

Programsko inženjerstvo

Ak. god. 2021./2022.

TrueBlood

Dokumentacija, Rev. 2

Grupa: *MEGATRON*

Voditelj: *Toni Ivankačić*

Datum predaje: *19. studenoga 2021.*

Nastavnik: *Ivan Lovrić*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
2.1 Cilj projekta i domena primjene	5
2.2 Postojeća rješenja	5
2.3 Odabir tehnologije	6
2.4 Detaljan opis sustava	7
2.5 Prepoznati i razriješeni rizici	8
2.6 Ograničenja u implementaciji	10
2.7 Mogućnosti za nadogradnju	11
3 Specifikacija programske potpore	15
3.1 Funkcionalni zahtjevi	15
3.1.1 Obrasci uporabe	18
3.1.2 Sekvencijski dijagrami	39
3.2 Ostali zahtjevi	42
4 Arhitektura i dizajn sustava	44
4.1 Baza podataka	45
4.1.1 Opis tablica	46
4.1.2 Dijagram baze podataka	49
4.2 Dijagram razreda	50
4.3 Dijagram stanja	55
4.4 Dijagram aktivnosti	56
4.5 Dijagram komponenti	59
5 Implementacija i korisničko sučelje	61
5.1 Korištene tehnologije i alati	61
5.2 Ispitivanje programskog rješenja	63
5.2.1 Ispitivanje komponenti	63
5.2.2 Ispitivanje sustava	67

5.3 Dijagram razmještaja	70
5.4 Upute za puštanje u pogon	71
6 Zaključak i budući rad	75
6.1 Analiza zadatka	75
6.2 Postignuti rezultati	75
6.3 Izazovi u radu	75
6.4 Mogućnosti za daljnji rad	76
Popis literature	77
Indeks slika i dijagrama	79
Dodatak: Prikaz aktivnosti grupe	80

1. Dnevnik promjena dokumentacije

Table 1.1: Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Toni	28.10.2021.
0.2	Započeto pisanje opisa projektnog zadatka	Toni	3.11.2021.
0.3	Napravljeni inicijalni dijagrami obrazaca uporabe	Toni	4.11.2021.
0.4	Popravljeni dijagrami obrazaca uporabe, dodani sekvencijski dijagrami	Toni	6.11.2021.
0.5	Dodani opisi dijagrama obrazaca uporabe	Toni	7.11.2021.
0.6	Započeto pisanje poglavlja o arhitekturi sustava	Jakov, Luka	12.11.2021.
0.7	Uređivanje dosadašnjeg dokumenta, razrada ostalih zahtjeva	Toni	14.11.2021.
0.8	Dovršen opis projektnog zadatka, dodane reference među dijelovima dokumenta, dodan dnevnik sastajanja	Toni	16.11.2021.
0.9	Dodani dnevnički sastajanja u Dodatak (1-5)	Dora	16.11.2021.
0.10	Preuređeni opisi UC 1-8, izmijenjeni UC na dijagramu <i>Uređivanje postojećih računa</i>	Toni	17.11.2021.
0.11	Preuređeni opisi UC 8-15, dodan UC 3.1, izmijenjeni UC na dijagramu, napisani rizici u sustavu i optionalne funkcionalnosti u poglavljima 2.5 i 2.7	Toni	18.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.12	Dodani dnevnički sastajanja (6-7)	Dora	18.11.2021.
0.13	Popravljeno poglavlje Arhitektura po komentarima mentora (model i opisi te same tablice baze podataka)	Jakov	19.11.2021.
0.14	Dodano poglavlje 4.1. Baza podataka, detaljnije opisano poglavlje 2.3 Odabir tehnologije	Ana	19.11.2021.
0.15	Preuređen UC3.1	Marko	19.11.2021.
0.16	Dodano poglavlje 4.2. Dijagram razreda	Jakov	19.11.2021.
1.0	Uređena tablica aktivnosti	Toni, Jakov, Ana, Luka, Marko, Borna, Dora	19.11.2021.
1.1	Popravak poglavlja 4.2.	Jakov	3.1.2022.
1.2	Dodana poglavlja 4.3 Dijagram stanja i 4.4 Dijagram aktivnosti te započeto pisanje poglavlja 5. Implementacija	Dora	10.1.2022.
1.3	Sitni prepravci obrazaca uporabe kako bi odgovarali stvarnom stanju sustava	Toni	11.1.2022.
1.4	Pisanje poglavlja o tehnologijama i alatima i puštanju aplikacije u pogon	Dora	13.1.2022.
2.0	Dodvršeno poglavlje implementacija, dodani sati aktivnosti i grafovi aktivnosti, provjera pravopisa	Dora, Jakov, Toni	14.1.2022.

2. Opis projektnog zadatka

2.1 Cilj projekta i domena primjene

Cilj projekta TrueBlood je napraviti programsku potporu te korisničko sučelje za sustav dobrovoljnog darivanja krvi. Web aplikacija nudi mogućnost stvaranja korisničkih računa za darivatelje (donore), evidenciju pokušaja doniranja krvi, evidenciju trenutnih zaliha krvi, i mnoge druge funkcionalnosti koje su opisane u nastavku. Svrha aplikacije je obogatiti iskustvo darivanja te potaknuti ljude da na jednostavniji način pristupe procesu i dobrom djelom pomognu opterećenom zdravstvenom sustavu. Aplikacija velik dio funkcionalnosti radi automatski, pa se time rasterećuje osoblje i ubrzava i poboljšava cjelokupni proces.

S obzirom na domenu u kojoj se ova aplikacija može primijeniti, potencijalni zainteresirani naručitelji bili bi zavod za javno zdravstvo, ministarstvo zdravstva, Crveni križ, ili slična institucija koja se masovno bavi transfuzijskom medicinom te ima potrebu za automatizacijom procesa darivanja krvi i održavanja stanja zaliha krvi.

2.2 Postojeća rješenja

Dio funkcionalnosti koje razvijamo u ovom projektu već postoji u sustavu Hrvatskog zavoda za transfuzijsku medicinu (<https://www.hztm.hr>), gdje je također moguće vidjeti trenutne zalihe krvi u odnosu na optimalne razine, a sustav ispisuje i upozorenje o nedostatku pojedinih krvnih grupa. S druge strane, sustav HZTM-a ne nudi funkcionalnost opisanu u našem zadatku, gdje se donori mogu sami prijaviti u sustav te imati uvid u svoje prethodne pokušaje darivanja, a ne nudi ni slanje potvrda mailom ni naknadno ispisivanje potvrda. Ipak, sustav HZTM-a poslužio nam je kao dobra baza iz koje smo mogli vidjeti dosadašnju realizaciju bar dijela sustava.



Slika 2.1: Postojeća stranica HZTM-a

2.3 Odabir tehnologije

Aplikacija je izrađena u razvojnog okviru *Java Spring Boot* s poslužiteljske strane, a u *Reactu* sa klijentske strane. Za pohranu, organizaciju i pristup podacima sustav koristi PostgreSQL bazu podataka pokrenutu u Docker spremniku. Aplikacija je puštena u pogon preko javno dostupne cloud platforme Heroku. Navedeni razvojni okviri nude sve funkcionalnosti potrebne za izradu opisanog sustava, što će biti detaljnije opisano u poglavljju 4 i u drugoj reviziji u poglavljju *Implementacija*.

2.4 Detaljan opis sustava

Aplikaciji mogu pristupiti i prijavljeni i neprijavljeni korisnici koji imaju različite ovlasti u sustavu.

Neprijavljeni korisnici imaju uvid u trenutne razine zaliha krvi i mogućnost prijaviti se ili se registrirati te time steći ovlasti prijavljenog korisnika.

Prijavljeni korisnici po razini ovlasti dijele se na doneure, djelatnike banke i administratore.

Donori se mogu registrirati samostalno i prije darivanja krvi na javnoj stranici aplikacije. Ako im račun pak stvori djelatnik banke na prvom darivanju krvi, donori ne moraju prolaziti kroz proces registracije, već se prijavljuju s postojećim podatcima.

U svakom slučaju, pri stvaranju bilo kojeg računa u sustavu na e-adresu šalje se poruka s generiranim identifikatorom, inicijalnom lozinkom i poveznicom za aktivaciju. Nakon pristupa poveznici za aktivaciju, račun je aktiviran i može se koristiti.

Nakon prijave u sustav, donori imaju pristup svim prošlim pokušajima darivanja krvi, imaju uvid u trenutno stanje zalihe njihove krvne grupe i imaju mogućnost izmijeniti neki od matičnih ili kontakt podataka svog računa (kao npr. adresu, broj telefona, e-adresu i sl.). Donori ne mogu mijenjati zdravstvene podatke računa, kao što su krvna grupa i trajno odbijanje. Donorima se također nudi mogućnost preuzimanja potvrde o bilo kojem prijašnjem darivanju krvi.

Djelatnici banke prijavom u sustav mogu mijenjati svoje matične i kontakt podatke, uređivati sve podatke donora, evidentirati slanje određenog broja doza krvi u vanjsku instituciju (bolnicu) te evidentirati novi pokušaj darivanja krvi. Pri evidenciji pokušaja darivanja, koji se tipično radi na akciji darivanja krvi, djelatnik banke evidentira potrebne zdravstvene podatke kao i uspješnost pokušaja darivanja te, ukoliko zabilježeni podatci sadrže neke od podataka koji to impliciraju, evidentira trajno odbijanje donora u njegovom računu.

Administratori sustava u sustavu imaju ovlast kreirati nove račune djelatnika banke, deaktivirati bilo koji račun u sustavu (u slučaju pogrešnog stvaranja, smrti ili prekida radnog odnosa) te definirati optimalne razine zaliha krvi za svaku krvnu grupu. Optimalne razine služe svim korisnicima sustava da jednostavnije zaključe značenje trenutnih zaliha krvi. Optimalne granice također služe i za okidanje slanja upozorenja korisnicima sustava.

Kada razina zalihe krvi pojedine krvne grupe padne ispod donje optimalne granice, svi djelatnici banke u sustavu dobivaju upozorenje o tome, a upozorenje dobivaju i zabilježeni donori te krvne grupe.

Upozorenje se izdaje i prijelazom gornje optimalne granice, ali samo djelatnicima banke.

Donori pak dodatnu obavijest dobivaju kada istekne njihov period zabrane darivanja. Naime, iz zdravstvenih razloga, donori ne smiju prečesto darivati krv, pa za muškarce minimalni period između dva darivanja krvi iznosi tri mjeseca, a za žene četiri mjeseca. Kako donori ne bi zaboravili na istek tog perioda zabrane, sustav im po isteku tog perioda šalje obavijest.

S obzirom na to da se pri stvaranju bilo kojeg računa lozinka generira automatski, svi prijavljeni korisnici sustava imaju mogućnost promijeniti svoju lozinku u nešto što će lakše zapamtiti.

2.5 Prepoznati i razriješeni rizici

Mnogi rizici u radu sustava i njegovom korištenju prepoznati su unaprijed i razriješeni. Neki rizici proizlaze iz domene primjene, a neki iz predviđanja nesavršenog ponašanja u radu sa sustavom. Neki od prepoznatih rizika i njihova implementirana rješenja ili razlozi odbacivanja su:

- Korisnik sustava zaboravlja svoj userId (donorId, bankworkerId) iako već ima stvoren račun
 - U e-poruci s poveznicom za aktivaciju navedeni su i donorId i inicijalna lozinka, pa se korisnik pronalaskom aktivacijske poruke može podsjetiti zaboravljenih podataka
 - Djelatnik banke donora i administrator djelatnika banke može pretražiti po bilo kojoj kombinaciji podataka, npr. imena, prezimena ili OIB-a. Na taj način može se pronaći račun u sustavu i podsjetiti donorIdja.
- Donor pri stvaranju računa nema neki podatak
 - Kao obavezni podatci označeni su samo oni koje bi većina ljudi trebala znati ili imati uvijek dostupno
 - Ako donor račun stvara samostalno, uvijek može odgoditi stvaranje računa dok ne pronađe obavezni podatak koji mu nedostaje

- Neobavezni podaci ne moraju se unijeti pri stvaranju računa pa ih donor može unijeti i kasnije prijavom u sustav, ili pri idućem doniranju
- Podatci koji se unesu za donora mogu zabunom biti semantički neispravni (tipfeleri pri unosu adrese, broja telefona, ...) što validacija sustava ne može prepoznati
 - Pri svakom doniranju, djelatnik banke provjerava unesene podatke s donorom te ih popravlja u slučaju greške
 - Donor pogrešne podatke može i samostalno ispraviti prijavom u sustav i pregledom osobnih podataka
- Sustav se može zablokirati zbog neadekvatnog sklopolja pa se neke aktivnosti ne mogu pravovremeno obaviti
 - U ovakvom ekstremnom slučaju može se ručno voditi evidencija podataka koji su neophodni za obaviti trenutno dok se problem ne riješi pa se podatci unesu naknadno, a aktivnosti koje nije nužno obaviti u istom trenutku mogu se odgoditi do oporavka sustava
- Baza podataka pri pokušaju unosa nema dovoljno prostora za pohranu novih zapisa
 - Sustav je dovoljno malen da ovo odstupanje ne predstavlja značajan rizik
 - Administrator sustava mogao bi s vremena na vrijeme voditi računa o popunjenoosti diska s bazom podataka i tako na vrijeme spriječiti nastanak opisane situacije
 - Rizik se može na vrijeme spriječiti većom investicijom u opremu
- Neovlašteni korisnik može nasumično pogoditi aktivacijsku poveznicu i aktivirati račun bez dopuštenja
 - Aktivacijska poveznica dovoljno je dugačka da ovo ne predstavlja značajan rizik
- Djelatnik banke želi promijeniti donorId nekog donora, a donorId je primarni ključ u bazi podataka (isto vrijedi i za odnos administrator - djelatnik banke)

- Primarni ključevi samo se ispisuju na ekran pri pregledu i uređivanju podataka računa i ne mogu se mijenjati.
- Nije prepoznata funkcionalno nužna situacija kada bi ova funkcionalnost bila potrebna, osim iz *estetskih razloga*
- Djelatnik banke na doniranju pogrešno tumači neki zdravstveni podatak i pogrešno evidentira trajno odbijanje donora koji ne bi trebao biti trajno odbijen
 - Trajno odbijanje, kao i ostali podatci, može se naknadno ispraviti u sustavu ako se ispostavi da je riječ o pogrešci
- Donor prešućuje neki od podataka zbog kojih ne bi trebao moći darivati krv
 - Sustav nikako ne može otkriti ovu informaciju, već je ovo problem iz domene primjene (koji je većinom riješen time da se doniranje krvi ne plaća i da se ampula krvi dodatno testira na krvnu grupu i patogene)

2.6 Ograničenja u implementaciji

S obzirom na ograničen vremenski rok i ograničene resurse za izradu sustava, na brojnim mjestima u dogовору с асистентом dogovoren je nužan i dovoljan opseg rješenja. Taj opseg definiran je funkcionalnim i nefunkcionalnim zahtjevima, dok su dodatne stvari ostavljene na slobodu timu za implementaciju.

Dio problema sa sustavom proizlazi iz ograničenih resursa dostupnih za puštanje aplikacije u pogon na besplatnom servisu *Heroku*, koji gasi aplikaciju nakon određenog vremena u kojem ne primi nijedan zahtjev. Zbog toga neke funkcionalnosti (poput slanja obavijesti o ponovnoj mogućnosti darivanja) neće raditi pouzdano. Kako bi se ove funkcionalnosti ispitale treba prvo napraviti neki zahtjev i na klijentski poslužitelj i na poslužitelj poslužiteljske aplikacije kako bi se oni pokrenuli, a nakon toga u redovnom korištenju aplikacije ne bi trebalo biti problema. Kako se obavijest o ponovnoj mogućnosti darivanja šalje svaki dan u 12:00 za sve donore koji su donirali krv tri/četiri mjeseca ranije, radi ispitivanja potrebno je netom prije 12:00 poslati bilo kakav zahtjev poslužiteljskoj aplikaciji kako bi se ona pokrenula (i naravno, imati donora u sustavu kojemu na taj dan istječe period nemogućnosti darivanja).

Potencijalan problem predstavlja i servis e-pošte, koji u nekom trenutku može onemogućiti korištenje njegovih usluga zbog sigurnosnih razloga (previše poruka

i slično). Oba problema bila bi riješena posjedovanjem vlastitog poslužitelja koji bi aplikaciju držao stalno upaljenom i na kojemu bi se mogao pokrenuti vlastiti servis za e-poruke.

Pri izradi sustava razmatrane su brojne dodatne i alternativne mogućnosti i scenariji koji u ovoj inačici aplikacije nisu podržane. Takve mogućnosti dokumentirane su u sljedećem odjeljku pa se u nekoj od budućih implementacija sustav može nadograditi da podržava neke od navedenih alternativnih mogućnosti. Te mogućnosti najčešće uključuju mogućnosti koje bi olakšale korisnicima korištenje sustava i dale dodatne funkcionalnosti koje se trenutno mogu izvoditi zaobilazno ili se zasad uopće ne mogu izvoditi.

