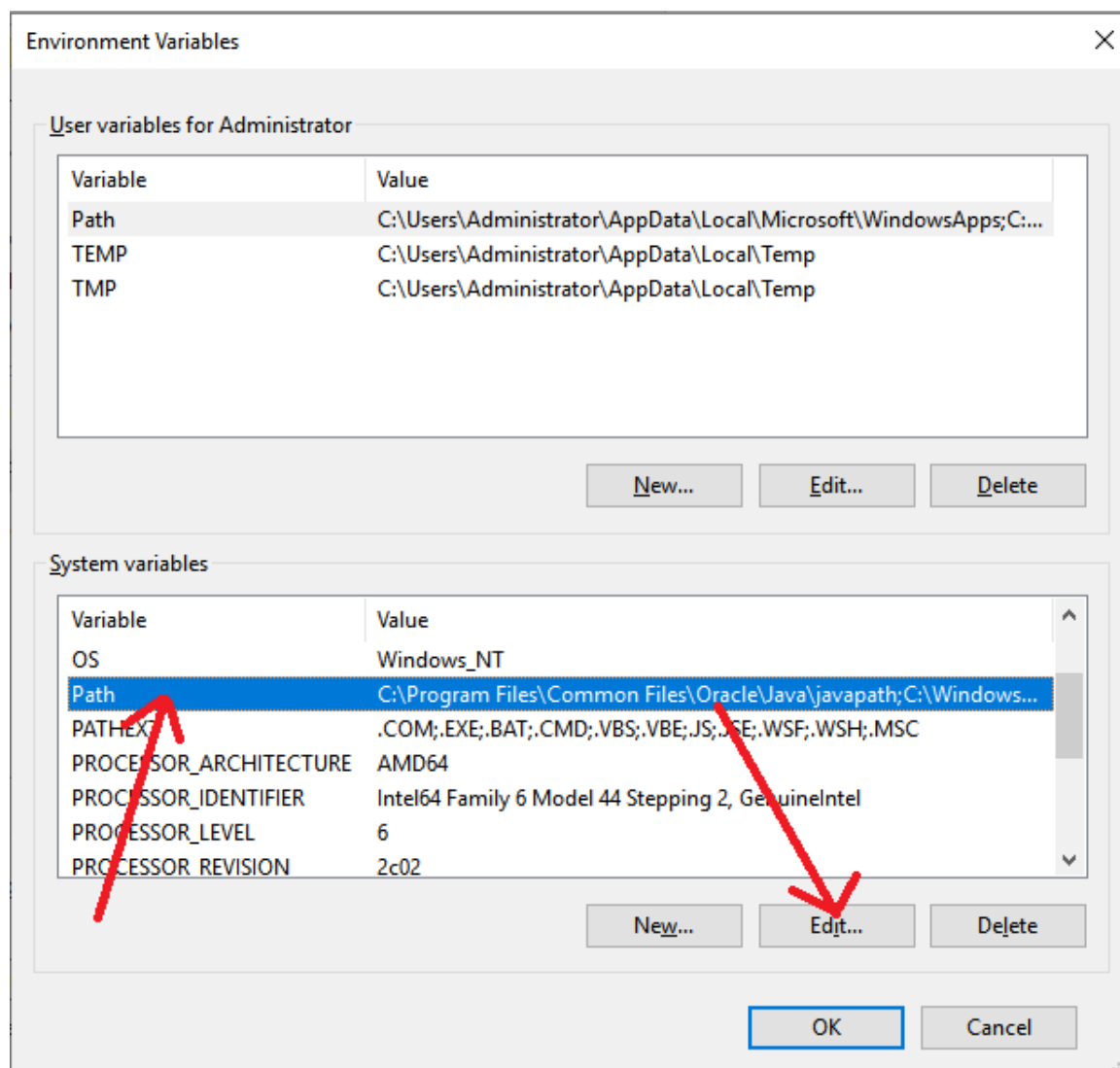
Slika 5.18: Prozor pokreni

Nakon toga otvara se drugi prozor gdje treba pritisnuti gumb *Environment Variables....*