2.7 Mogućnosti za nadogradnju

S obzirom na prepoznate rizike vezane za olakšanje korištenja sustava i izvođenje dosad nemogućih radnji, osmišljene su sljedeće mogućnosti za buduću nadogradnju:

- Korisnik sustava pri stvaranju računa ima zabilježenu pogrešnu e-adresu pa ne dobije poruku s aktivacijskom poveznicom, a novi račun ne može stvoriti jer su u sustavu već zabilježeni podatci koji moraju biti jedinstveni
ILI
Korisnik je zagubio poruku s aktivacijskom poveznicom i donorId-jem prije aktivacije računa (ili ju nije primio zbog prepunjene memorije pretinca e-pošte)
 - Moguća je nadogradnja koja bi djelatniku banke / administratoru pri pregledu računa donora / djelatnika banke ostavila mogućnost ponovnog slanja poruke s aktivacijskom poveznicom uz eventualni ispravak e-adrese
 - Moguća je nadogradnja koja bi ostavila mogućnost stvaranja novog računa s istim podatcima sve dok se račun ne aktivira te bi se neaktivirani račun s istim podatcima automatski obrisao
- Korisnik sustava želi ostati prijavljen odmah po otvaranju aktivacijske poveznice, bez ponovnog upisa podataka za prijavu
 - Iako je ovo samo jednokratno produljenje procesa koje ne utječe značajno na iskustvo korisnika (koji, ako ima aktivacijsku poveznicu, sigurno ima

i podatke za prijavu jer se nalaze u istoj poruci), moguća je nadogradnja koja bi stvorila sjednicu korisnika odmah nakon pristupa poveznici za aktivaciju

- Administrator zabunom deaktivira račun nekog korisnika sustava
 - Moguća je nadogradnja koja bi administratoru omogućila reaktivaciju nekog računa u sustavu
- Donor svoju krvnu grupu može ne znati unaprijed, a u sustavu je pretpostavljeno da se podatak može odmah otkriti iako je uobičajena praksa da se krv testira naknadno
 - Krvna grupa može ostati nepotpunjena pa se popuniti nakon što se obavi test na bilo koji način
- Doza krvi na serološkom testiranju može završiti kao nepovoljna i mora se odbaciti
 - Moguća je nadogradnja gdje bi u sustavu postojala i evidencija pojedinih doza krvi koje se onda mogu evidentirati kao nepovoljne i povezati s donorima
 - Trenutno moguće rješenje je vođenje evidencije o dozama van sustava i ručno povezivanje s donorima
- Administrator nije definirao optimalne granice zaliha pa stanje zaliha ne prikazuje podatke koji vizualno imaju korisno značenje
 - Moguća je nadogradnja u kojoj bi sustav administratoru slao upozorenja dok ne definira granice
- Korisnik sustava zaboravio je lozinku pa ne može pristupiti računu
 - Moguća je nadogradnja koja bi, u slučaju promijenjene lozinke koja se ne može pronaći u e-poruci s aktivacijskom poveznicom, omogućila promjenu lozinke putem e-adrese koja je navedena u računu ili slanje postojeće lozinke na tu e-adresu
- Donor želi potvrdu o doniranju poslanu na e-adresu umjesto preuzimanja

- Moguća je nadogradnja koja bi uz preuzimanje ponudila i mogućnost slanja potvrde na e-adresu, vrlo slično aktivnosti opisanoj u koraku 8 obrasca uporabe UC6
- Donor želi potvrdu o doniranju i za neuspješan pokušaj doniranja
 - Moguća je nadogradnja koja bi omogućila generiranje PDF potvrde za sve pokušaje doniranja, ne samo za uspješne
- U sustavu je zabilježena pogrešna e-adresa donora na koju će se slati poruka s aktivacijskom poveznicom i PDF potvrde o doniranjima
 - PDF potvrda ne sadrži nikakve osjetljive podatke pa ona ne predstavlja značajan rizik za slanje na tuđu e-adresu
 - Na idućem doniranju krvi moguće je izmijeniti e-adresu na koju će se slati potvrde i (kako je već opisano u jednoj od prethodnih dodatnih mogućnosti) poslati novu aktivacijsku poveznicu
- Stanje zaliha krvi u sustavu nije u skladu sa stanjem zaliha u skladištu
 - Moguća je nadogradnja koja bi administratoru omogućila *rekalibraciju* stanja (trenutno se problem djelomično riješiti ručnim smanjivanjem stanja zaliha krvi)
- Čestim slanjem doza krvi u bolnice i novim doniranjem može doći do čestog prelaska donje granice i prečestog slanja obavijesti
 - Moguća je nadogradnja koja bi uvela period vremena u kojemu se korisnicima ne bi slale poruke češće od npr. svaka dva dana
- Donori koji su donirali krv u posljednja 3/4 mjeseca nikako ne mogu ponovno donirati krv i obavijest im je uglavnom nepotrebna
 - Moguća je nadogradnja koja bi zaustavila slanje obavijesti o nedostatku krvi za vrijeme nedopuštenog perioda darivanja
 - Moguća je nadogradnja koja bi omogućila potpunu odjavu s upozorenja
- Baza podataka može propustiti stvoriti okidač za slanje upozorenja donorima kojima je upravo istekao nedopušteni period darivanja

- Iako svatko može na profilu vidjeti kada je zadnji put darivao krv, moguća je nadogradnja koja bi za svakog donora čuvala je li mu poslano upozorenje ili ne (koje bi se resetiralo na *ne* pri svakom novom doniranju), pa bi se onda upozorenje slalo svim korisnicima kojima je prošlo više od 3 ili 4 mjeseca od prošlog darivanja, a obavijest im nije poslana (pa bi se proces slanja mogao pokretati i rjeđe)
- S obzirom na potencijalne zainteresirane strane kao što je zavod za javno zdravstvo, nepraktično je što sustav nije povezan s postojećim sustavima koji se koriste u zdravstvu
 - Sustav bi u budućoj implementaciji mogao podržati i spajanje s postojećim sustavima koji se već uhodano koriste u zdravstvu. Tako bi se obogatile funkcionalnosti oba softvera i stvorilo još veće rasterećenje zdravstvenom sustavu.
- Jedan administrator u sustavu mogao bi biti nedovoljan za obavljanje posla administracije računa
 - Sustav bi u budućoj implementaciji mogao podržati stvaranje novog računa administratora pa bi se tako posao mogao podijeliti na više osoba
- Korisnici bi mogli htjeti promijeniti inicijalnu lozinku, bilo radi lakšeg pamćenja, ili zbog sigurnosti
 - Sustav bi u budućoj implementaciji mogao podržati promjenu lozinke - funkcionalnost već postoji sa poslužiteljske strane, ali se može napraviti i sa kljentske strane

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj projekta (zavod za javno zdravstvo, Crveni križ...)
2. Razvojni tim
3. Donori
4. Djelatnici banke krvi
5. Administratori sustava
6. Svi korisnici interneta koji koriste javnu stranicu sustava radi pregleda trenutne razine krvi

Aktori i njihovi funkcionalni zahtjevi:

1. Administrator (inicijator) može:
 - (a) Administrirati korisničke račune:
 - i. Dodati račun djelatnika banke
 - ii. Deaktivirati bilo koji račun u sustavu
 - iii. Definirati optimalne granice razina krvi
2. Djelatnik banke (inicijator) može:
 - (a) Prikupiti podatke o donoru
 - i. Prikupiti matične i kontakt podatke
 - ii. Prikupiti zdravstvene podatke (krvna grupa)
 - (b) Stvoriti korisnički račun donora
 - (c) Nadopuniti podatke postojećeg računa donora (uključujući i zdravstvene podatke)
 - (d) Evidentirati pokušaj doniranja
 - i. Zabilježiti trenutne zdravstvene podatke o donoru
 - ii. Zabilježiti uspješnost pokušaja doniranja

- (e) Evidentirati trajno odbijanje donora (postojanje + razlog)
- (f) Povećati zalihu krvi u sustavu uspješnim doniranjem
- (g) Evidentirati slanje krvi u vanjsku instituciju (bolnicu)

3. Djelatnik banke (sudionik) može:

- (a) Primiti obavijest o prekoračenju bilo koje od optimalnih granica zaliha krvi
- (b) Primiti aktivacijski link radi aktivacije računa

4. Donor (inicijator) može:

- (a) Krearati svoj korisnički račun prije prvog darivanja
- (b) Nadopuniti podatke računa (osim zdravstvenih podataka - krvne grupe i trajnog odbijanja)
- (c) Pregledati povijest darivanja krvi
- (d) Preuzeti PDF potvrdu o postojećem doniranju krvi

5. Donor (sudionik) može:

- (a) Primiti aktivacijski link radi aktivacije računa
- (b) Dobiti generirani donorID
- (c) Primiti upozorenje o prekoračenju donje optimalne granice zaliha krvi (ako donor nije trajno odbijen)
- (d) Pri spajanju u sustav dobiti informaciju o trenutnim zalihama krvi svoje krvne grupe (ako donor nije trajno odbijen)
- (e) Primiti obavijest o isteku perioda zabrane darivanja u trajanju od 3 do 4 mjeseca od zadnjeg darivanja
- (f) Primiti e-mail s PDF potvrdom o darivanju (nakon darivanja i nakon vlastitog iniciranja izdavanja potvrde)

6. Korisnik javnog weba (inicijator) može:

- (a) Krearati novi korisnički račun donora
- (b) Pregledati trenutno stanje zaliha krvi svake krvne grupe

7. Baza podataka (inicijator) može:

- (a) Izdati obavijest o isteku nedopuštenog perioda darivanja u trajanju od tri do četiri mjeseca

8. Baza podataka (sudionik) može:

(a) Pohranjivati:

- Postojeće račune u sustavu
- Podatke o donorima
- Podatke o djelatnicima banke
- Količine doza krvi svake krvne grupe u sustavu
- Pokušaje doniranja i sve podatke vezane uz njih
- Potrošnju krvi u sustavu

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

*Baza podataka sudionik je u svim obrascima uporabe, stoga ju ne navodimo za-sebno u svakom obrascu (ako je u nekom obrascu posebno navedena, to je kako bi se naglasilo da je u tom obrascu baza podataka glavni sudionik)

UC1.1 - Stvori novi račun donora

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Stvaranje novog računa donora u sustavu pri doniranju krvi
- **Sudionici:** donor
- **Preduvjet:** Donor nema stvoren račun, djelatnik banke je prijavljen u sustav (UC3.1)
- **Opis osnovnog tijeka:**
 1. Djelatnik banke otvara obrazac za stvaranje pokušaja doniranja
 2. Djelatnik banke otvara obrazac za stvaranje računa donora
 3. Djelatnik banke unosi potrebne podatke o donoru koje saznaje ispitivanjem (matični i kontakt podatci, krvna grupa)
 4. Djelatnik banke pokreće stvaranje računa klikom na gumb *Kreiraj račun*
 5. Sustav validira podatke
 6. Sustav stvara račun u bazi podataka, baza podataka generira donorID
 7. Djelatniku banke na ekran se ispisuje poruka o uspješno izrađenom računu
 8. Pokreće se UC2 (*Pošalji e-mail za aktivaciju* - novom donoru)
- **Opis mogućih odstupanja:**
 - 2.a Donor nema / ne zna neki od obveznih podataka
 1. Proces se neuspješno završava
 - 2.b Donor ne zna svoju krvnu grupu
 1. Djelatnik banke testom određuje krvnu grupu donora
 2. Djelatnik banke upisuje krvnu grupu u račun donora
 3. Proces kreiranja računa nastavlja se gdje je i stao na koraku 2 osnovnog tijeka
 - 4.a Dani podatci su sintaksno pogrešni
 1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka

4.b Unesen je podatak koji je već evidentiran kod nekog drugog donora, a radi se o podatku koji mora biti jedinstven

1. Sustav na ekran ispisuje poruku o nejedinstvenosti unesenog podatka
2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC1.2 - Stvori novi račun sebi

- **Glavni sudionik:** donor
- **Cilj:** Osobno stvaranje novog računa donora u sustavu
- **Sudionici:**
- **Preduvjet:** Donor nema stvoren račun, donor ima pristup internetu
- **Opis osnovnog tijeka:**
 1. Donor otvara obrazac za registraciju
 2. Donor unosi potrebne podatke (matični i kontakt podaci, ne i krvna grupa)
 3. Donor pokreće stvaranje računa klikom na gumb *Kreiraj račun*
 4. Sustav validira podatke
 5. Sustav stvara račun u bazi podataka, baza podataka generira donorID
 6. Donoru se na ekran ispisuje poruka o uspješno izrađenom računu
 7. Pokreće se UC2 (*Pošalji e-mail za aktivaciju* - novom donoru)
- **Opis mogućih odstupanja:**
 - 2.a Donor nema / ne zna neki od obveznih podataka
 1. Proces se neuspješno završava
 - 4.a Dani podatci su sintaksno pogrešni
 1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
- 4.b Unesen je podatak koji je već evidentiran kod nekog drugog donora, a radi se o podatku koji mora biti jedinstven
 1. Sustav na ekran ispisuje poruku o nejedinstvenosti unesenog podatka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC1.3 - Stvori novi račun djelatnika banke

- **Glavni sudionik:** administrator
- **Cilj:** Stvaranje novog računa djelatnika banke u sustavu

- **Sudionici:** djelatnik banke
- **Preduvjet:** Djelatnik banke nema stvoren račun, administrator ima sve potrebne podatke, administrator je prijavljen u sustav (UC3.1)
- **Opis osnovnog tijeka:**
 1. Administrator otvara obrazac za stvaranje računa djelatnika banke
 2. Administrator unosi potrebne podatke (matične i kontakt podatke djelatnika banke)
 3. Administrator pokreće stvaranje računa klikom na gumb *Kreiraj račun*
 4. Sustav validira podatke
 5. Sustav stvara račun u bazi podataka, baza podataka generira bankworkerID
 6. Administratoru se na ekran ispisuje poruka o uspješno izrađenom računu djelatnika
 7. Pokreće se UC2 (*Pošalji e-mail za aktivaciju* - novom djelatniku banke)
- **Opis mogućih odstupanja:**
 - 2.a Administrator nema / ne zna neki od obveznih podataka
 1. Proces se neuspješno završava
 - 4.a Dani podatci su sintaksno pogrešni
 1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
 - 4.b Unesen je podatak koji je već evidentiran kod nekog drugog djelatnika banke, a radi se o podatku koji mora biti jedinstven
 1. Sustav na ekran ispisuje poruku o nejedinstvenosti unesenog podatka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC2 - Pošalji e-mail za aktivaciju

- **Glavni sudionik:** korisnik sustava (donor, djelatnik banke ili administrator)
- **Cilj:** Poslati e-mail s korisničkim podatcima i poveznicom za aktivaciju računa
- **Sudionici:**
- **Preduvjet:** Novi korisnik pokrenuo je stvaranje računa; sustav ima generirane userID i inicijalnu lozinku
- **Opis osnovnog tijeka:**
 1. Sustav generira poveznicu za aktivaciju računa
 2. Sustav generira e-poruku s navedenom poveznicom za aktivaciju te inicijalnim podatcima za prijavu

3. Sustav na e-adresu definiranu pri kreiranju računa šalje e-poruku s generiranim podatcima

- **Opis mogućih odstupanja:**

UC3 - Aktiviraj svoj račun

- **Glavni sudionik:** novi korisnik sustava (donor, djelatnik banke ili administrator)
- **Cilj:** Aktivirati korisnički račun radi buduće prijave u sustav
- **Sudionici:**
- **Preduvjet:** Novi korisnik sustava ima generiran račun, korisnički podaci dostavljeni su na e-adresu, korisnik sustava ima pristup korisničkim podatcima
- **Opis osnovnog tijeka:**
 1. Korisnik pristupa poveznici za aktivaciju računa
 2. Sustav u bazi podataka evidentira račun kao aktiviran
- **Opis mogućih odstupanja:**

UC3.1 - Prijavi se u sustav

- **Glavni sudionik:** korisnik sustava
- **Cilj:** Prijaviti se u sustav i dobiti pristup značajkama dostupnima samo prijavljenim korisnicima
- **Sudionici:**
- **Preduvjet:** Korisnik sustava ima generiran račun, korisnik sustava ima pristup korisničkim podatcima
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici pritišće gumb *Prijavi se*
 2. Korisnik u polja za prijavu unosi svoj userID (donorID, bankworkerID) i lozinku
 3. Korisnik pritiskom na gumb *Prijava* pokreće proces prijave u sustav
 4. Sustav provjerava ispravnost podataka i nakon potvrde vraća kolačić s identifikatorom sjednice
 5. Sustav preusmjerava korisnika na početnu stranicu profila
- **Opis mogućih odstupanja:**
 - 4.a Korisnik nema aktiviran račun
 - (a) Sustav na ekran ispisuje poruku o nemogućnosti prijave zbog neaktiviranog računa

- (b) Proces se prekida
- 4.b Korisnik je unio neki od pogrešnih podataka za prijavu
- (a) Sustav na ekran ispisuje poruku o nemogućnosti prijave zbog neispravnih podataka za prijavu
 - (b) Proces se nastavlja na koraku 2 osnovnog tijeka

UC4 - Pregledaj svoje podatke

- **Glavni sudionik:** donor
- **Cilj:** Pregledati podatke računa trenutnog korisnika
- **Sudionici:** djelatnik banke
- **Preduvjet:** Trenutni korisnik ima račun i prijavljen je u sustav (UC3.1)
- **Opis osnovnog tijeka:**
 1. Korisnik na podstranici *Profil* pritišće gumb *Uredi podatke*
 2. Sustav učitava stranicu s postojećim podatcima i gumbom za uređivanje podataka
- **Opis mogućih odstupanja:**

UC4.1 - Pronađi i pregledaj podatke donora

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Pronaći donora u sustavu i pregledati njegove podatke
- **Sudionici:** administrator
- **Preduvjet:** Donor ima generiran račun, trenutni korisnik je prijavljen u sustav (UC3.1) i ima dovoljno podataka o donoru kako bi ga pronašao u sustavu
- **Opis osnovnog tijeka:**

*Korisnikom se smatra djelatnik banke ili administrator

1. Korisnik pristupa obrascu za traženje donora klikom na gumb *Pronađi donora* na stranici za stvaranje pokušaja doniranja ili na stranici za deaktivaciju računa
2. Korisnik u obrazac unosi neki od podataka donora (donorID, OIB, ime, prezime) ili više njih u kombinaciji, odvojene razmakom
3. Sustav iz baze podataka dobavlja podatke o traženim donorima
4. Sustav u tabličnom prikazu ispisuje sve doneure kojima se osobni podatci podudaraju s traženima
5. Korisnik odabire željenog donora klikom na željeni redak
6. Sustav preusmjerava korisnika na prethodnu stranicu s koje je došao, pritom noseći i podatak o odabranom donoru

- **Opis mogućih odstupanja:**

- 3.1 Sustav od baze podataka ne dobiva podatke jer ne postoji donor s navedenim podatcima
 1. Sustav na ekran ispisuje da nema pronađenih rezultata
 2. Izvođenje se nastavlja na koraku 2 osnovnog tijeka
- 3.2 Traženi račun je trajno deaktiviran
 1. Sustav iz baze podataka dobavlja podatke samo o nedeaktiviranim korisnicima
 2. Baza podataka ne pronalazi zapis traženog donora
 3. Sustav na ekran ispisuje da nema pronađenih rezultata
 4. Izvođenje se nastavlja na koraku 2 osnovnog tijeka

UC4.2 - Pronađi i pregledaj podatke djelatnika banke

- **Glavni sudionik:** administrator

- **Cilj:** Pronaći djelatnika banke u sustavu i pregledati njegove podatke

- **Sudionici:**

- **Preduvjet:** Djelatnik banke ima generiran račun, administrator je prijavljen u sustav (UC3.1), administrator ima dovoljno podataka o djelatniku banke kako bi ga pronašao u sustavu

- **Opis osnovnog tijeka:**

1. Administrator pristupa obrascu za traženje djelatnika banke klikom na gumb *Pronađi djelatnika* sa stranice za deaktivaciju računa
2. Administrator u obrazac unosi neki od podataka donora (donorID, OIB, ime, prezime) ili više njih u kombinaciji, odvojene razmakom
3. Sustav iz baze podataka dobavlja podatke o traženim djelatnicima banke
4. Sustav u tabličnom prikazu ispisuje sve djelatnike banke kojima se osobni podatci podudaraju s traženima
5. Administrator odabire želenog djelatnika banke klikom na željeni redak
6. Sustav preusmjerava administratora na prethodnu stranicu s koje je došao, pritom noseći i podatak o odabranom djelatniku banke

- **Opis mogućih odstupanja:**

- 3.1 Sustav od baze podataka ne dobiva podatke jer ne postoji djelatnik banke s navedenim podatcima
 1. Sustav na ekran ispisuje poruku o neuspješnom dohvaćanju djelatnika banke

2. Izvođenje se nastavlja na koraku 2 osnovnog tijeka

3.2 Traženi račun je trajno deaktiviran

1. Sustav iz baze podataka dobavlja podatke samo o nedeaktiviranim korisnicima
2. Baza podataka ne pronalazi zapis traženog djelatnika banke
3. Sustav na ekran ispisuje da nema pronađenih rezultata
4. Izvođenje se nastavlja na koraku 2 osnovnog tijeka