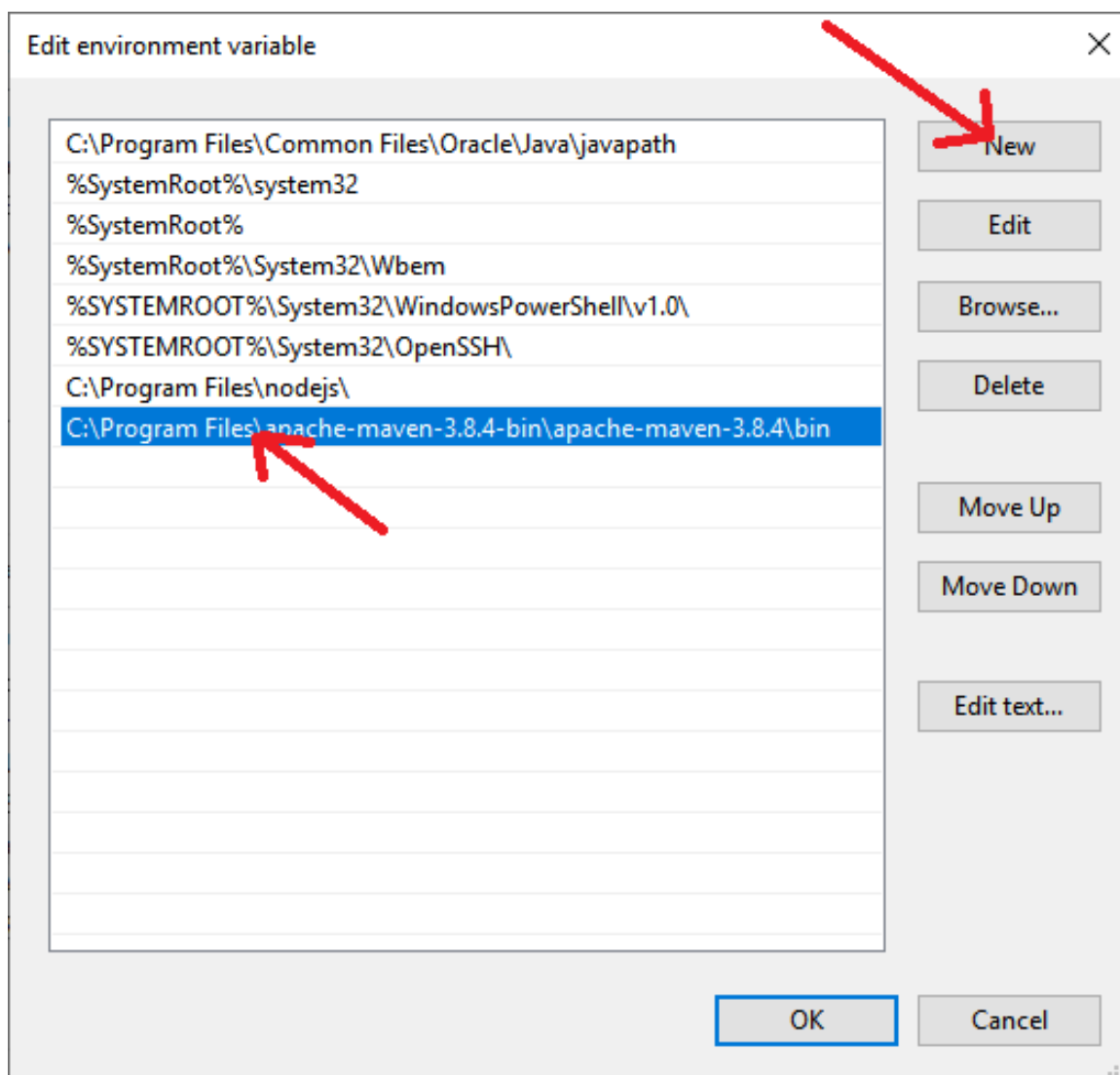
Slika 5.19: Prozor naprednih postavka sustava

Time se otvara i treći prozor gdje treba locirati sistemsku *Path* varijablu, označiti ju te pritisnuti gumb *Edit...*



Slika 5.20: Prozor varijabli okruženja

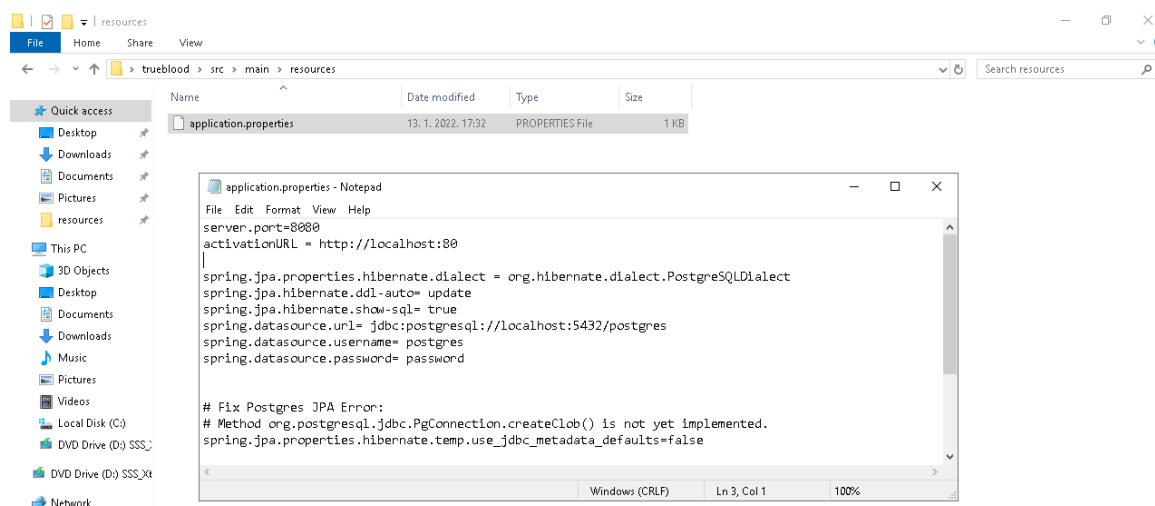
Nakon toga, u novootvorenom prozoru, pritiskom na tipku *New* potrebno je napisati putanju do *bin* foldera prethodno raspakiranog Maven alata. U ovom slučaju to je *C/Program Files/apache-maven-3.8.4-bin/apache-maven-3.8.4/bin*.