UC5 - Uredi matične i kontakt podatke donora

- **Glavni sudionik:** donor
- **Cilj:** Uređivanje svojih postojećih matičnih i kontakt podataka
- **Sudionici:**
- **Preduvjet:** Donor ima stvoren račun i nalazi se na stranici za uređivanje osobnih podataka (vidi UC4 (*Pregledaj svoje podatke*))
- **Opis osnovnog tijeka:**
 1. Donor izmjenjuje potrebne matične ili kontakt podatke (ali ne i zdravstvene podatke)
 2. Donor pritišće gumb *Pohrani promjene*
 3. Sustav validira ispravnost podataka
 4. Sustav ažurira postojeći zapis u bazi podataka
- **Opis mogućih odstupanja:**
 - 4.a Donor želi promijeniti OIB na vrijednost koja više ne bi bila jedinstvena
 1. Sustav ispisuje poruku o nejedinstvenosti podatka te odbija spremiti promjene
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
 - 4.b Dani podatci su sintaksno pogrešni
 1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC5.1 - Uredi zdravstvene, matične i kontakt podatke donora

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Uređivanje postojećih podataka o računu donora
- **Sudionici:** donor
- **Preduvjet:** Donor ima stvoren račun, djelatnik banke ga je pronašao koristeći UC4.1 (*Pronadi i pregledaj podatke donora*)
- **Opis osnovnog tijeka:**

1. Djelatnik banke sa stranice za stvaranje pokušaja donacije pristupa obrascu za izmjenu podataka donora klikom na gumb *Uredi podatke*
 2. Djelatnik banke izmjenjuje bilo koji potrebni podatak o donoru (uključujući i zdravstvene podatke)
 3. Djelatnik banke pritišće gumb *Pohrani promjene*
 4. Sustav validira ispravnost podataka
 5. Sustav ažurira postojeći zapis u bazi podataka
- **Opis mogućih odstupanja:**
 - 4.a Djelatnik banke želi promijeniti OIB donora na vrijednost koja više ne bi bila jedinstvena
 1. Sustav ispisuje poruku o nejedinstvenosti podatka te odbija spremiti promjene
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
 - 4.b Dani podatci su sintaksno pogrešni
 1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
 2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC5.2 - Uredi podatke djelatnika banke

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Uređivanje već postojećih podataka
- **Sudionici:**
- **Preduvjet:** Djelatnik banke ima stvoren račun i nalazi se na stranici za uređivanje osobnih podataka (vidi UC4 (*Pregledaj svoje podatke*))
- **Opis osnovnog tijeka:**
 1. Djelatnik banke izmjenjuje potrebni podatak
 2. Djelatnik banke pritišće gumb *Pohrani promjene*
 3. Sustav validira ispravnost podataka
 4. Sustav ažurira postojeći zapis u bazi podataka
- **Opis mogućih odstupanja:**
 - 4.a Djelatnik banke želi promijeniti OIB na vrijednost koja više ne bi bila jedinstvena
 1. Sustav ispisuje poruku o nejedinstvenosti podatka te odbija spremiti promjene
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
 - 4.b Dani podatci su sintaksno pogrešni

1. Sustav na ekran ispisuje poruku o neispravnosti unesenih podataka
2. Proces se nastavlja na koraku 2 osnovnog tijeka

UC5.3 - Deaktiviraj račun

- **Glavni sudionik:** administrator
- **Cilj:** Deaktivacija postojećeg računa
- **Sudionici:**
- **Preduvjet:** Korisnički račun koji se treba deaktivirati postoji
- **Opis osnovnog tijeka:**
 1. Administrator pristupa stranici za deaktivaciju računa pritiskom na gumb *Upravljam računima*
 2. Administrator koristi tražilicu kako bi pronašao traženi račun pritiskom na gumb *Pronađi donora* ili *Pronađi djelatnika* (vidi UC4.1 (*Pronađi i pregledaj podatke donora*) ili UC4.2 (*Pronađi i pregledaj podatke djelatnika banke*))
 3. Administrator nakon pronalaska računa i povratka na stranicu za deaktivaciju pritišće gumb *Deaktiviraj račun*
 4. Sustav u bazi podataka evidentira da je račun deaktiviran (ali račun se ne briše)
- **Opis mogućih odstupanja:**

UC6 - Stvori pokušaj doniranja

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Stvaranje pokušaja doniranja, evidentiranje trenutnih zdravstvenih podataka
- **Sudionici:** donor
- **Preduvjet:** Donor je došao na doniranje i ima izrađen korisnički račun
- **Opis osnovnog tijeka:**
 1. Djelatnik banke otvara obrazac za unos novog pokušaja darivanja
 2. Djelatnik banke pronalazi donora u sustavu koristeći tražilicu pritiskom na gumb *Pronađi donora* - vidi UC4.1 (*Pronađi i pregledaj podatke donora*)
 3. Djelatnik banke u sustavu ispunjava zdravstveni upitnik prema odgovorima koje mu daje donor
 4. Djelatnik banke pritiskom na gumb *Spremi pokušaj doniranja* pokreće proces spremanja
 5. Donor odlazi darivati krv, a pokušaj doniranja evidentira se kao uspješan

6. Sustav u bazi podataka stvara novi pokušaj doniranja s prethodno navedenim podatcima
 7. Sustav u bazi podataka evidentira povećanje zalihe krvi krvne grupe donora iz ovog pokušaja za jednu dozu
 8. Sustav generira PDF potvrdu o darivanju krvi
 9. Djelatniku banke na ekran se ispisuje poruka o uspješnom darivanju krvi
 10. Sustav šalje potvrdu na mail (vidi UC12 za sličnu funkcionalnost gdje se potvrda preuzima na računalo)
- **Opis mogućih odstupanja:**
 - 4.a Neki od zdravstvenih podataka implicira privremeno odbijanje donora
 1. Sustav pokušaj doniranja evidentira kao neuspješan
 2. Obavlja se korak 6 osnovnog tijeka (spremanje pokušaja doniranja)
 3. Djelatniku banke se na ekran ispisuje poruka o neuspješnom darivanju krvi
 4. Proces završava
 - 4.b Neki od zdravstvenih podataka implicira trajno odbijanje donora
 1. Sustav pokušaj doniranja evidentira kao neuspješan
 2. Sustav evidentira trajno odbijanje u računu donora s razlogom odbijanja poslanom u obrascu
 3. Obavlja se korak 6 osnovnog tijeka (spremanje pokušaja doniranja)
 4. Djelatniku banke se na ekran ispisuje poruka o neuspješnom darivanju krvi
 5. Proces završava
 - 7 S novom dozom krvi prekoračuje se gornja optimalna granica zaliha doza krvi
 1. Pokreće se UC14 (*Izdaj upozorenje o prekoračenju optimalne granice*)
 2. Proces se nastavlja gdje je i stao na koraku 7 osnovnog tijeka

UC7 - Pregledaj zalihe svih krvnih grupa

- **Glavni sudionik:** korisnik interneta
- **Cilj:** Na javnom webu pregledati stanja svih krvnih grupa i dobiti informaciju o stanju zaliha svake grupe u odnosu na optimalne granice
- **Sudionici:**
- **Preduvjet:**

- **Opis osnovnog tijeka:**

1. Korisnik interneta pristupa početnoj stranici aplikacije
2. Sustav dohvaća trenutna stanja zaliha krvi i optimalne granice za svaku krvnu grupu iz baze podataka
3. Sustav vizualno prikazuje stanja zaliha svih krvnih grupa i njihovih optimalnih granica tako da je gornja optimalna granica na 90% visine, donja optimalna granica i trenutna razina krvi proporcionalno tome (donja optimalna granica na visini od $\text{donjaGranica}/\text{gornjaGranica} * 90\%$, a trenutna razina krvi na visini od $\text{razina}/\text{gornjaGranica} * 90\%$)
4. Sustav provjerava nalaze li se sva stanja zaliha između optimalnih granica

- **Opis mogućih odstupanja:**

- 1 Neko stanje zaliha nalazi se ispod donje optimalne granice za tu krvnu grupu
 1. Sustav na stranici ispisuje i poruku o nedostatku krvnih grupa čije je stanje ispod donje optimalne granice
 2. Proces završava

UC9 - Definiraj optimalne granice zaliha krvi

- **Glavni sudionik:** administrator

- **Cilj:** Postaviti gornje i donje optimalne granice za svaku krvnu grupu

- **Sudionici:**

- **Preduvjet:** Administrator ima korisnički račun i prijavljen je u sustav (UC3.1)

- **Opis osnovnog tijeka:**

1. Administrator pristupa obrascu za postavljanje optimalnih granica pritiskom na gumb *Uredi granice*
2. Administrator za svaku krvnu grupu podešava gornje i donje optimalne granice zaliha krvi
3. Administrator pokreće proces spremanja pritiskom na gumb *Spremi granice*
4. Sustav spremi promjene u bazu podataka

- **Opis mogućih odstupanja:**

- 3 Administrator podešava novu donju optimalnu granicu iznad trenutne zalihe krvi, ili gornju optimalnu granicu ispod trenutne zalihe krvi za bilo koju krvnu grupu
 1. Pokreće se UC14 (*Izdaj upozorenje o prekoračenju optimalne granice*)

2. Proces se nastavlja gdje je i stao na koraku 3 osnovnog tijeka

UC10 - Ispiši poruku o stanju zalihe krvne grupe donora

- **Glavni sudionik:** donor
- **Cilj:** Informirati donora o okvirnom stanju zaliha njegove krvne grupe
- **Sudionici:**
- **Preduvjet:** Donor je prijavljen u sustav (UC3.1) i nedostaje njegove krvne grupe
- **Opis osnovnog tijeka:**
 1. Donor dolazi na početnu stranicu prijavljenog korisnika (profil)
 2. Sustav na ekranu prikazuje poruku o nedostatku zalihe te krvne grupe
- **Opis mogućih odstupanja:**
 - 2 Stanje zaliha krvi je iznad donje optimalne granice
 1. Poruka se ne ispisuje
 2. Proces završava
 - 2 Stanje zaliha krvi je ispod donje optimalne granice
 1. Sustav ispisuje dodatno istaknutu poticajnu poruku za darivanje krvi
 2. Proces završava

UC11 - Pregledaj povijest pokušaja doniranja

- **Glavni sudionik:** donor
- **Cilj:** Dati donoru uvid u sve svoje prijašnje pokušaje doniranja i njihov ishod, te ponuditi mogućnosť preuzimanja potvrde
- **Sudionici:** djelatnik banke
- **Preduvjet:** Donor je prijavljen u sustav (UC3.1) / djelatnik banke je pronašao donora u sustavu pri darivanju
- **Opis osnovnog tijeka:**

*Korisnikom se smatra donor ili djelatnik banke opisani u preduvjetima

1. Korisnik otvara stranicu s povijesti pokušaja doniranja pritiskom na gumb *Moje donacije / Prošle donacije*
2. Sustav za svaki pokušaj doniranja navodi osnovne podatke definirane u ostalim zahtjevima
3. Sustav za svaki uspješni pokušaj stvara i gumb *Preuzmi*

- **Opis mogućih odstupanja:**
 - 2 Korisnik nema nijedan prethodan pokušaj doniranja

1. Sustav ispisuje informativnu poruku o nepostojanju prijašnjih darivanja
 2. Proces završava
- 3 Korisnik pritišeće gumb *Preuzmi* za neki pokušaj doniranja
1. Pokreće se (UC12 - *Preuzmi PDF potvrdu*) od koraka 2 osnovnog tijeka

UC12 - Preuzmi PDF potvrdu

- **Glavni sudionik:** donor
- **Cilj:** Preuzimanje PDF potvrde o pokušaju doniranja iz povijesti doniranja u sustavu
- **Sudionici:** djelatnik
- **Preduvjet:** Donor se nalazi na stranici s prethodnim pokušajima doniranja i ima obavljen barem jedno uspješno doniranje / djelatnik banke pronašao je donora u sustavu i pristupio je stranici s njegovim prijašnjim donacijama (vidi UC11)
- **Opis osnovnog tijeka:**
*Korisnikom se smatra donor ili djelatnik banke opisani u preduvjetima
 1. Korisnik pritišeće gumb *Preuzmi PDF potvrdu*
 2. Sustav generira PDF potvrdu o doniranju
 3. Sustav pokreće preuzimanje PDF potvrde na računalo korisnika
- **Opis mogućih odstupanja:**

UC13 - Evidentiraj slanje krvi u bolnicu

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Evidentirati smanjenje zalihe krvi radi slanja u bolnicu
- **Sudionici:**
- **Preduvjet:** Djelatnik banke prijavljen je u sustav (UC3.1)
- **Opis osnovnog tijeka:**
 1. Djelatnik banke u sustavu otvara obrazac za slanje krvi u bolnicu pritiskom na gumb *Slanje krvi*
 2. Djelatnik banke popunjava broj doza koji se šalje od svake krvne grupe
 3. Djelatnik banke inicira smanjenje zalihe krvi pritiskom na gumb *Pošalji*
 4. Sustav u bazi podataka smanjuje zalihe krvi za odgovarajuće iznose
- **Opis mogućih odstupanja:**

- 3 Djelatnik banke popunio je da se za neku krvnu grupu šalje više doza krvi nego što je postoji u sustavu
 1. Sustav djelatniku banke ispisuje poruku upozorenja o nemogućnosti slanja tolikog broja doza krvi
 2. Proces se nastavlja na koraku 2 osnovnog tijeka
- 4 Novo stanje krvi za bilo koju krvnu grupu nalazi se ispod, a prije je bilo iznad donje optimalne granice
 1. Pokreće se UC14 (*Izdaj upozorenje o prekoračenju optimalne granice*)

UC14 - Izdaj upozorenje o prekoračenju optimalne granice

- **Glavni sudionik:** djelatnik banke
- **Cilj:** Upozoriti djelatnike banke u slučaju prekoračenja optimalnih granica zaliha krvi te upozoriti donore u slučaju prekoračenja donje optimalne granice
- **Sudionici:** donor
- **Preduvjet:** U sustavu postoje definirane optimalne granice zaliha krvi; dogodio se bar jedan od tri okidača u alternativnim slijedovima sljedećih obrazaca uporabe:
 1. Evidentiranje slanja krvi u bolnicu (vidi UC13)
 2. Redefiniranje optimalnih granica (vidi UC9)
 3. Dodavanje nove doze krvi u sustav (vidi UC6)

U osnovnom slijedu, pretpostavlja se da je zaliha krvi pala ispod donje optimalne granice:

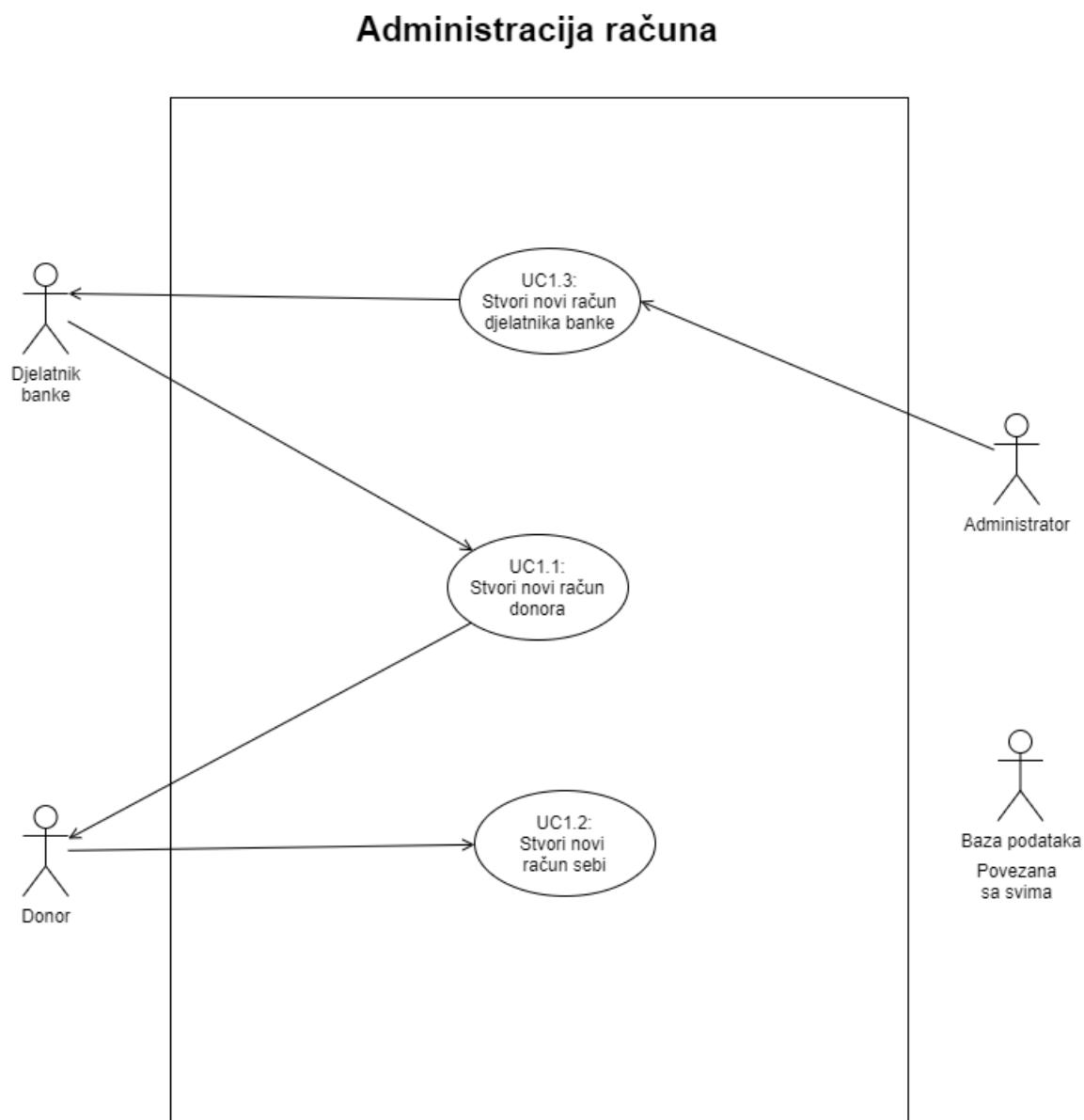
- **Opis osnovnog tijeka:**
 1. Sustav dohvaća e-adrese svih djelatnika banke iz baze podataka i izdaje im e-mail upozorenje o nedovoljnem stanju zaliha te krvne grupe
 2. Sustav dohvaća e-adresu svih donora te krvne grupe koji nisu trajno odbijeni iz baze podataka i izdaje e-mail upozorenje o nedovoljnem stanju zaliha te krvne grupe s poticajem na doniranje
- **Opis mogućih odstupanja:**

Zaliha krvi nije pala ispod donje optimalne granice nego je narasla iznad gornje optimalne granice