Slika 5.21: Nova putanja u PATH varijabli okruženja

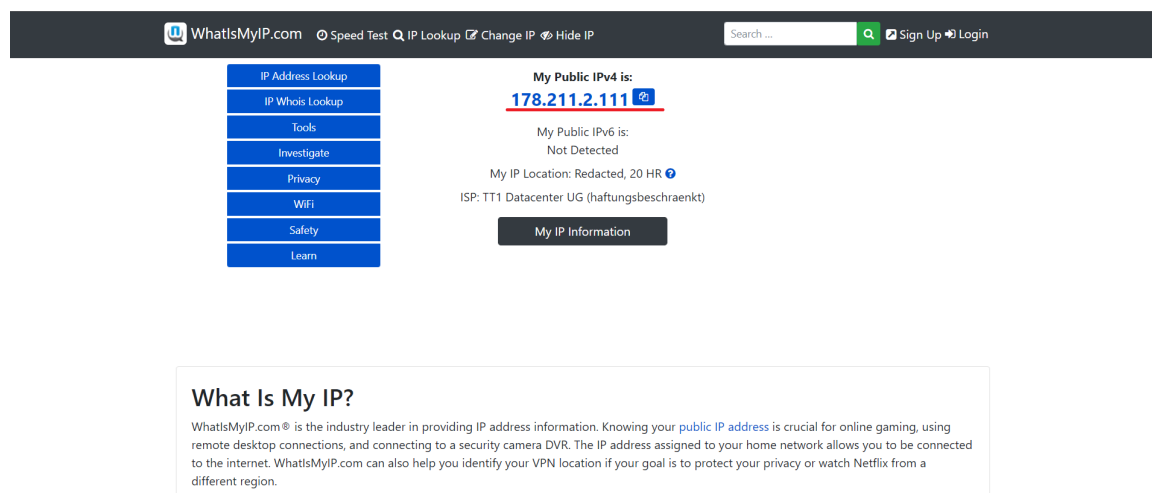
5.4.5 Podešavanje i pokretanje backend dio aplikacije te inicijalno punjenje baze

Kako bi aplikacija ispravno radila, potrebno je postaviti parametre za povezivanje s bazom podataka te ispravnu IP adresu i port za aktivacijski link u datoteci *application.properties* (koja se nalazi na putanji *IzvorniKod/trueblood/src/main/resources*).



Slika 5.22: application.properties datoteka

Na samom vrhu nalazi se konfiguracijska linija *server.port=8080* kojom se može promijeniti port backend servera (default je 8080). Ispod toga, nalazi se linija *activationURL = http://localhost:80*. Ovdje je potrebno specificirati javnu IP adresu servera (ili DNS name tj. link ukoliko se koristi), te port frontend dijela aplikacije (ukoliko nije default 80). Jedan od načina kako se to može napraviti jest tako da se na serveru otvori stranica <https://www.whatismyip.com/> te se kopira ispisana IP adresa na mjesto (*activationURL = http://IPadresa:80*)



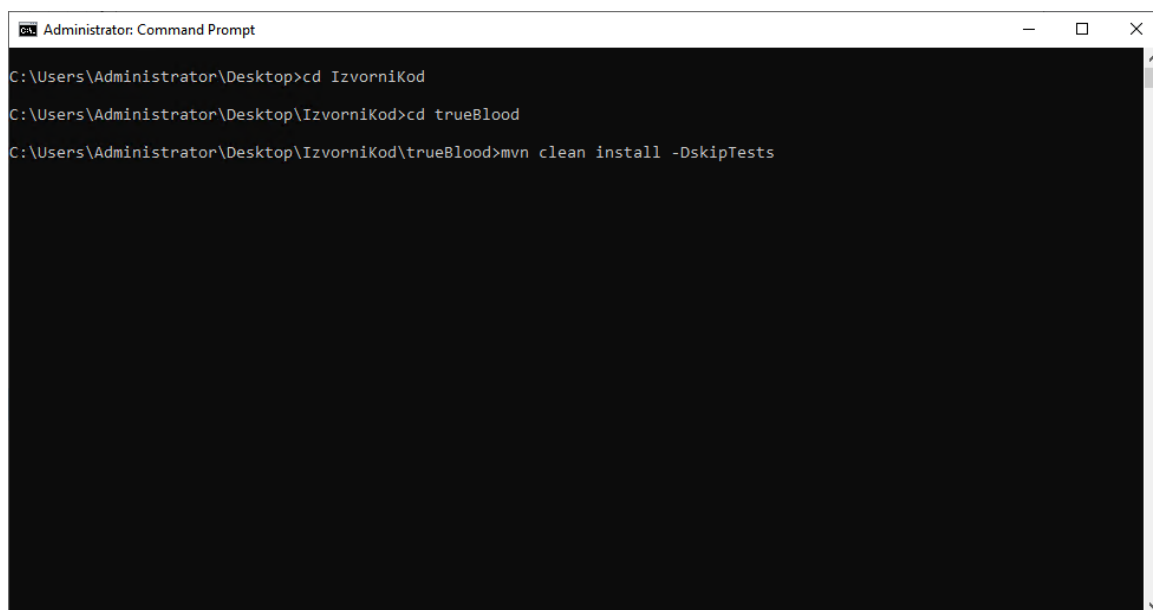
Slika 5.23: Kopiranje javne IPv4 adrese servera

Nadalje, ispod se nalaze 3 ključne linije za povezivanje s bazom:

- `spring.datasource.url=jdbc:postgresql://localhost:5432/postgres`
- `spring.datasource.username= postgres`
- `spring.datasource.password= password`

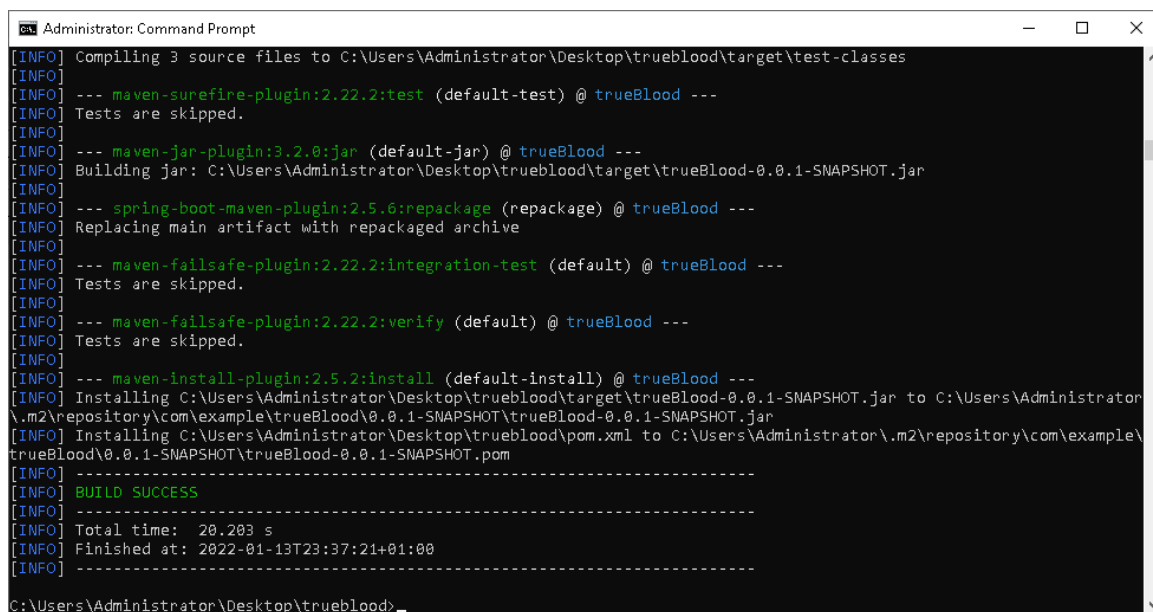
Na prvoj liniji je potrebno specificirati odabrani port prilikom instalacije `../localhost:portnumber/..` te ime baze `...5432/imebaze` ukoliko se ne koristi default baza. Na drugoj liniji specificira se ime korisnika kojeg aplikacija koristi za pristup bazi, te na trećoj liniji lozinka tog korisnika. Ukoliko se koristi default `postgres` korisnik za koji je definirana lozinka prilikom instalacije, ovdje se upiše ta ista lozinka.

Nakon modificiranja i spremanja konfiguracijske datoteke, potrebno je otvoriti cmd, pozicionirati se u trueBlood folder `IzvorniKod/trueBlood` te izvršiti komandu `mvn clean install -DskipTests` kojom će Maven alat preuzeti potrebne dependency-je te izraditi .jar datoteku spremnu za pokretanje.



```
Administrator: Command Prompt
C:\Users\Administrator\Desktop>cd IzvorniKod
C:\Users\Administrator\Desktop\IzvorniKod>cd trueBlood
C:\Users\Administrator\Desktop\IzvorniKod\trueBlood>mvn clean install -DskipTests
```

Slika 5.24: Pozicioniranje i izvršavanje komandi

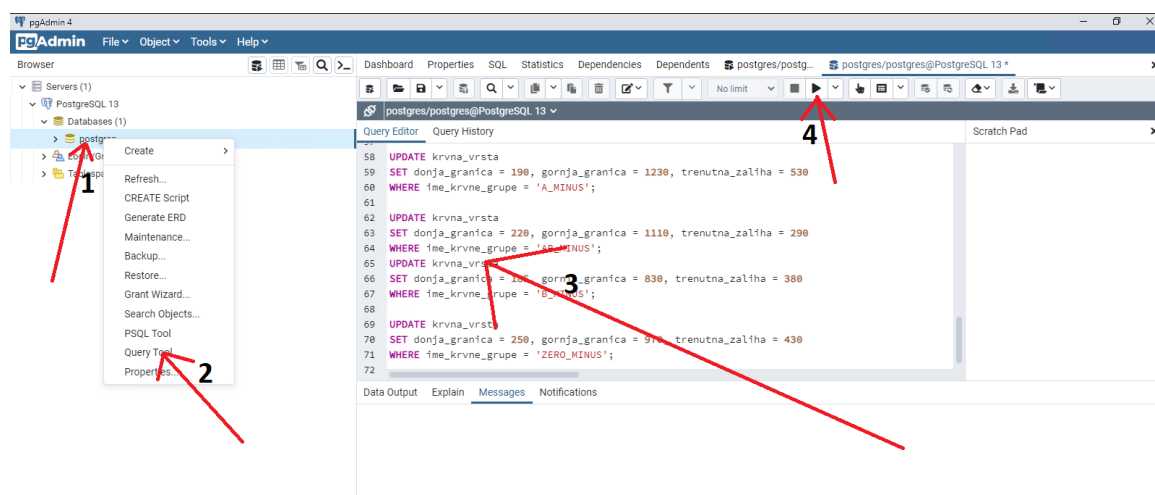


```
Administrator: Command Prompt
[INFO] Compiling 3 source files to C:\Users\Administrator\Desktop\trueblood\target\test-classes
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ trueBlood ---
[INFO] Tests are skipped.
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ trueBlood ---
[INFO] Building jar: C:\Users\Administrator\Desktop\trueblood\target\trueBlood-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot-maven-plugin:2.5.6:repackage (repackage) @ trueBlood ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-failsafe-plugin:2.22.2:integration-test (default) @ trueBlood ---
[INFO] Tests are skipped.
[INFO] --- maven-failsafe-plugin:2.22.2:verify (default) @ trueBlood ---
[INFO] Tests are skipped.
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ trueBlood ---
[INFO] Installing C:\Users\Administrator\Desktop\trueblood\target\trueBlood-0.0.1-SNAPSHOT.jar to C:\Users\Administrator\
.m2\repository\com\example>trueBlood\0.0.1-SNAPSHOT\trueBlood-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\Administrator\Desktop\trueblood\pom.xml to C:\Users\Administrator\m2\repository\com\example\
trueBlood\0.0.1-SNAPSHOT\trueBlood-0.0.1-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 20.203 s
[INFO] Finished at: 2022-01-13T23:37:21+01:00
[INFO] -----
C:\Users\Administrator\Desktop\trueblood>
```