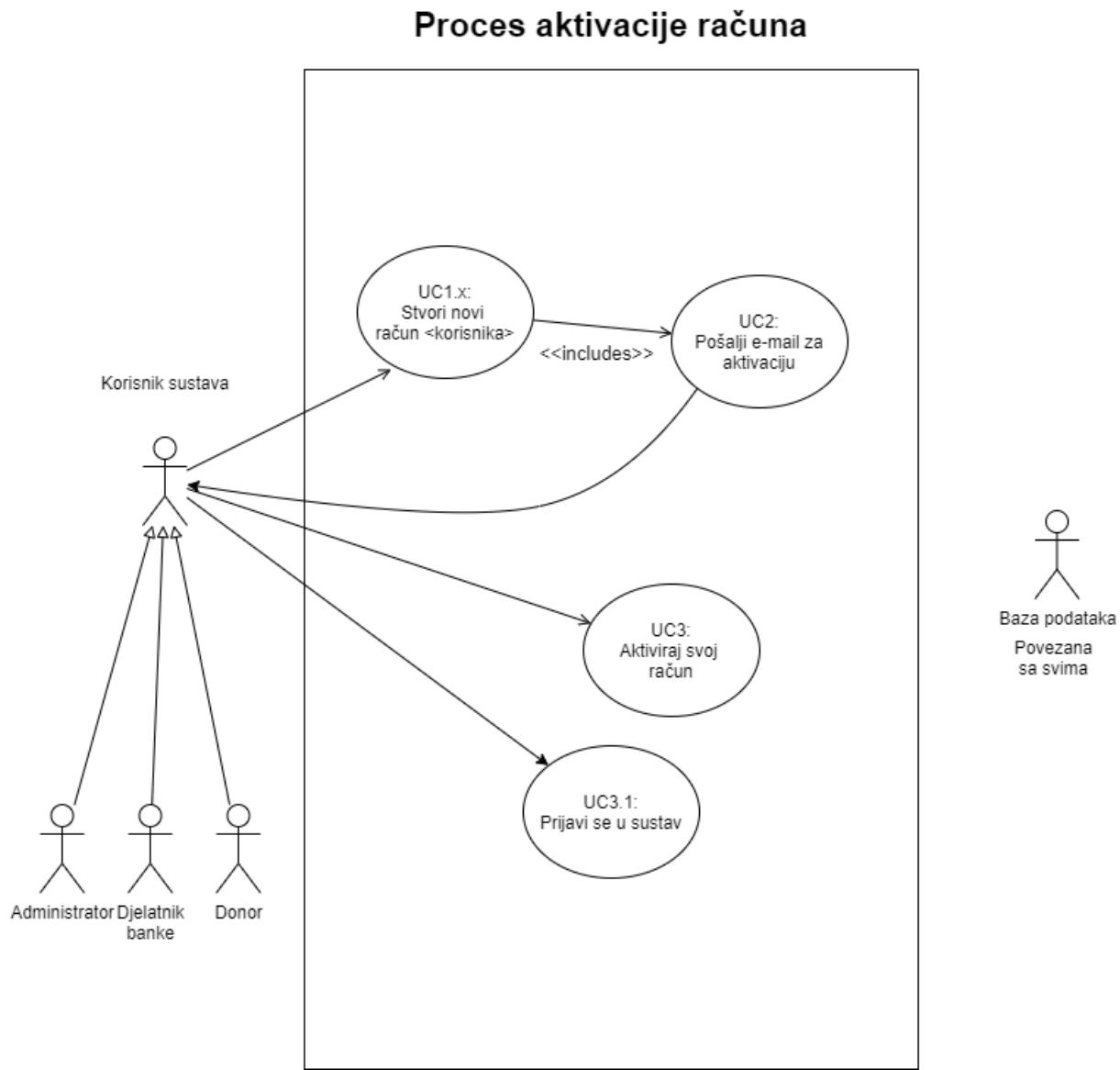
1. Sustav dohvaća e-adrese svih djelatnika banke iz baze podataka i izdaje im e-mail upozorenje o prevelikom stanju zaliha te krvne grupe
2. Proces završava

UC15 - Izdaj obavijest o isteku perioda nemogućnosti doniranja

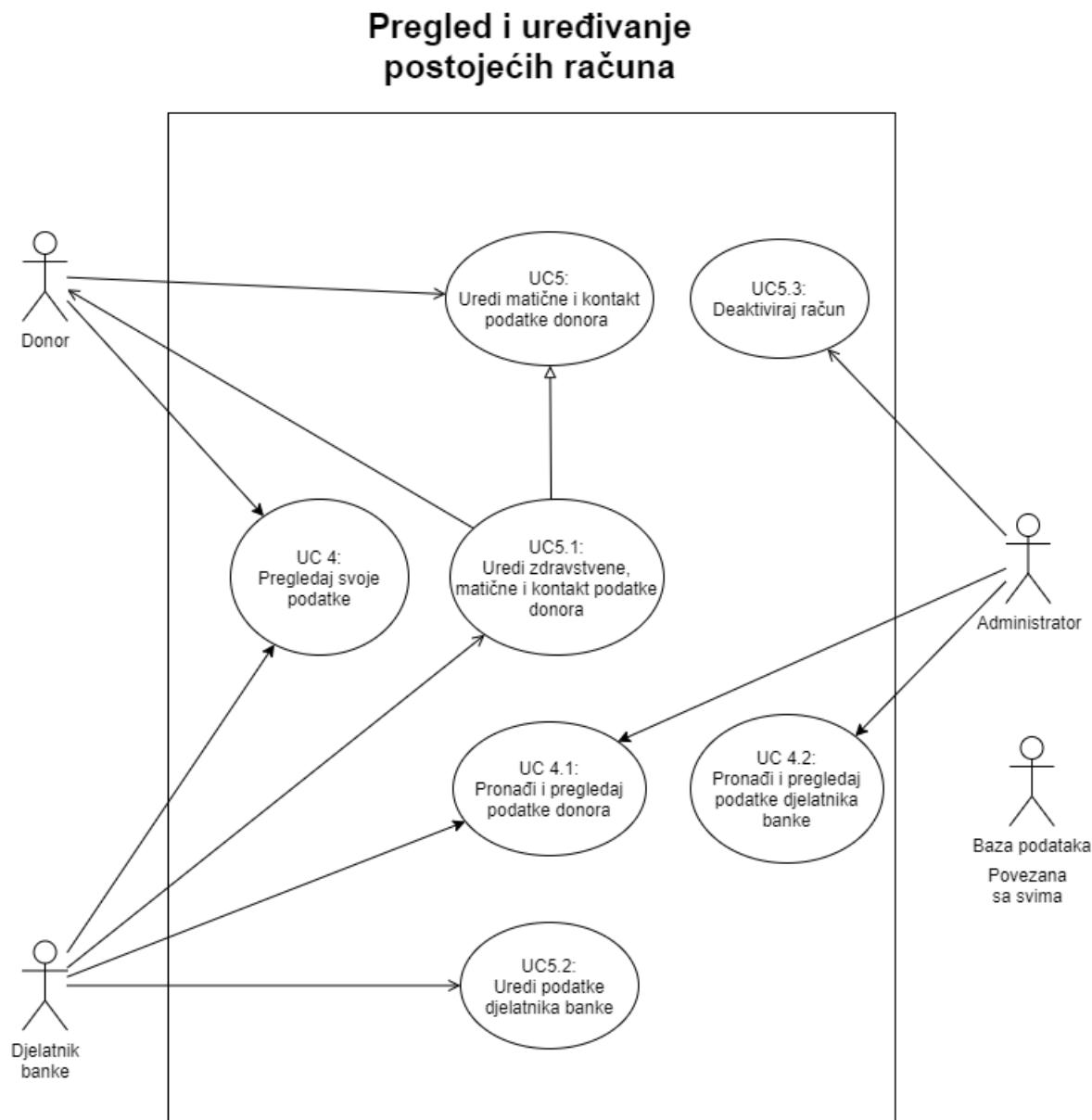
- **Glavni sudionik:** baza podataka
- **Cilj:** Obavijestiti donora o ponovnoj mogućnosti doniranja krvi
- **Sudionici:** donor
- **Preduvjet:** Donor je već darivao krv, započinje dan na datum na koji donoru istječe period nemogućnosti doniranja
- **Opis osnovnog tijeka:**
 1. Sustav pokreće proces u kojemu računa periode od posljednjeg darivanja do današnjeg dana za svakog donora
 2. Sustav generira i šalje e-poruke za svakog donora kojemu je na taj dan istekao period nemogućnosti darivanja
- **Opis mogućih odstupanja:**

Dijagrami obrazaca uporabe

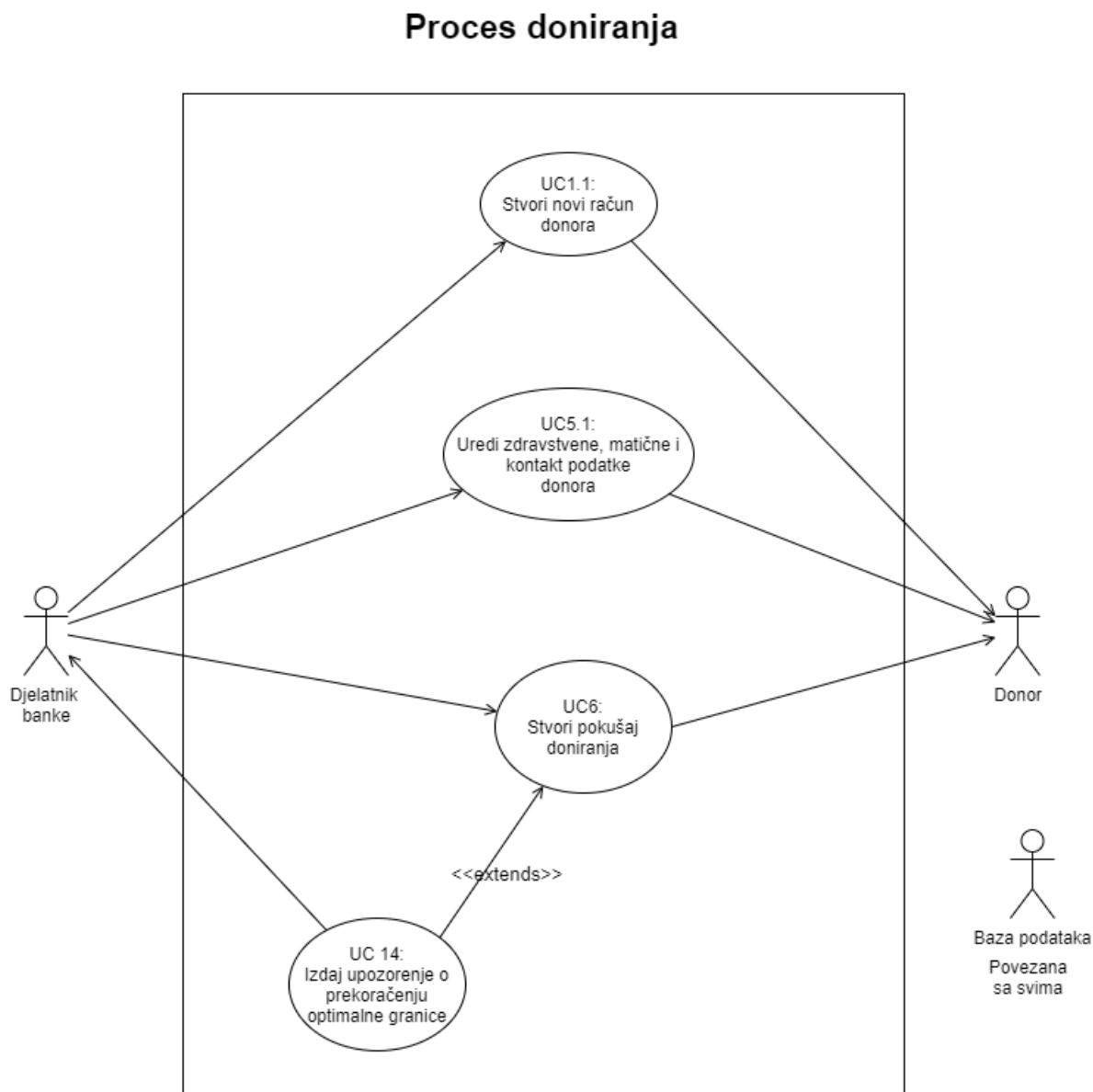
Slika 3.1: Dijagram obrazaca uporabe 1 - Administracija računa



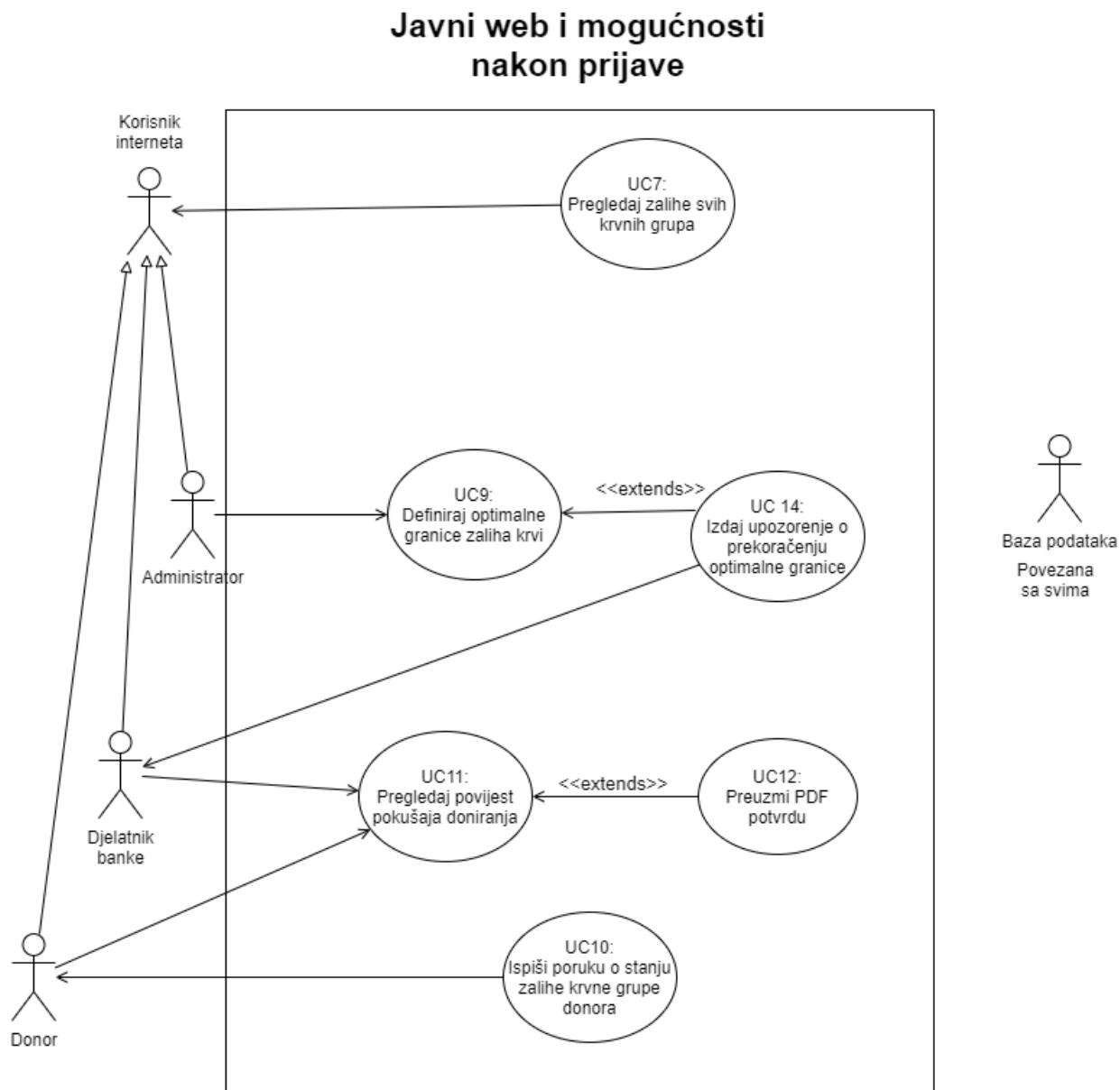
Slika 3.2: Dijagram obrazaca uporabe 2 - Proces aktivacije računa



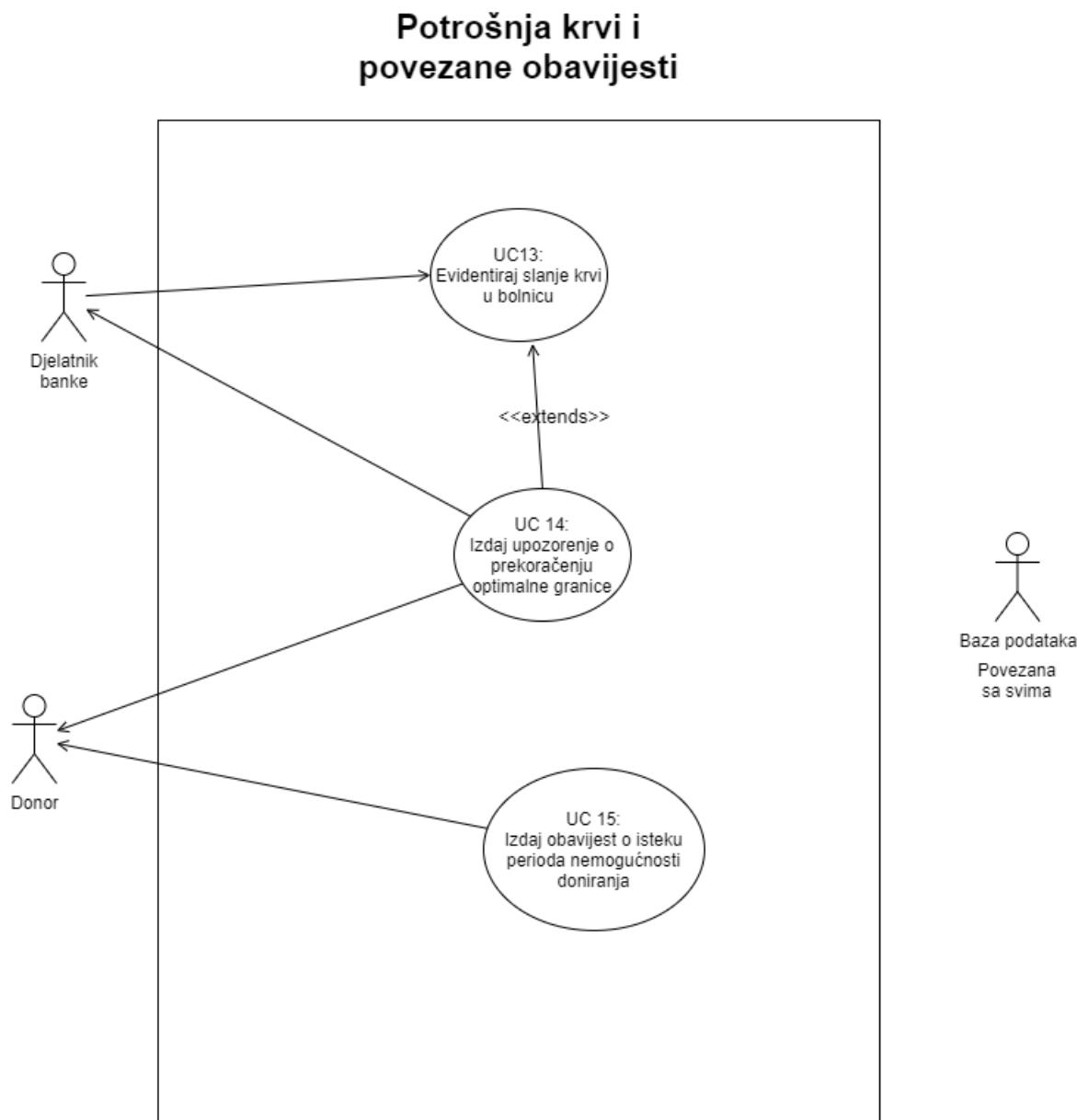
Slika 3.3: Dijagram obrazaca uporabe 3 - Uređivanje postojećih računa



Slika 3.4: Dijagram obrazaca uporabe 4 - Proces doniranja

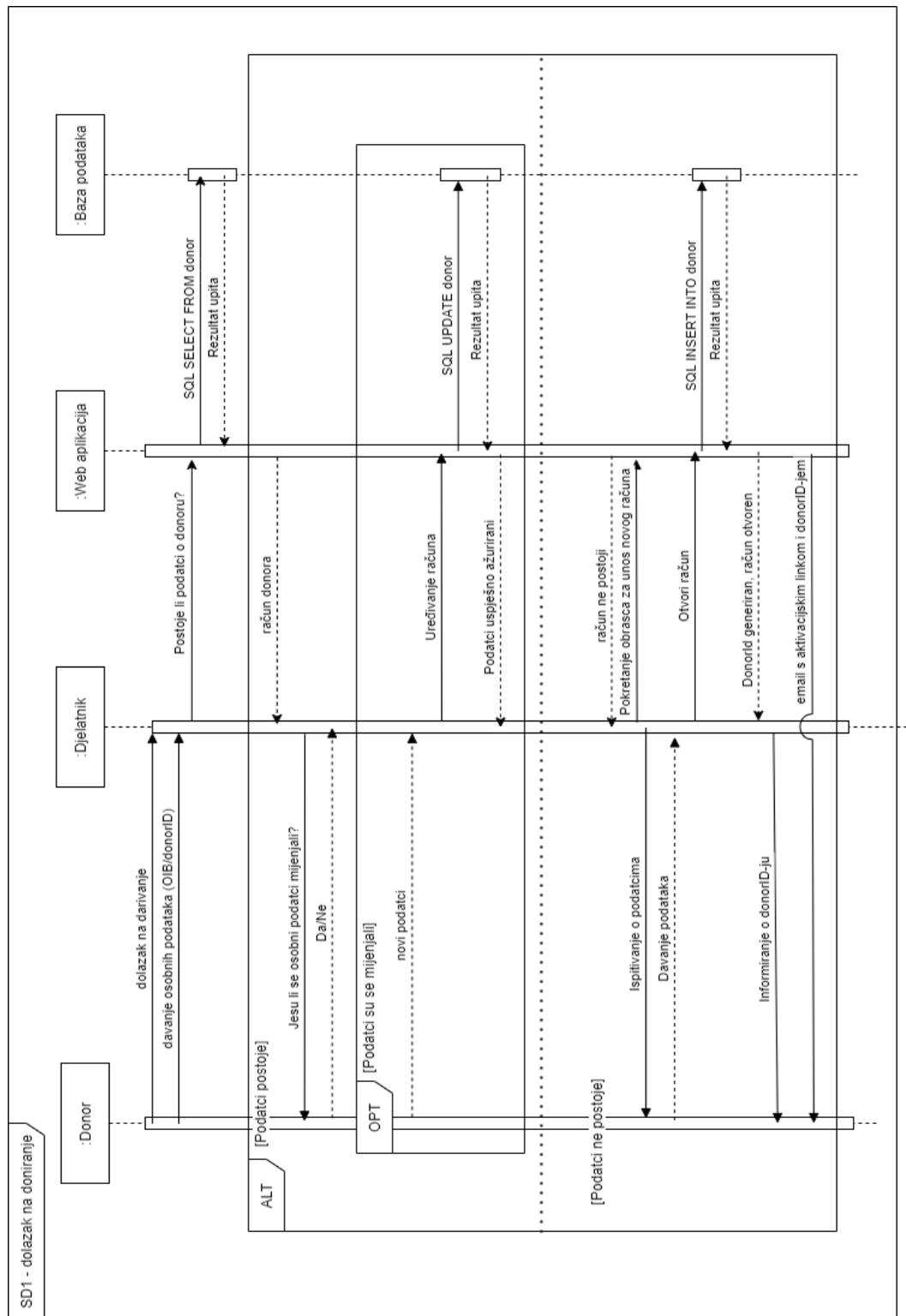


Slika 3.5: Dijagram obrazaca uporabe 5 - Javni web i mogućnosti nakon prijave



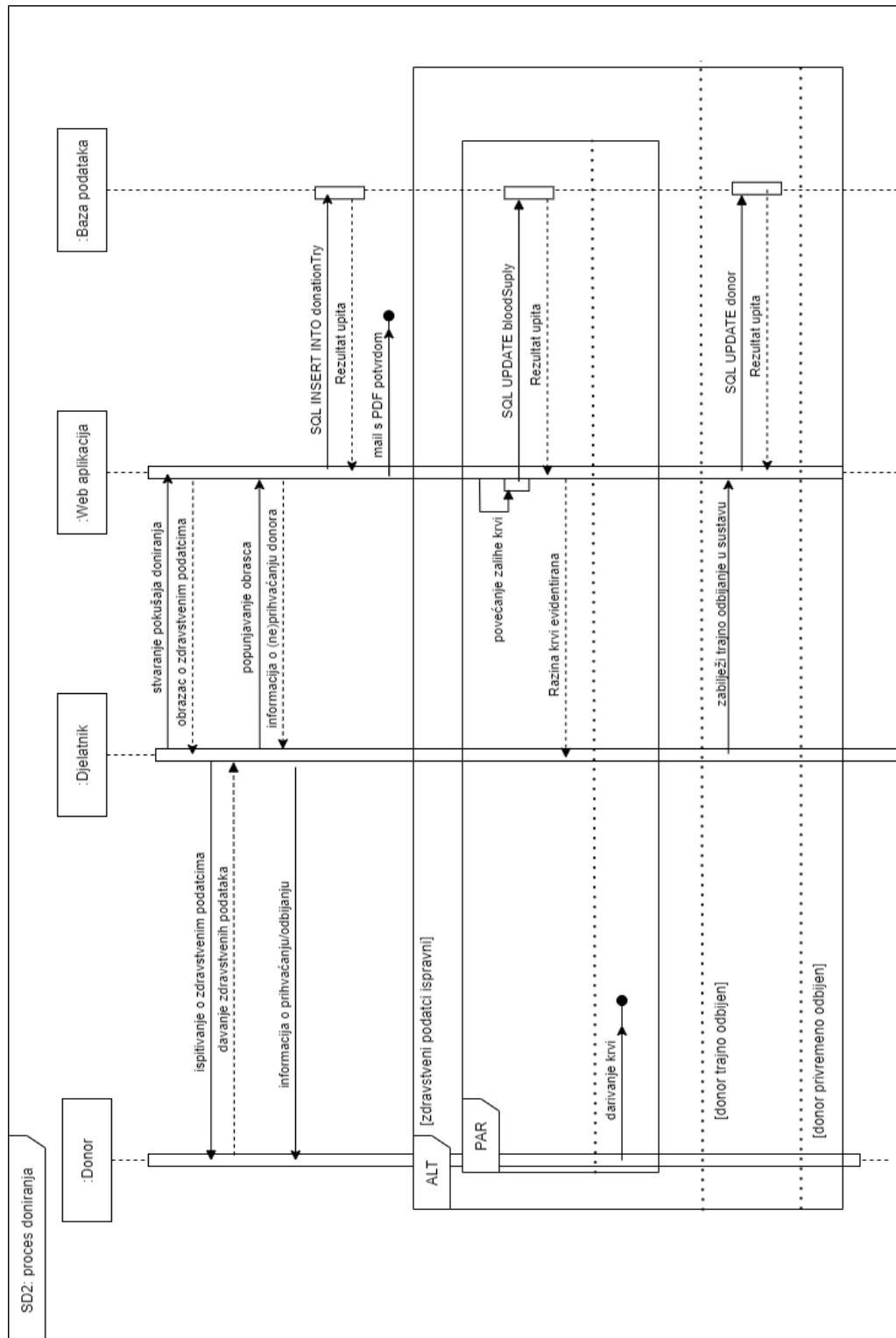
Slika 3.6: Dijagram obrazaca uporabe 6 - Potrošnja krvi i povezane obavijesti

3.1.2 Sekvencijski dijagrami



Slika 3.7: Sekvencijski dijagram 1 - Potrošnja krvi i povezane obavijesti

Sekvencijski dijagram 1: Donor dolazi na akciju darivanja krvi i djelatniku banke daje neki identifikacijski podatak (OIB ili donorID, ukoliko ga posjeduje). Djelatnik banke provjerava postoji li u sustavu donor s tim podatkom (koji je jedinstven za svaki račun). Web aplikacija djelatniku banke daje odgovor slanjem upita u bazu podataka. Ukoliko podatci postoje, djelatnik banke u komunikaciji s donorom provjerava jesu li se podatci mijenjali i ažurira ih ako jesu. U slučaju promjene, sustav ažurira podatke u bazi podataka. Ukoliko nema postojećih podataka, sustav to dojavljuje djelatniku, koji kreće u proces kreiranja novog računa. Djelatnik banke donora ispituje o osobnim podatcima, unosi ih u sustav koji te podatke unosi u bazu podataka. Generira se donorID, o kojem djelatnik banke izvještava donora. Sustav također šalje e-mail s generiranim podatcima o računu i poveznicom za aktivaciju računa.



Slika 3.8: Sekvencijski dijagram 2 - Potrošnja krvi i povezane obavijesti

Sekvencijski dijagram 2: Nakon provjere postojanja i ispravnosti podataka u sustavu, djelatnik banke za donora stvara pokušaj doniranja. U obrazac koji se otvara u web aplikaciji djelatnik banke evidentira zdravstvene podatke koje otkriva u komunikaciji s donorom (ili iz popunjenoj obrasca koji mu dostavi donor). Ukoliko neki od podataka implicira nemogućnost donora za darivanje krvi, djelatnik banke donora informira o tome. Pokušaj doniranja sprema se u bazu podataka te se donoru na mail šalje potvrda o pokušaju doniranja. U slučaju da je donor nije odbijen (zdravstveni podaci su ispravni), u sustavu se povećava zaliha krvi za jednu dozu ažuriranjem podataka u bazi, a donor odlazi donirati krv. U slučaju da je donor trajno odbijen, djelatnik banke dodatno u njegovom računu zabilježava trajno odbijanje ažuriranjem donorovih podataka u bazi. Ako je donor samo privremeno odbijen, ne poduzimaju se nikakvi dodatni koraci.

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika
- U sustavu treba postojati tri vrste korisnika - donor, djelatnik banke i administrator
- Aplikacija mora biti izvedena kao web-aplikacija
- Aplikacija mora biti prilagodljiva različitim veličinama ekrana te mobilnim uređajima
- Autentikacija korisnika radi se korisničkim imenom (donorID) i lozinkom
- Lozinke u sustavu moraju biti enkriptirane radi zaštite u slučaju neovlaštenog pristupa
- Lozinke korisnika moraju biti dovoljno sigurne radi sprječavanja neovlaštenog pristupa računu (trebaju imati bar 2 od 4 zahtjeva - mala slova, velika slova, brojevi, specijalni znakovi)
- Računi su pri stvaranju neaktivirani, a aktiviraju se aktivacijskim linkom dostavljenim u e-mailu koji se šalje pri kreiranju računa na e-adresu navedenu pri kreiranju računa
- Sustav korisnicima ne smije otežavati rad, stoga mora biti napravljen intuitivno i kao jednostavan za korištenje

- Sustav mora biti otporan na pogreške, ne smije se srušiti, nego treba dojaviti pogrešku i omogućiti izmjenu neispravno unesenih podataka

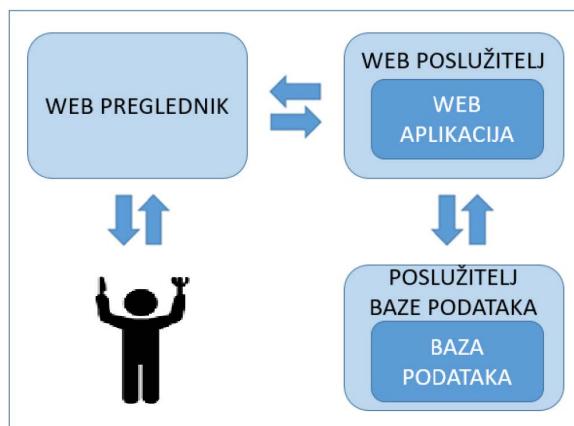
4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka

Web poslužitelj osnova je rada web aplikacije. Njegova primarna zadaća je komunikacija klijenta s aplikacijom. Komunikacija se odvija preko HTTP (engl. Hyper Text Transfer Protocol) protokola, što je protokol u prijenosu informacija na webu. Poslužitelj je onaj koji pokreće web aplikaciju te joj prosljeđuje zahtjev.

Korisnik koristi web aplikaciju za obrađivanje željenih zahtjeva. Web aplikacija obrađuje zahtjev te ovisno o zahtjevu, pristupa bazi podataka, vraća web-poslužitelju odgovor u JSON formatu, a zatim web poslužitelj te informacije prikazuje na svoj način.



Slika 4.1: Arhitektura sustava

Programski jezik koji smo odabrali za izradu naše web aplikacije je Java zajedno sa SpringBoot radnim okvirom te programski jezik Javascript s React bibliotekom (eng. library). Odabrano razvojno okruženje je IntelliJ.

Arhitektura sustava sastoji se od 3 razine:

- **Sučelje** - Najviša razina aplikacije, glavna funkcija sučelja je prevesti procese i rezultate u format koji korisnik može razumijeti
- **Logika** - Ova razina koordinira aplikaciju, obrađuje naredbe, izvršava logiku i računa. Također prenosi podatke između podatkovne razine i sučelja.
- **Podatci** - Ova razina uključuje spremanje i dohvatanje podatke iz baze podataka

Funkcionalnosti aplikacije se izvršavaju slanjem upita na *endpointe*. Oni definiraju adresu ili točku spajanja na Web poslužitelj. Tipično je reprezentiran jednostavnim HTTP URL-om (adresom, *linkom*).

Implementirani Endpointi su:

- **Prijava** - Omogućava pristup korisničkom računu, provjerava jeli dobro postavljen session cookie
- **Kreiranje donora od strane donora**
- **Kreiranje donora od strane radnika**
- **Prikaz korisničkih podataka**

4.1 Baza podataka

Sustav koristi relacijsku bazu podataka. Relacijske baze podataka pohranjuju i pružaju pristup podatcima. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Prednost relacijskih baza jest lakše definiranje odnosa između podataka. Baza podataka ove aplikacije sastoji se od ovih entiteta:

- Korisnik
 - Donor krvi
 - Djelatnik banke

- Zaliha krvi
- Pokušaj donacije

4.1.1 Opis tablica

Korisnik - Ovaj entitet sadržava sve administrativne podatke o korisniku aplikacije, njegovi osnovni podatci nalaze se u drugim, personaliziranim entitetima. Sadrži atributе: user id, user role, password, acc activated, perm deactivated, opt out. Ovaj entitet u vezi je *One-to-one* sa entitetima Donorom i Djelatnikom banke preko atributa donor id, odnosno bank worker id.

Table 4.1: Tablica *korisnik* u bazi podataka

Korisnik		
user id	BIGINT	Jedinstveni identifikator korisnika
user role	VARCHAR(20)	Je li korisnik donor, djelatnik banke ili administrator
password	VARCHAR(128)	Enkriptirana lozinka korisnika
acc activated	INT	Je li korisnik preko e-maila aktivirao svoj račun
perm deactivated	INT	Je li korisnički račun trajno deaktiviran
opt out	INT	Je li uključena opcija koja isključuje notifikacije

Donor - Ovaj entitet sadržava osobne podatke o korisniku aplikacije koji je donor. Sadrži atributе: donor id, first name, last name, oib, birth date, birth place, address, work place, private contact, work contact, email, blood type. Ovaj entitet u vezi je *One-to-many* sa entitetom Pokušaj donacije preko atributa donor id. Također je u vezi *One-to-one* sa entitetom Korisnik preko atributa donor id.

Table 4.2: Tablica *donor* u bazi podataka

Donor		
donor id	BIGINT	Jedinstveni identifikator donora krvi, odgovara identifikatoru korisnika
first name	VARCHAR(50)	Ime donora krvi
last name	VARCHAR(50)	Prezime donora krvi
oib	CHAR(11)	OIB donora krvi
birth date	DATE	Datum rođenja donora krvi
birth place	VARCHAR(100)	Mjesto rođenja donora krvi
address	VARCHAR(100)	Adresa stanovanja donora krvi
work place	VARCHAR(100)	Mjesto zaposlenja donora krvi
private contact	VARCHAR(20)	Osobni kontakt broj mobitela donora krvi
work contact	VARCHAR(20)	Poslovni kontakt broj mobitela donora krvi
email	VARCHAR(50)	e-mail adresa donora krvi
blood type	VARCHAR(3)	Krvna grupa donora krvi
perm rejected reason	VARCHAR(100)	Ako je račun donora krvi trajno deaktiviran, razlog deaktivacije

Djelatnik banke - Ovaj entitet sadržava osobne podatke djelatnika banke. Sadrži atribute: bank worker id, first name, last name, oib, birth date, birth place, address, work place, private contact, work contact, email. U vezi je *One-to-one* sa entitetom Korisnik preko atributa bank worker id.

Table 4.3: Tablica *djelatnik banke* u bazi podataka

Djelatnik banke		
bank worker id	BIGINT	Jedinstveni identifikator djelatnika banke, odgovara identifikatoru korisnika
first name	VARCHAR(50)	Ime djelatnika banke

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Djelatnik banke		
last name	VARCHAR(50)	Prezime djelatnika banke
oib	CHAR(11)	OIB djelatnika banke
birth date	DATE	Datum rođenja djelatnika banke
birth place	VARCHAR(100)	Mjesto rođenja djelatnika banke
address	VARCHAR(100)	Adresa stanovanja djelatnika banke
work place	VARCHAR(100)	Mjesto zaposlenja djelatnika banke
private contact	VARCHAR(20)	Osobni kontakt broj mobitela djelatnika banke
work contact	VARCHAR(20)	Poslovni kontakt broj mobitela djelatnika banke
email	VARCHAR(50)	e-mail adresa djelatnika banke

Zaliha krvi - Ovaj entitet sadržava podatke o količini krvi u banci. Sadrži attribute: number of units, max units, min units.