Slika 5.25: Završni izlaz nakon uspješnog builda aplikacije

Izrađena .jar datoteka (koja u sebi sadrži sve što je potrebno za njezino pokretanje, uključujući i Tomcat web server) nalazi se na putanji *IzvorniKod/trueBlood/target* pod imenom *trueBlood-0.0.1-SNAPSHOT*, te se sa cmd-om potrebno pozicionirati u navedeni folder i izvršiti komandu *java -jar trueBlood-0.0.1-SNAPSHOT.jar* čime se pokreće backend dio aplikacije. Potrebno je ostaviti otvoren cmd prozor kako bi aplikacija radila.

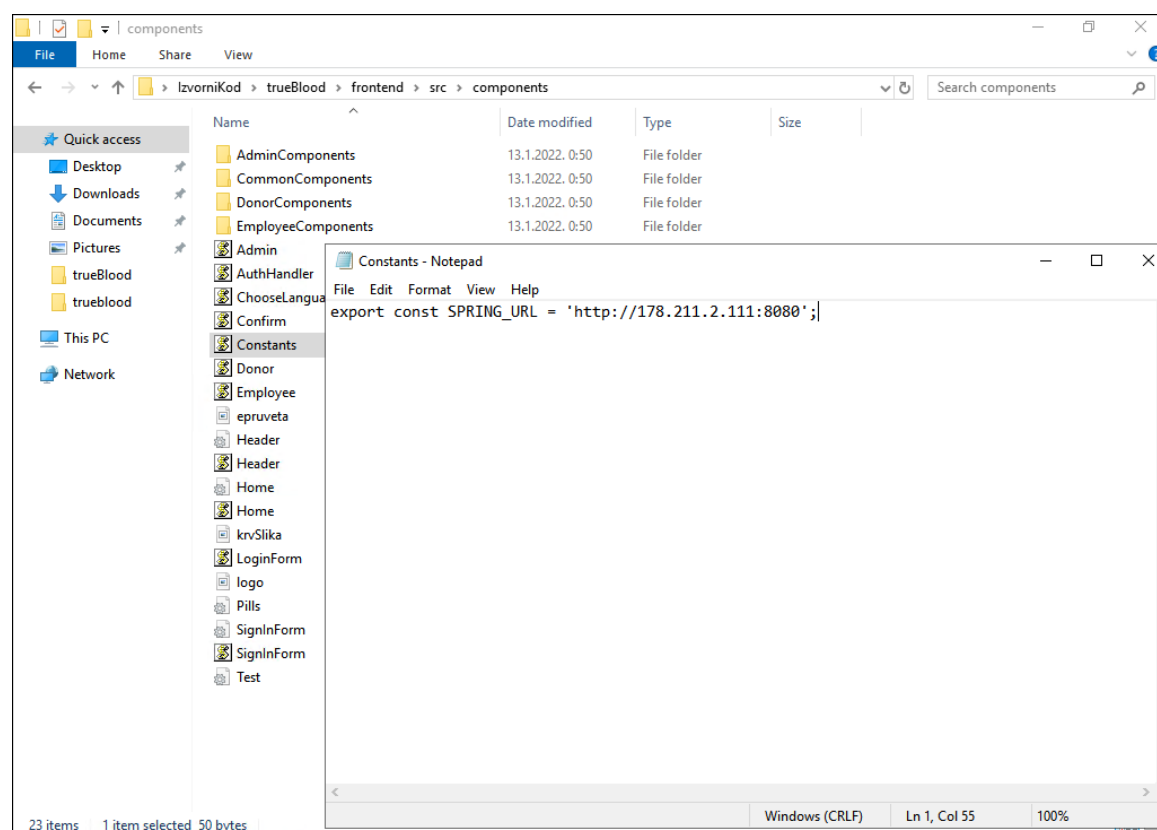
Aplikacija sama stvara tablice i strukturu podataka u schemu *public* unutar baze specificirane u *application.properties*. No za ispravan rad potrebno je još dodavanje statičkih redova na način da se otvori prije instalirana aplikacija PgAdmin, desnim klikom na korištenu bazu odabere se *Query Tool*, te u tekstni prozor se kopira sadržaj datoteke *puniteljBaze.sql* (koja se nalazi na putanji *IzvorniKod/trueblood/puniteljBaze.sql*). Ovim početnim punjenjem baze podataka, definira se početni admin korisnički račun (username: *admin*, password: *admin*) pomoću kojeg je moguće daljnje upravljanje aplikacijom.



Slika 5.26: Punjenje baze statičkim podacima

5.4.6 Podešavanje i pokretanje frontend dijela aplikacije

Nakon uspješno pokrenutog backend dijela aplikacije, potrebno je još posložiti i pokrenuti i frontend dio aplikacije. Potrebno je javnu IP adresu servera i port backend dijela aplikacije (definiranog prije u *application.properties*) upisati u datoteku *Constants.js* koja se nalazi na putanji *IzvorniKod/trueBlood/frontend/src/components* na liniji *export const SPRINGURL = 'http://localhost:8080'*.