Table 4.4: Tablica *zaliha krvi* u bazi podataka

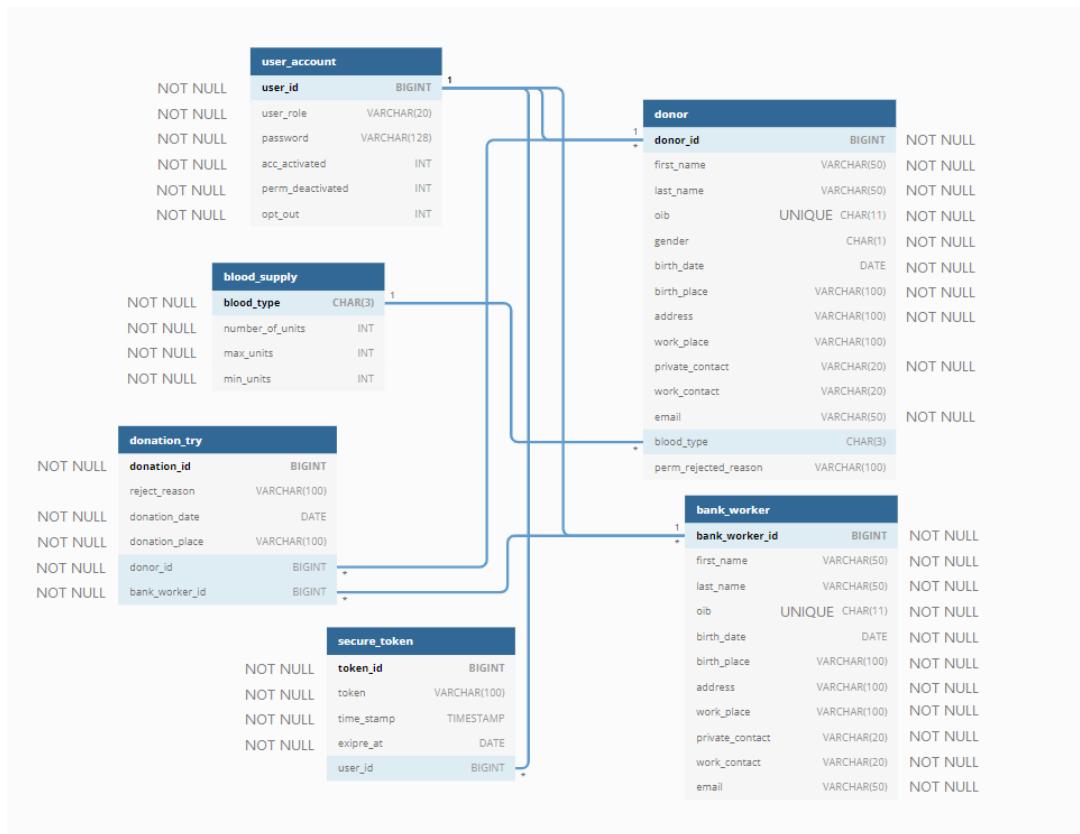
Zaliha krvi		
blood type	CHAR(3)	Krvna grupa jedinice krvi
number of units	INT	Trenutni broj jedinica krvi
max units	INT	Maksimalni broj jedinica krvi
min units	INT	Minimalni broj jedinica krvi

Pokušaj donacije - Ovaj entitet sadržava podatke o pojedinoj donaciji krvi. Sadrži atributе: donation id, rejected reason, blood type, donor id, bank worker id. U vezi je *Many-to-one* sa entitetom Donor preko atributa donor id.

Table 4.5: Tablica *pokušaj donacije* u bazi podataka

Pokušaj donacije		
rejected reason	VARCHAR(100)	Razlog neuspješnosti pokušaja darivanja krvi
blood type	CHAR(3)	Kvrna grupa donora krvi koji obavlja ovu donaciju
donor id	BIGINT	Jedinstveni identifikator donora krvi koji obavlja ovu donaciju krvi
bank worker id	BIGINT	Jedinstveni identifikator djelatnika banke koji nadgleda ovu donaciju krvi

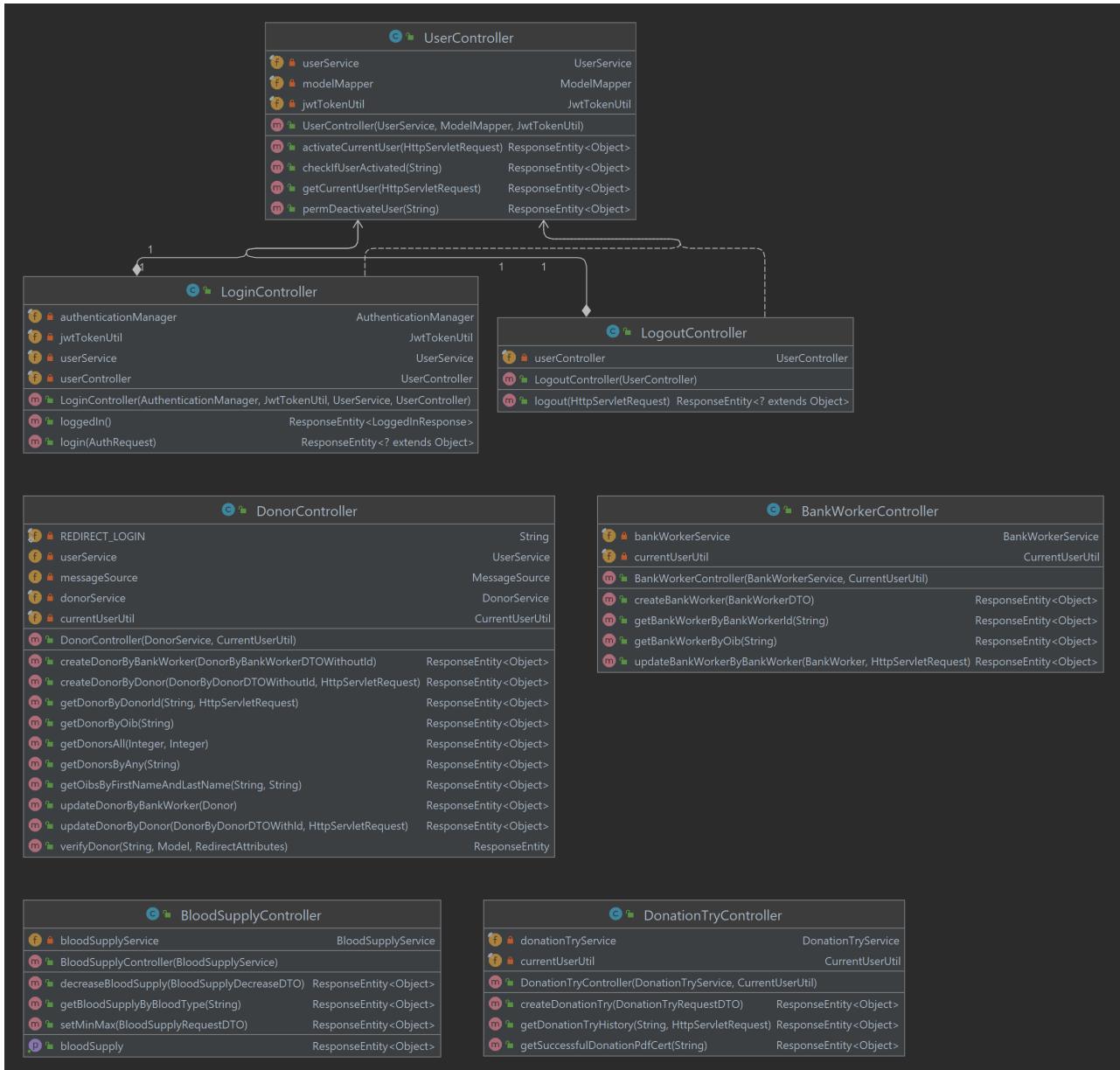
4.1.2 Dijagram baze podataka



Slika 4.2: Dijagram baze podataka

4.2 Dijagram razreda

Na slikama u nastavku prikazani su razredi koji pripadaju arhitekturi poslužiteljske aplikacije. Radi preglednosti, dijagram razreda razlomljen je na više njih, grupiranih prema sličnim razinama apstrakcije i srodnim funkcionalnostima.



Slika 4.3: Dijagram controller klasa

Controller klase vraćaju entitet odgovora (*response entity*) kao odgovor na zahjeve na krajnje točke. Entitet odgovora je objekt koji sadrži HTTP status i JSON (*JavaScript Object Notation*) podatke. Obično se radi o dohvaćanju podataka koje

korisnik aplikacije zahtjeva. U *Controller* klasama je također određeno kojim krajnjim točkama korisnici imaju pristup (ovisno o ulozi korisnika) pomoću anotacija. Metode u Controller klasama obično pozivaju metode istog imena u odgovarajućim Service klasama te ih se može shvatiti kao klase koje svoju funkcionalnost vrše koristeći poslovnu logiku iz Service klase. Ako neki Controller sadrži objekt tipa Service (većina ih sadrži), tada on koristi neke od metoda iz odgovarajuće Service klase. Bitno je napomenuti da se koristi @Controller notacija.

LoginController zadužen je za ostvarivanje funkcionalnosti prijave. Login prima HTTP GET zahtjev. Koristi se osnovna autentikacija (*basic authentication*), što znači da su u autorizacijskom zaglavlju (*Authorization header*) zapisani userId i lozinka. Provjerava se postoji li u sustavu korisnik s navedenim identifikacijskim brojem, te ako postoji provjeriti je li lozinka u zahtjevu jednaka onoj zapisanoj u bazi. Lozinke se spremaju i provjeravaju u kodiranom obliku pomoću *BCryptEncoder*. Kod vraćanja odgovora postavlja se kolačić sa sjednicom (*session cookie*), a ukoliko su navedeni podaci ispravni, zapisuju se podatci o prijavljenom korisniku (poput imena i uloga). Ukoliko podaci nisu ispravni vraća se kod o pogrešci 401. Uz naredne zahtjeve korisnika prilaže se kolačić sa sjednicom čime poslužiteljska strana može prepoznati korisnikovu sjednicu i potvrditi da je korisnik prijavljen.

UserController služi za:

- Aktivaciju korisnika i provjeru stanja aktiviranosti korisnika
- Dohvaćanje korisnika
- Trajnu deaktivaciju korisničkog računa.

DonorController služi za:

- Dohvaćanje i kreiranje donora
- Dohvaćanje OIB-a donora
- Upravljanje podatcima donora

DonationTryController služi za:

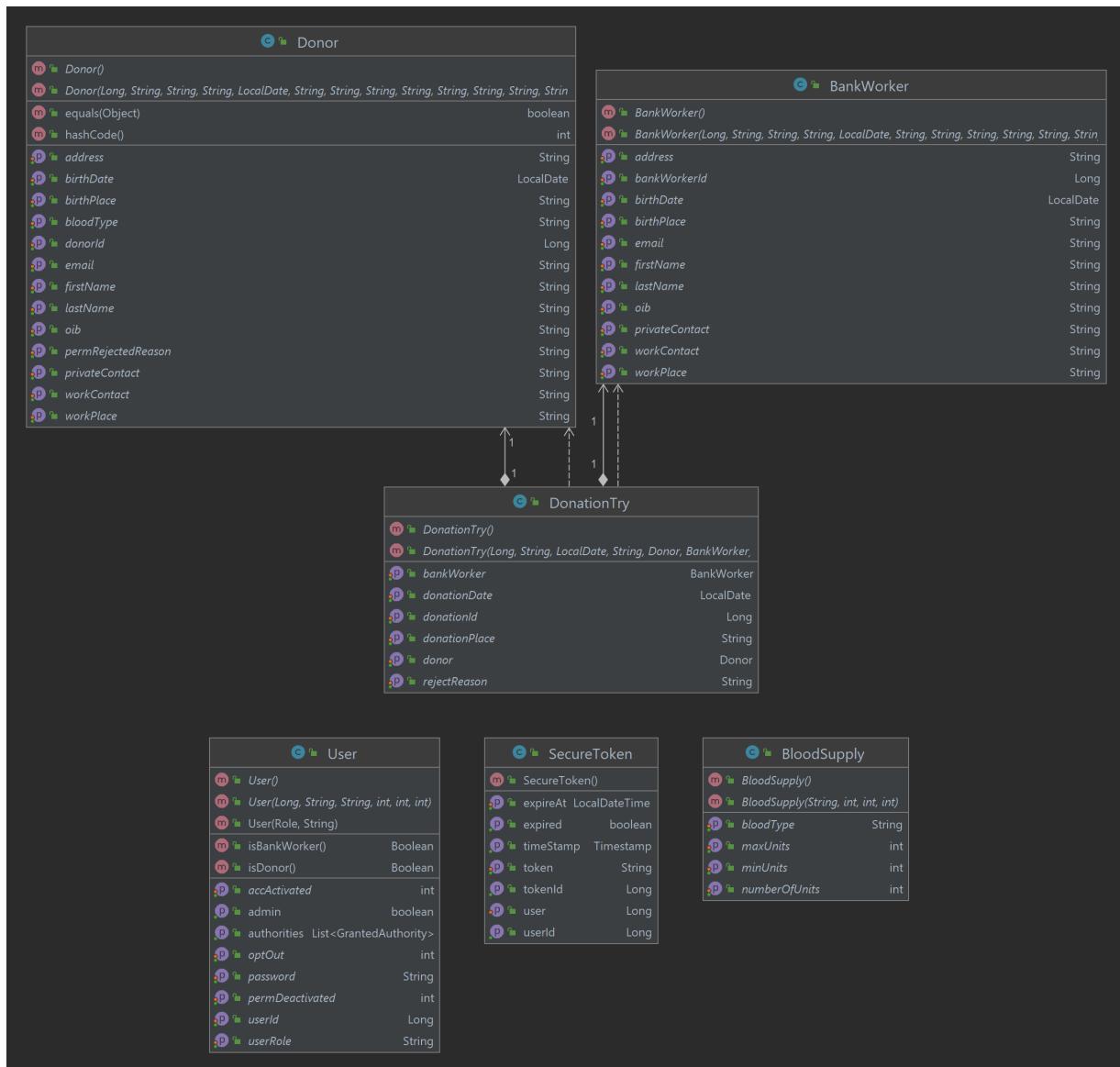
- Kreiranje pokušaja donacije
- Dohvaćanje povijesti pokušaja donacije
- Zahtjevanje pdf potvrde donacije

BloodSupplyController služi za:

- Smanjivanje razine krvi
- Dohvaćanje pojedinačnih ili svih zaliha krvi (ovisno o krvnoj grupi)
- Postavljanje minimalne i maksimalne prihvatljive zalihe

BankWorkerController služi za:

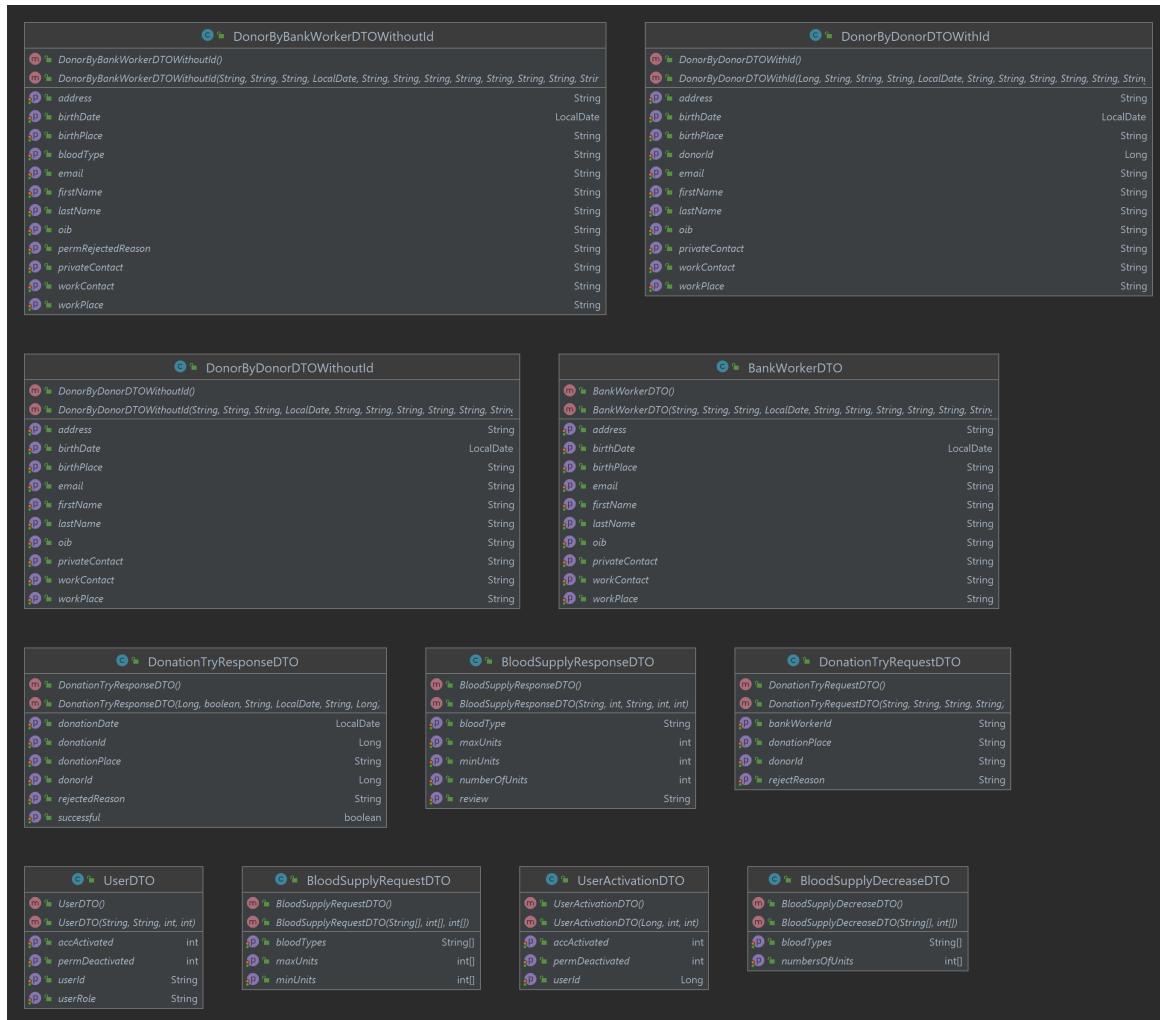
- Kreiranje, dohvaćanje i upravljanje djelatnikom banke



Slika 4.4: Dijagram model klasa

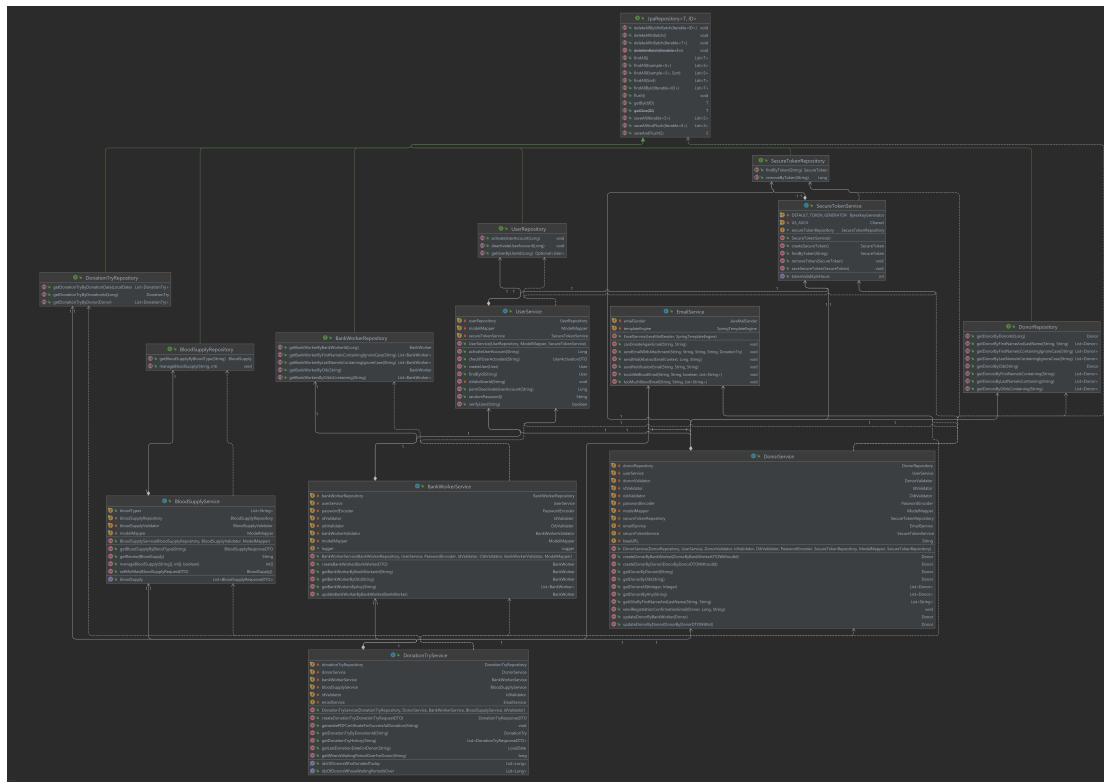
Model klase služe za stvaranje objekata ekvivalentnim onima u bazi podataka. Ove klase ne sadrže metode koje obavljaju poslovnu logiku (samo gettere, setttere,

konstruktore itd.). Valja navesti jednu iznimku, SecureToken model klasu, koja nema ekvivalent u bazi podataka. Taj objekt se koristi u svrhu generiranja jedinstvene poveznice. Više informacija nalazi se u opisu SecureTokenService klase. *User*, *Donor* i *BankWorker* klase implementiraju sučelje *Serializable*.



Slika 4.5: Dijagram DTO klasa

DTO (Data transfer object) klase služe kako bi se mogle vratiti na krajnje točke, i time izbjegći nepotrebno (ili nesigurno) slanje informacija koje nisu potrebne.



Slika 4.6: Dijagram Repository sučelja i Service klasa

Na sljedećoj poveznici nalazi se dijagram klase za Repository i Service klase koji se može uvećati: [Dijagram Repository sučelja i Service klasa](#)

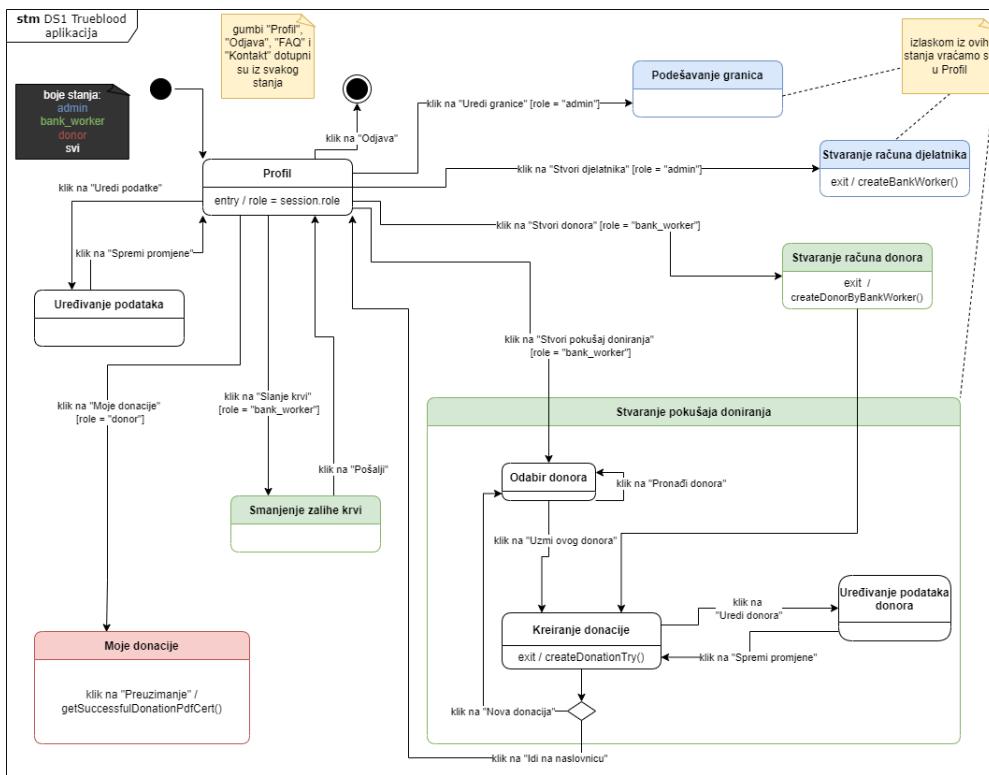
Repository - Dohvaća i sprema podatke iz baze podataka, koristeći sučelje *JpaRepository*, koje sva Repository sučelja implementiraju. *JpaRepository* je sučelje s metodama za pristup bazi podataka iz Springa, a metode se kreiraju same na temelju imenovanja metoda.

Service klase obavljaju poslovnu logiku aplikacije. Mnoge od metoda u Service klasama pozivaju su iz istoimenih metoda u odgovarajućim Controller klasama. Ako metode trebaju upravljati podatcima, koriste metode iz Repository sučelja. Service klase uglavnom obavljaju logiku iza funkcionalnosti koje su navedene za Controller klase, ali valja navesti dvije iznimke:

EmailService koji služi za slanje e-mailova (aktivacijskih i obavijesti) pomoću *JavaMailSender* sučelja i **SecureTokenService** koji služi isključivo za generiranje jedinstvene poveznice koja se šalje u aktivacijskoj e-poruci. SecureTokenService u sebi sadrži objekt tipa *BytesKeyGenerator* koji se koristi u spomenutom generiranju.

[Dijagram svih klasa aplikacije](#)-Na ovoj poveznici nalazi se dijagram svih klasa aplikacije.

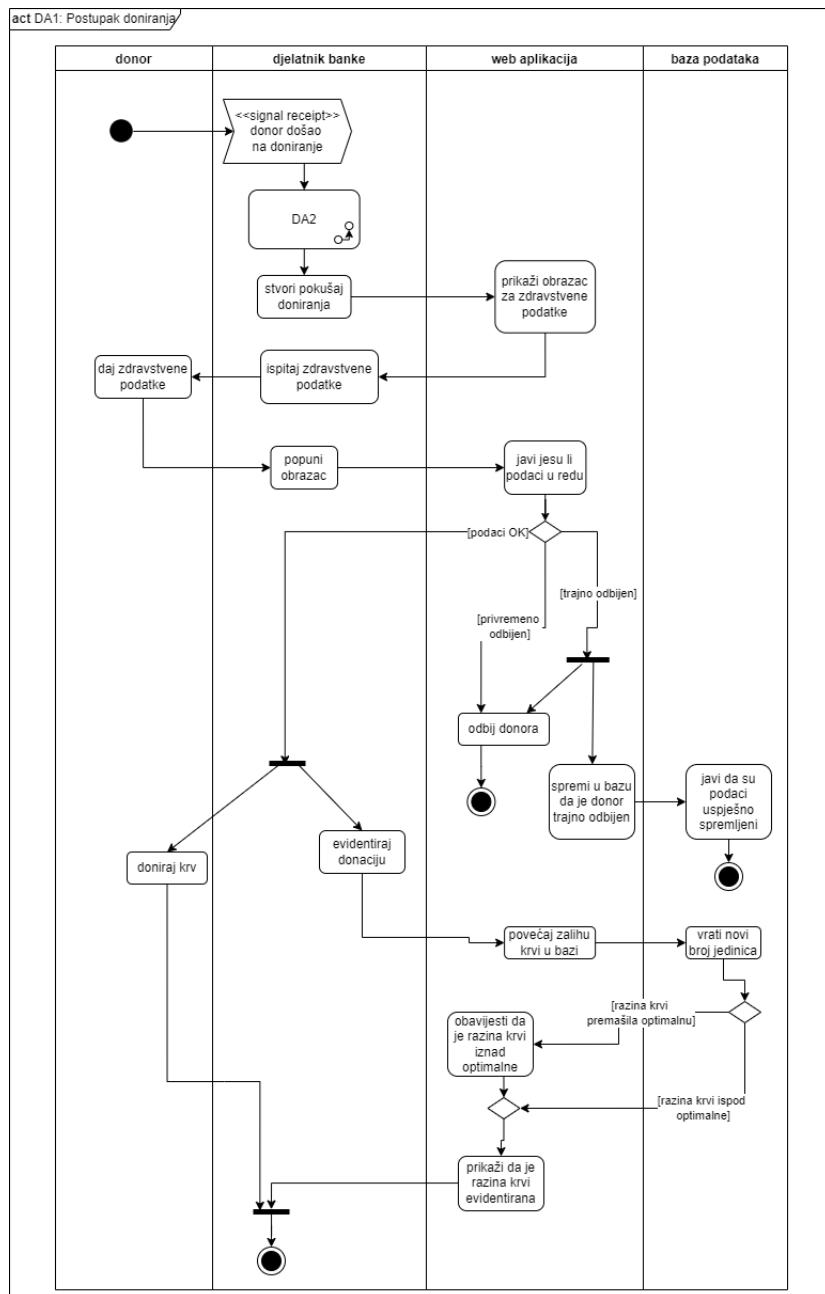
4.3 Dijagram stanja



Slika 4.7: DS1 Trueblood web aplikacija

Dijagram stanja prikazuje stanja u koje je moguće doći različitim akcijama korisnika. Stanja na dijagramu na slici 4.7 predstavljaju različite komponente Trueblood web aplikacije. Plavom bojom prikazane su komponente kojima može pristupiti administrator, zelenom djelatnik banke, crvenom donor, a bijelom svi. Sve funkcije navedene u dijagramu nalaze se u odgovarajućim *controller* klasama. U početno stanje korisnik dolazi prijavom u sustav. Iz svakog stanja moguće se je odjaviti, doći u stanje profil te vidjeti često postavljana pitanja (FAQ) i kontakt, pritiskom na odgovarajuće gume. Iz stanja profil, korisnik može doći u razna druga stanja ovisno o svojoj ulozi. Svaki korisnik može urediti svoje podatke. Administrator ima mogućnost podešavanja optimalnih granica razina krvi u baci te može stvoriti račun djelatnika banke. Djelatnik banke ima mogućnost stvaranja računa donora, smanjivanja zaliha krvi te stvaranja pokušaja doniranja. Prilikom postupka doniranja, djelatnik banke može odabrati postojećeg donora ili stvoriti novog. Prema podacima donora djelatnik banke kreira donaciju koja se evidentira klikom na gumb "Doniraj".

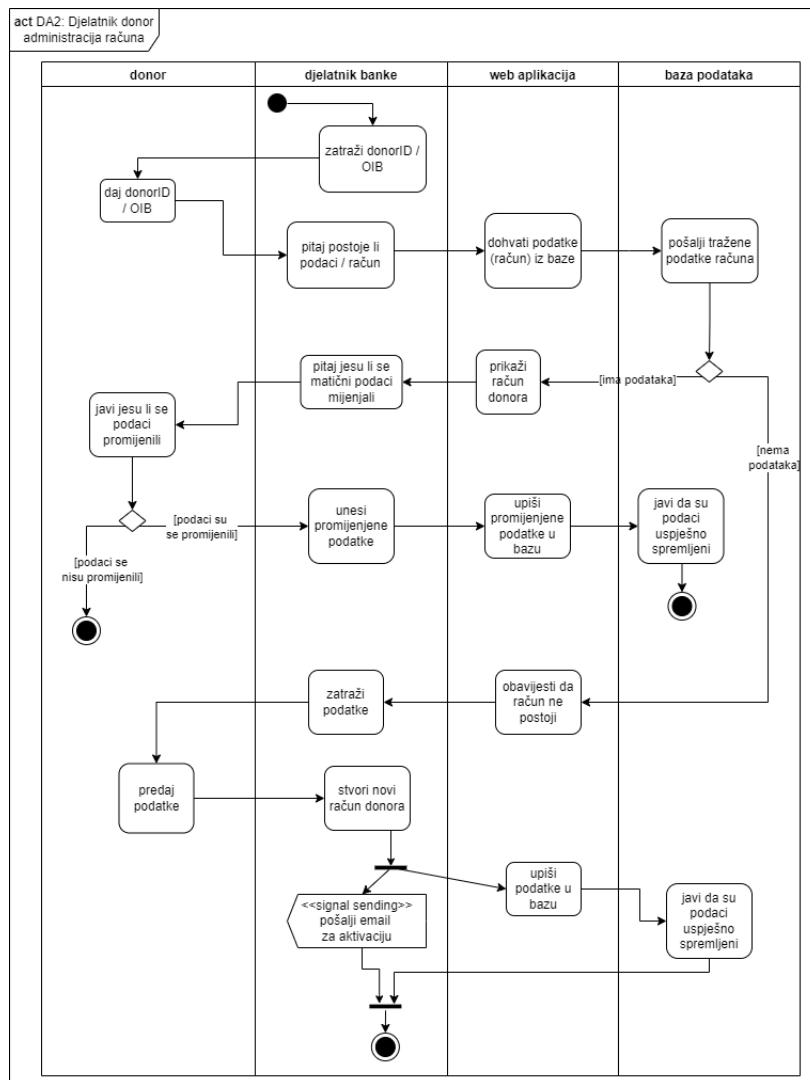
4.4 Dijagram aktivnosti



Slika 4.8: DA1 Postupak doniranja

Dijagram aktivnosti prikazan na slici 4.8 modelira tijek aktivnosti prilikom postupka doniranja. Po dolasku donora na doniranje, djelatnik banke vrši administraciju donorovog računa, opisanu na *DA2 Djelatnik donor administracija računa* (slika 4.9). Djelatnik banke zatim stvara novi pokušaj doniranja koji sadrži matične podatke donora te od donora saznaće njegove trenutne zdravstvene podatke. Ako

zdravstveni podaci nisu u redu, donor se odbija te ne može donirati krv. Ako su zdravstveni podaci donora u redu, donor odlazi donirati krv, dok djelatnik banke u sustavu evidentira njegovu donaciju te web aplikacija povećava razinu krvi u bazi. Ako razina krvi prijeđe gornju optimalnu granicu, sustav obavještava djelatnike banke porukom na profilu i e-porukom.



Slika 4.9: DA2 Djelatnik donor administracija računa

Dijagram aktivnosti prikazan na slici 4.9 prikazuje upravljanje djelatnika banke donorovim računom. Djelatnik banke koristi donorov OIB kako bi utvrdio postoji li već račun koji odgovara tom donoru. Ako račun ne postoji, djelatnik banke kreira novi račun donora te sustav donoru šalje e-poruku za aktivaciju računa. Ukoliko račun postoji, a donorovi podaci su se promjenili, tada djelatnik banke ispravlja donorove podatke.

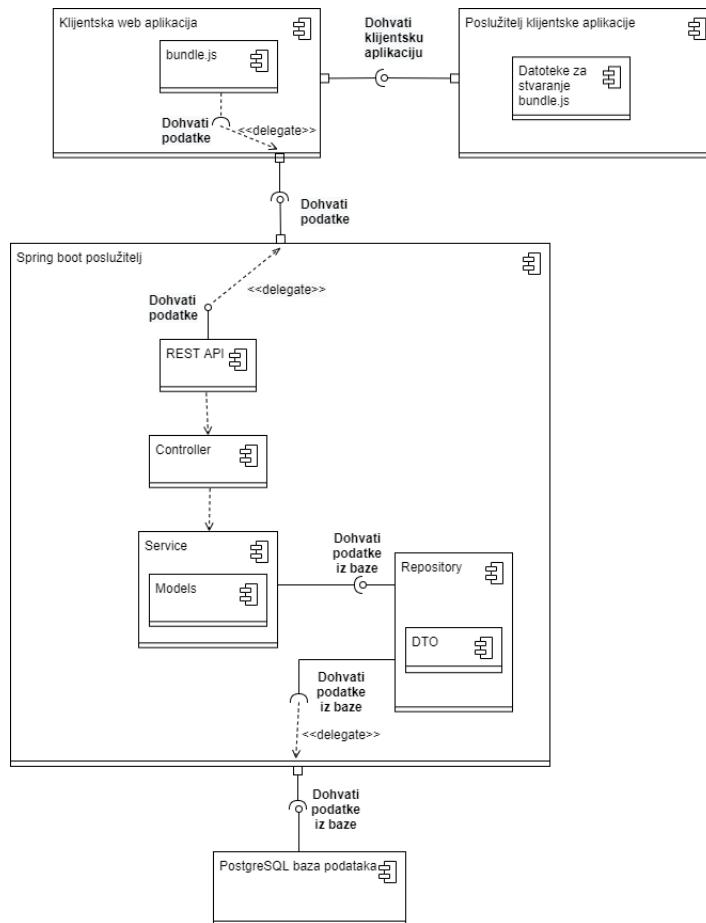
4.5 Dijagram komponenti

Dijagram komponenti na slici 4.10 opisuje internu strukturu sustava, kako su komponente međusobno povezane te kako im se pristupa iz okoline.

Poslužitelj klijentske aplikacije poslužuje *bundle.js* datoteku koja na klijentskom računalu pokreće klijentsku web aplikaciju, a ona generira sve potrebne HTML i CSS datoteke te JavaScript kod.

Poslužiteljska aplikacija na krajnijim točkama (eng. *endpoint*) API-ja (aplikacijsko-programskog sučelja) dočekuje zahtjeve klijentske aplikacije, obrađuje ih i vraća tražene podatke (ako oni postoje i ako klijentska aplikacija ima dopuštenje tražiti ih). API radi na REST principu, stoga ga nazivamo i REST API. REST API dio je Spring Boot kontrolera (eng. *Controller*). Jednom kada neki kontroler primi zahtjev preko API-ja, on će odgovarajućem servisu proslijediti zahtjev, a od njega dobiti odgovor koji treba vratiti aplikaciji. Servisi (eng. *service*) obrađuju zahtjeve koje im šalju kontroleri i dohvaćaju podatke iz repozitorija (eng. *repository*), obrađuju te podatke te stvaraju odgovor koji vraćaju kontroleru. Repozitoriji podatke dobavljaju iz baze podataka pomoću SQL upita. Kako bi dobivene podatke mogli vratiti servisima koji će te podatke koristiti u obradi, koriste se modeli (eng. *models*), koji sadrže varijable koje odgovaraju određenim poljima iz baze podataka. Modeli su u pravilu jednostavnii i sadrže varijable koje odgovaraju stupcima jedne tablice iz baze podataka. Kako nisu uvijek potrebni svi stupci neke tablice, podatci u prijenosu modeliraju se objektima za prijenos podataka (eng. *DTO*), a moguće je da ti objekti ne sadrže sve varijable, već varijable koje odgovaraju samo nekim stupcima tablice.

PostgreSQL baza podataka na SQL upite vraća podatke iz baze podataka.



Slika 4.10: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Kao platforma za zajednički razvoj projekta, korišten je [GitLab](#). Na njoj je dostupan udaljeni repozitorij projekta. Korištenjem GitLaba i [Gita](#) - sustava za upravljanjem izvornim kodom projekta, članovi razvojnog tima mogu istovremeno raditi na projektu, imajući dostupne promjene čim ih pojedini član tima objavi. Za komunikaciju, razvojni tim koristio je platformu [Slack](#). To je platforma koja omogućava efikasnu komunikaciju članova kroz više specijaliziranih kanala komunikacije.

[IntelliJ IDEA](#) odabran je kao razvojno okruženje za backend dio aplikacije. JetBrains je razvio to integrirano razvojno okruženje (IDE) koje nudi inteligentnu podršku programiranju u programskom jeziku [Java](#).

Kao sredstvo za olakšavanje izrade backend dijela aplikacije korišten je [Java Spring Framework](#) te [Spring Boot](#). Spring Framework pojednostavljuje izradu aplikacija koje se izvode na Java virtualnom stroju (engl. *Java Virtual Machine (JVM)*) kroz 4 strategije:

- labavo povezivanje objekata ostvareno kroz umetanje ovisnosti (eng. dependency injection) i korištenjem sučelja,
- lagan i jednostavan razvoj koristeći obične Java objekte,
- deklarativno programiranje kroz aspekte i uobičajene konvencije,
- eliminiranje ponavljajućeg koda koristeći aspekte i predloške.