Slika 5.27: Constants.js datoteka

Nakon modificiranja i spremanja datoteke Constants.js, potrebno je otvoriti novi cmd prozor, pozicionirati se u frontend folder (putanja *IzvorniKod/trueBlood/frontend*) te izvršiti komandu *npm install* kojom se instaliraju potrebni node moduli. Zatim komandu *npm run build* kojom se kreira folder u kojem se nalaze izvršne datoteke spremne za korištenje u produkciji. Te datoteke mogu se upogoniti u bilo kojem proizvoljnom web serveru (poput microsoftovog IIS-a), no radi jednostavnosti, nakon izvršenja prethodne naredbe, naredbom *npm install -g serve* dodatno se instalira *serve* modul, koji koristi za posluživanje prije kreiranih datoteka. Napokon, naredbom *serve -s build -l 80* pokreće se Node.js web server na portu 80, te je ovime kompletna aplikacija spremna za upotrebu.

5.4.7 Napomene

Ukoliko je uključen, potrebno je odabrane portove frontend i backend dijela aplikacije propustiti kroz firewall.

Ukoliko nakon pokretanja backend ili frontend dijela aplikacije neki od njih izgledno ne radi, moguće da se slučajnim lijevim klika mišom na cmd prozor (koji ga pokreće taj dio aplikacije zamrznuo. Potrebno je na taj isti prozor pritisnuti desni klik miša, što će nastaviti rad zamrznutog dijela aplikacije. Preporuča se minimiziranje cmd prozora kako ne bi došlo do tog problema.

6. Zaključak i budući rad

Naš zadatak bio je izraditi banku krvi koju bi primarno koristili donori i djelatnici banke.

Ideja banke krvi i upravljanje količinama krvi je sveprisutna u svijetu, a dostupna je i na mrežnim stranicama [HZTM](http://hztm.hr)¹.

Projekt smo proveli u tri faze:

1. početna razrada funkcionalnih i nefunkcionalnih zahtjeva
2. implementacija zahtjeva i daljnje razrađivanje
3. testiranje i završno dokumentiranje

Na početku prve faze upoznivali smo ideju aplikacije, razrađivali smo funkcionalne zahtjeve i dogovarali se što sve aplikacija može i treba raditi. U prvoj fazi smo se također počeli upoznavati s tehnologijama koje smo koristili. Ni jedan od članova tima nije bio u potpunosti upoznat s tim tehnologijama. Naučili smo da u stvarnosti, prije izrade samog projekta, potrebno je naučiti kako učiti alate, a zatim i naučiti sam alat.

U drugoj fazi projekta krenuli smo sa implementacijom, što je na početku teklo jako sporo jer još nismo bili upoznati s alatima, no kako je vrijeme prolazilo tako smo postajali bolji i implementirali smo zahtjeve sve brže. Također, u ovoj fazi uvidjeli smo propuste u definicijama funkcionalnih i nefunkcionalnih zahtjeva, pa smo naučili biti precizniji u svojim definicijama i uzimati više slučajeva u obzir.

U trećoj, odnosno posljednjoj fazi, nakon implementacije programskog rješenja, dovršili smo dokumentaciju projekta. Ovaj korak nam je dodatno pomogao u shvaćanju obujma našeg projekta, i ukazao nam na arhitekturne probleme koje nismo predvidjeli. Također, testiranjem smo ustanovili da sustav ima par grešaka.

Članovi tima su prije projekta bili upoznati s Javom i NodeJS-om, pa bi odabirom tih tehnologija umjesto Reacta pomogao pri ubrzanju razvoja aplikacije.

Naučili smo važnost koordinacije i komunikacije među članovima tima. Nekad članovi nisu bili dobro koordinirani pa bi jedni implementirali funkcionalnost na

¹<http://hztm.hr/hr/content/22/zalihe-krvi/831/zalihe-krvi>

ovaj, a drugi na onaj način. Naučili smo meke vještine povezane s projektima općenito, kao što je korištenje alata `git`, pisanje značajnih commit poruka, korištenje GitLab-a, te poštivanje unaprijed definiranih obrazaca programiranja.

Najviše od svega, naučili smo značenje vremenske koordiniranosti u grupnom radu. Projekti poput ovoga vremenski su zahtjevni i ponekad se može dogoditi da jedan modul programskog rješenja kasni za drugim, pa drugi ne može nastaviti svoj razvoj. U ovakvim slučajevima, iznimno je važno vremenski se uskladiti unutar tima kako bi se takve neučinkovitosti izbjegle.

Od definiranih funkcionalnih zahtjeva uspjeli smo implementirati sve koju su bili predstavljeni zadatkom.

U budućnosti bismo mogli ažurirati aplikaciju i nadodati neke nove stvari. Primjerice, mogli bi donirati ljudi iz nekih drugih zemalja, pa bi aplikaciju preveli na engleski jezik. Također dodali bismo da djelatnik i administrator mogu biti ujedno i donori, što trenutno nije slučaj.

Zaključno, zahvaljujući iskustvima i znanjima koja smo stekli na ovom projektu, kad bismo krenuli iz početka, napravili bismo mnogo bolji projekt mnogo brže.

7. Popis literature

Kontinuirano osvježavanje

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

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

Indeks slika i dijagrama

2.1	Primjer javno dostupnih podataka o stanju banke krvi Hrvatskog zavoda za transfuzijsku medicinu	6
2.2	Primjer web sučelja logiranog korisnika Američkog crvenog križa	7
3.1	Dijagram obrasca uporabe, funkcionalnost neregistriranog / neprijavljenog korisnika i donora	20
3.2	Dijagram obrasca uporabe, funkcionalnost djelatnika	21
3.3	Dijagram obrasca uporabe, funkcionalnost administratora	22
3.4	Dijagram obrasca uporabe, funkcionalnost sustava za automatske poslove	23
3.5	Sekvencijski dijagram za UC8	24
3.6	Sekvencijski dijagram za UC9	25
3.7	Sekvencijski dijagram za UC12	26
3.8	Sekvencijski dijagram za UC14	27
4.1	Arhitektura sustava	29
4.2	Podjela slojeva	31
4.3	MVC model	32
4.4	relacijski model baze podataka	38
4.5	Dijagram razreda koji opisuje model	40
4.6	Dijagram razreda koji opisuju repozitorije	42
4.7	Dijagram razreda koji opisuju kontrolere	43
4.8	Dijagram razreda koji opisuju DTO-ove	44
4.9	Dijagram stanja	45
4.10	Dijagram aktivnosti	47
4.11	Dijagram komponenti	48
5.1	Unit test1	50
5.2	Unit test2	51
5.3	Unit test3	51
5.4	Unit test4	52

5.5	Unit test5	52
5.6	Unit test6	53
5.7	Rezultati ispitivanja komponenti	54
5.8	Funkcija za usporavanje sustava	55
5.9	Selenium test1	56
5.10	Selenium test2	57
5.11	Selenium test3	58
5.12	Selenium test4	59
5.13	Funkcija main	60
5.14	Rezultat ispitivanja sustava	60
5.15	Dijagram razmještaja	61
5.16	Download JDK17 Installera	63
5.17	Download Maven Binary-a	63
5.18	Prozor pokreni	64
5.19	Prozor naprednih postavka sustava	65
5.20	Prozor varijabli okruženja	66
5.21	Nova putanja u PATH varijabli okruženja	67
5.22	application.properties datoteka	68
5.23	Kopiranje javne IPv4 adrese servera	69
5.24	Pozicioniranje i izvršavanje komandi	70
5.25	Završni izlaz nakon uspješnog builda aplikacije	70
5.26	Punjenje baze statičkim podacima	71
5.27	Constants.js datoteka	72
7.1	Aktivnosti na main grani	82
7.2	Aktivnosti članova 1	82
7.3	Aktivnosti članova 2	83

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 14. listopada 2021.
- Prisustvovali: Svi
- Teme sastanka:
 - inicijalni dogovor s CROZ djelatnicima

2. sastanak

- Datum: 21. listopada 2021.
- Prisustvovali: Svi
- Teme sastanka:
 - dogovor oko podjele zadataka
 - podijeljeni zadaci iz dokumentacije na sedmero ljudi

3. sastanak

- Datum: 28. listopada 2021.
- Prisustvovali: Jurinić, Kerman
- Teme sastanka:
 - pripremljeni obrasci uporabe
 - podjela oko crtanja i opisivanja dijagrama

4. sastanak

- Datum: 05. studenoga 2021.
- Prisustvovali: Jurinić, Hudiček, Vugrinec, Šlezak, Kerman
- Teme sastanka:
 - klijentski dio aplikacije
 - dogovor oko poslužitelja

5. sastanak

- Datum: 11. studenoga 2021.
- Prisustvovali: Jurinić, Hudiček, Vugrinec, Šlezak, Okreša, Kerman
- Teme sastanka:
 - poslužiteljski dio aplikacije, dogovor oko dizajna baze podataka

6. sastanak

- Datum: 12. studenoga 2021.
- Prisustvovali: Kerman
- Teme sastanka:
 - konzultacije s asistentom i demosom oko obrazaca uporabe i baze podataka

7. sastanak

- Datum: 17. studenoga 2021.
- Prisustvovali: svi
- Teme sastanka:
 - demonstracija aplikacije asistentu i demosu

8. sastanak

- Datum: 9. prosinca 2021.
- Prisustvovali: svi
- Teme sastanka:
 - dogovor oko daljnjih aktivnosti

9. sastanak

- Datum: 22. prosinca 2021.
- Prisustvovali: svi
- Teme sastanka:
 - demonstracija alfa inačice aplikacije asistentu i demosu

Tablica aktivnosti

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

	David Kerman	Ivan Jurinić	Matija Vugrinec	Jakov Šlezak	Matej Hudiček	Marko Okreša	Josip Pardon
Upravljanje projektom	10						
Opis projektnog zadatka				4			
Funkcionalni zahtjevi	2	2					
Opis pojedinih obrazaca	6	6					
Dijagram obrazaca	4	2					
Sekvencijski dijagrami					4		
Opis ostalih zahtjeva						2	
Arhitektura i dizajn sustava	5						
Baza podataka			10				
Dijagram razreda	3				5		6
Dijagram stanja	2						
Dijagram aktivnosti	3						
Dijagram komponenti	1						
Korištene tehnologije i alati	2						
Ispitivanje programskog rješenja	5						5
Dijagram razmještaja	1						
Upute za puštanje u pogon							6
Dnevnik sastajanja	1						

Nastavljeno na idućoj stranici

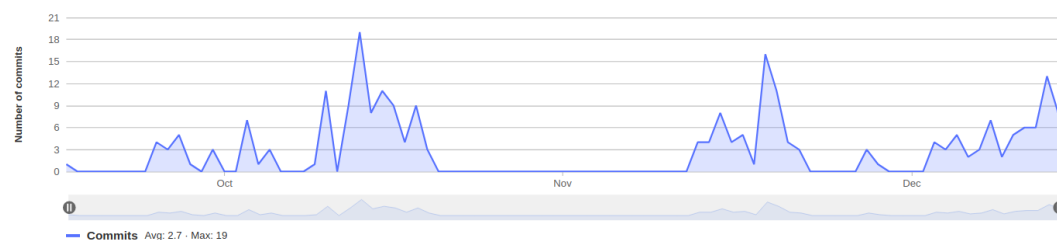
Nastavljeno od prethodne stranice

	David Kerman	Ivan Jurinić	Matija Vugrinec	Jakov Šlezak	Matej Hudiček	Marko Okreša	Josip Pardon
Zaključak i budući rad	1						
Popis literature	1						
frontend		20			45		30
backend	40		40	45		30	
baza podataka			5				

Dijagrami pregleda promjena

Commits to main

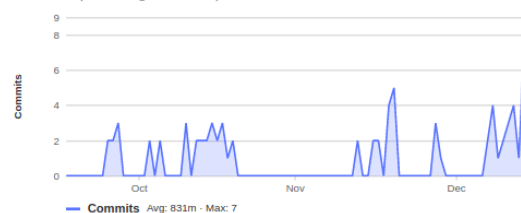
Excluding merge commits. Limited to 6,000 commits.



Slika 7.1: Aktivnosti na main grani

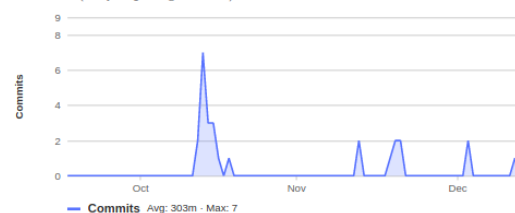
David Kerman

74 commits (dk522014@outlook.com)



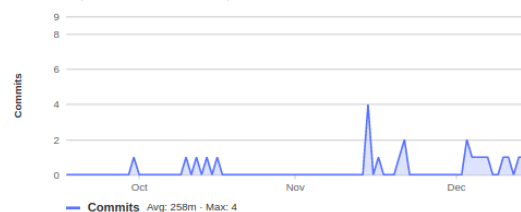
matijavugrinec

27 commits (matija.vugrinec@net-net.hr)



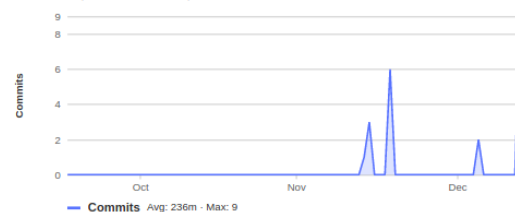
Matej Hudiček

23 commits (matej.hudicek@gmail.com)



Jakov

21 commits (jakov.slezak@fer.hr)



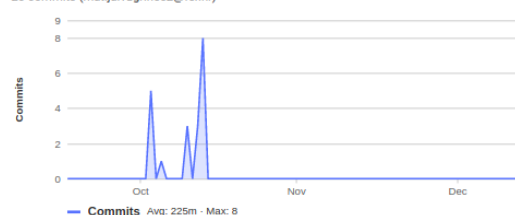
David Kerman

20 commits (dk52201@fer.hr)



Matija Vugrinec

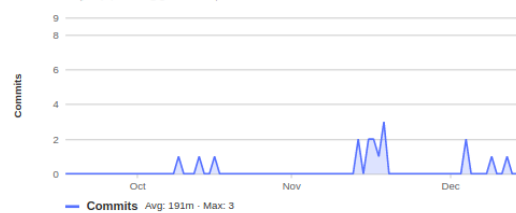
20 commits (matija.vugrinec2@fer.hr)



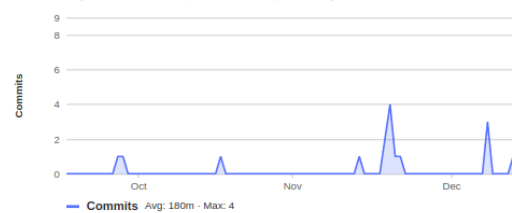
Slika 7.2: Aktivnosti članova 1

Josip Pardon

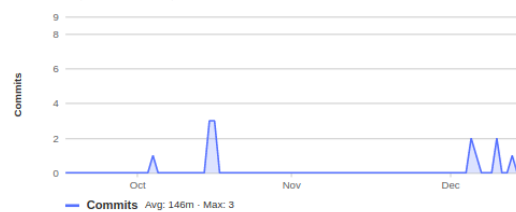
17 commits (josip.pardon@gmail.com)

**Jurinich**

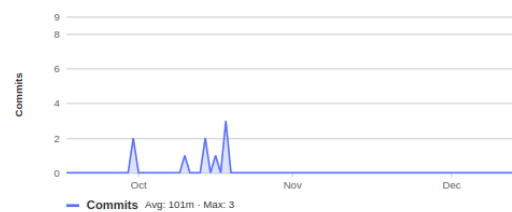
16 commits (92642291+Jurinich@users.noreply.github.com)

**okresa.m**

13 commits (mo52775@fer.hr)

**Jakov Šlezak**

9 commits (jakovslezak@gmail.com)



Slika 7.3: Aktivnosti članova 2