Spring Boot je aplikacijski okvir koji dodatno proširuje Spring okvir. Njegove glavne značajke su automatska konfiguracija i samostalno odlučivanje o početnim ovisnostima, ovisno o potrebama projekta. Kao lokalni spremnik za bazu podataka korišten je [Docker](#). Docker je skup platformi za isporuku softvera u paketima (spremnicima). Popularan je jer olakšava postavljanje lokalnog razvojnog okruženja te njegovo korištenje spremnika omogućava stvaranje više izoliranih okruženja.

[VSC](#) odabran je kao razvojno okruženje za frontend dio aplikacije. Izvorni kod pisan je u jeziku [JavaScript](#) uz upotrebu [Reacta](#). React je JavaScript biblioteka koja služi za izgradnju korisničkih sučelja jednostraničnih aplikacija (engl. *Single-page application*). Baziran je na komponentama koje se mogu ponovno koristiti.

Kako bi aplikacija bila javno dostupna, korišten je [Heroku](#). To je platforma za puštanje u pogon te daljnje održavanje aplikacija.

Za izradu dokumentacije korišten je [Overleaf](#), online uređivač dokumenata pisanih u LaTeX markup jeziku. Za izradu dijagrama korišten je [draw.io](#) - online alat za izradu UML, ER te raznih drugih dijagrama.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje je provedeno koristeći Junit biblioteku za Maven. Ispitivani su redovni slučajevi, ali i izazivanje pogrešaka. Testirane su temeljna funkcionalnost aplikacije i funkcije koje upravljaju korisnicima. Slijede priložene slike testova:

```
//tests UserService.findNotDeactivatedUserById()
@Test
void testFindNotDeactivatedUserById(){
    User user = new User(Role.DONOR, password: "59263487");
    Mockito.when(userRepository.getNotDeactivatedUserByUserId(Mockito.any())).thenReturn(java.util.Optional.of(user));
    ReflectionTestUtils.setField(userService, name: "userRepository", userRepository);
    User retUser = userService.findNotDeactivatedUserById( userIdString: "12355");

    assertEquals(retUser, user);
}

//tests UserService.findNotDeactivatedUserById()
@Test
void testFindNotDeactivatedUserByIdForNonExistingUser(){

    Mockito.when(userRepository.getNotDeactivatedUserByUserId(Mockito.any())).thenReturn(Optional.empty());
    ReflectionTestUtils.setField(userService, name: "userRepository", userRepository);

    UsernameNotFoundException ex = assertThrows(
        UsernameNotFoundException.class,
        () -> userService.findNotDeactivatedUserById( userIdString: "12355"),
        message: "Expected findById to throw, but it didnt");

    assertEquals( expected: "Ne postoji korisnik s tim id-em.", ex.getMessage());
}

@Test
void testFindNotDeactivatedUserByIdForNonNumericId(){

    Mockito.when(userRepository.getNotDeactivatedUserByUserId(Mockito.any())).thenReturn(Optional.empty());
    ReflectionTestUtils.setField(userService, name: "userRepository", userRepository);

    WrongUserException ex = assertThrows(
        WrongUserException.class,
        () -> userService.findNotDeactivatedUserById( userIdString: "abcd"),
        message: "Expected findNotDeactivatedUserById to throw, but it didnt");

    assertEquals( expected: "Id korisnik nije numerički.", ex.getMessage());
}
```

Slika 5.1: Testovi, prvi dio

```
//tests UserService.randomPassword()
@Test
void testRandomPassword(){
    String randomPass = userService.randomPassword();
    assertEquals(randomPass.length(), actual: 8);
}
```

Slika 5.2: Testovi, drugi dio

```
@Test
void testUpdateDonorByBankWorker(){
    Donor donor = new Donor( donorId: 1234L, firstName: "Josip", lastName: "Josipovic", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);
    Donor newDonor = new Donor( donorId: 1234L, firstName: "Josip", lastName: "Horvat", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);

    Mockito.when(donorRepository.getNotDeactivatedDonorByDonorId(Mockito.any())).thenReturn(donor);
    Mockito.when(donorRepository.save(Mockito.any())).thenReturn(newDonor);
    Mockito.when(modelMapper.map(Mockito.any(), Mockito.any())).thenReturn(newDonor);

    Donor retDonor = donorService.updateDonorByBankWorker(newDonor);

    assertEquals(retDonor, newDonor);
}

@Test
void testUpdateDonorByBankWorkerNoId(){
    Donor donor = new Donor( donorId: null, firstName: "Josip", lastName: "Josipovic", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);

    WrongDonorException ex = assertThrows(
        WrongDonorException.class,
        () -> donorService.updateDonorByBankWorker(donor),
        message: "Expected findNotDeactivatedUserById to throw, but it didn't");

    assertEquals( expected: "Id darivatelja krvi nije definiran.", ex.getMessage());
}
```

Slika 5.3: Testovi, treći dio

```
@Test
void testUpdateDonorByBankWorkerNoUser(){
    Donor donor = new Donor( donord: 1234L, firstName: "Josip", lastName: "Josipovic", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);
    Mockito.when(donorRepository.getNotDeactivatedDonorByDonorId(Mockito.any())).thenReturn(null);

    WrongDonorException ex = assertThrows(
        WrongDonorException.class,
        () -> donorService.updateDonorByBankWorker(donor),
        message: "Expected findNotDeactivatedUserById to throw, but it didnt");

    assertEquals( expected: "Ne postoji darivatelj krvi s tim id-em.", ex.getMessage());
}

@Test
void testUpdateDonorByDonor(){
    Donor donor = new Donor( donord: 1234L, firstName: "Josip", lastName: "Josipovic", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);
    DonorByDonorDTOWithId newDonor = new DonorByDonorDTOWithId( donord: 1234L, firstName: "Josip", lastName: "Horvat",
        oib: "12345678", gender: "M", LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb",
        address: "Ilica 2", workPlace: "Fer", privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io");
    Donor referenceDonor = new Donor( donord: 1234L, firstName: "Josip", lastName: "Horvat", oib: "12345678", gender: "M",
        LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb", address: "Ilica 2", workPlace: "Fer",
        privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io", bloodType: "A-", permRejectedReason: null);

    Mockito.when(donorRepository.getNotDeactivatedDonorByDonorId(Mockito.any())).thenReturn(donor);
    Mockito.when(donorRepository.save(Mockito.any())).thenReturn(referenceDonor);
    Mockito.when(modelMapper.map(Mockito.any(), Mockito.any())).thenReturn(referenceDonor);

    Donor retDonor = donorService.updateDonorByDonor(newDonor);

    assertEquals(retDonor, referenceDonor);
}
}
```

Slika 5.4: Testovi, četvrti dio

```

@Test
void testUpdateDonorByDonorNoUser(){
    DonorByDonorDTOWithId newDonor = new DonorByDonorDTOWithId( donorId: 1234L, firstName: "Josip", lastName: "Horvat",
        oib: "12345678", gender: "M", LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb",
        address: "Ilica 2", workPlace: "Fer", privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io");
    Mockito.when(donorRepository.getNotDeactivatedDonorByDonorId(Mockito.any())).thenReturn(null);

    WrongDonorException ex = assertThrows(
        WrongDonorException.class,
        () -> donorService.updateDonorByDonor(newDonor),
        message: "Expected findNotDeactivatedUserById to throw, but it didnt");

    assertEquals( expected: "Ne postoji darivatelj krvi s tim ID-jem.", ex.getMessage());
}

@Test
void testUpdateDonorByDonorNoId(){
    DonorByDonorDTOWithId newDonor = new DonorByDonorDTOWithId( donorId: null, firstName: "Josip", lastName: "Horvat",
        oib: "12345678", gender: "M", LocalDate.of( year: 2000, month: 9, dayOfMonth: 23), birthPlace: "Zagreb",
        address: "Ilica 2", workPlace: "Fer", privateContact: "0911234123", workContact: "0991234123", email: "josip@email.io");

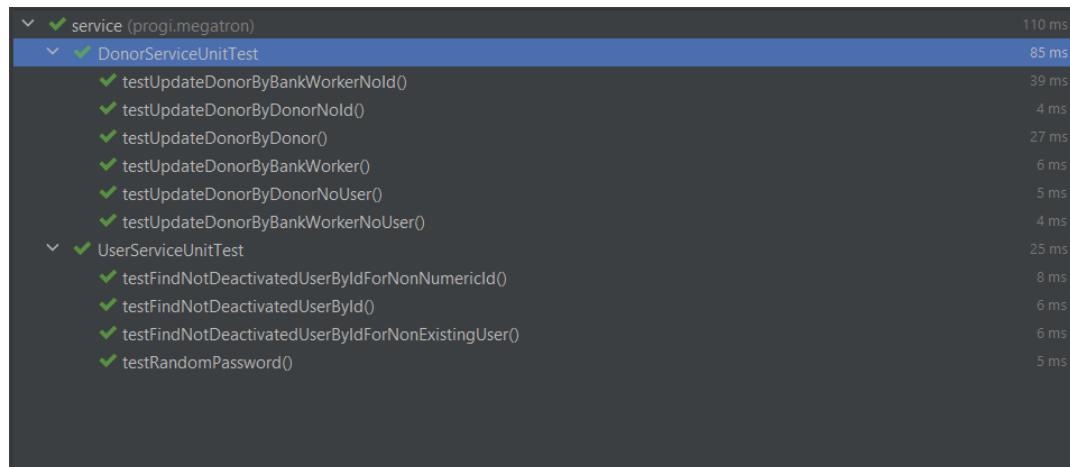
    WrongDonorException ex = assertThrows(
        WrongDonorException.class,
        () -> donorService.updateDonorByDonor(newDonor),
        message: "Expected findNotDeactivatedUserById to throw, but it didnt");

    assertEquals( expected: "Id darivatelja krvi nije definiran.", ex.getMessage());
}

```

Slika 5.5: Testovi, peti dio

Svi testovi vraćaju prolaz te sve testirane funkcionalnosti rade. Slijedi dokaz:



Slika 5.6: Rezultati testiranja

5.2.2 Ispitivanje sustava

Koristeći Selenium IDE plugin za Chrome, provedeno je testiranje sustava. Korištena su četiri testa (slijede njihovi nazivi, opisi te rezultati):

- 1. *Prijava donora - odjava:* Radi se o jednostavnom testu osnovne funkcionalnosti svakog korisnika.



Slika 5.7: Rezultati prvog testa - profilna stranica prijavljenog korisnika

- 2. *Registracija (greška: Postojeći OIB):* Testira javljanje greške pri pravljenju računa sa već zauzetim OIB-om.

08/01/1955 □ Zagreb

nigdje

Kontakt podaci

jakov.matosic@gmail.com

0911540907

Fer Kontakt (poslovni)

Zdravstveni podaci*

Krvna grupa --- ▾

*Vašu krvnu grupu popunjava djelatnik pri prvom doniranju krvi.

Već postoji darivatelj krvi s tim OIB-om.

Kreiraj račun

Slika 5.8: Rezultati drugog testa - greška zbog postojećeg korisnika

- 3. prijava radnika - nova donacija (uključuje traženje donora) - odjava: Testira osnovnu funkcionalnost svakog radnika.

Trueblood ☰

Nova donacija

Stvori donora Pronađi donora Uredi donora Prošle donacije

Mjesto doniranja

Osobni podaci

donorId: 1000001

Ime: Jakov Prezime: Matošić

OIB: 40692048012

Krvna grupa O- ▾

Zdravstveni podaci

Slika 5.9: Rezultati trećeg testa - uspješno pronađeni korisnik i započeta donacija

- 4. Prijava radnika - registracija novog donora sa greškama (krvna grupa nedefinirana, neispravna adresa e-pošte) - odjava:

Spol: Muški • Ženski ○

01/01/1999 Zagreb

Doma 123

Kontakt podaci

jakov.matosic

0911540905

Mjesto zaposlenja (firma) Kontakt (poslovni)

Zdravstveni podaci*

Krvna grupa

*Vašu krvnu grupu popunjava djelatnik pri prvom doniranju krvi.

Email nije validan.

Slika 5.10: Rezultati četvrtog testa - greška

01/01/1999 Zagreb

Doma 123

Kontakt podaci

jakov.matosic@gmail.com

0911540905

Mjesto zaposlenja (firma) Kontakt (poslovni)

Zdravstveni podaci*

Krvna grupa

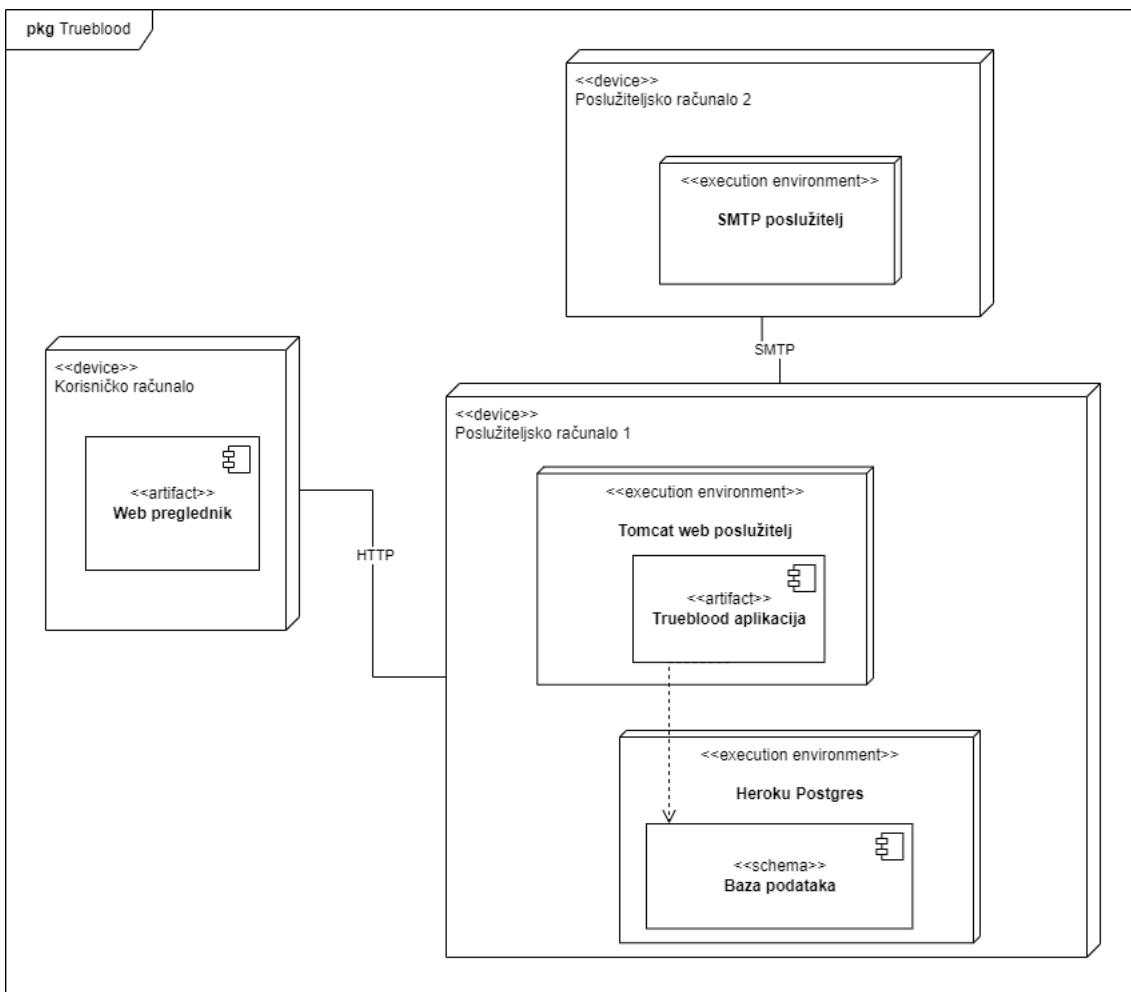
*Vašu krvnu grupu popunjava djelatnik pri prvom doniranju krvi.

Krvna grupa nije validna, validne krvne grupe su: A+, A-, B+, B-, O+, O-, AB+, AB-

Kreiraj račun

Slika 5.11: Rezultati četvrtog testa - greška

5.3 Dijagram razmještaja



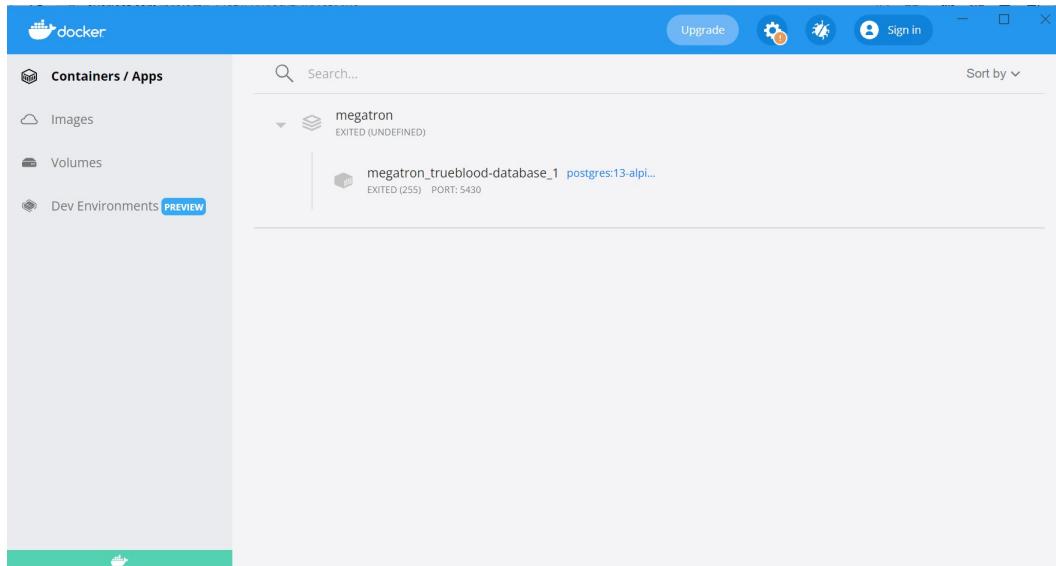
Slika 5.12: Dijagram razmještaja

Dijagram razmještaja je UML dijagram koji opisuje topologiju sustava i prikazuje odnos sklopoških i programske dijelova. Korisničkom sučelju aplikacije pristupa korisnik sa svog računala putem web preglednika. HTTP vezom dohvaca i šalje podatke na poslužiteljsko računalo na kojem se nalaze web poslužitelj i poslužitelj baze podataka. Web poslužitelj također kontaktira SMTP poslužitelj kako bi se odvila funkcionalnost slanja mailova.

5.4 Upute za puštanje u pogon

Instalacija i pokretanje Docker servisa

- Preuzmите prikladnu verziju Dockera za svoj operacijski sustav
- Pokrenite Docker servis



Slika 5.13: Korisničko sučelje Dockera

- U terminalu pozicioniranom u korijenskom direktoriju projekta upišite naredbu

```
$ docker-compose up
```

```
megatron > docker-compose up
Starting megatron_trueblood-database_1 ... done
Attaching to megatron_trueblood-database_1
trueblood-database_1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
trueblood-database_1 | 2022-01-14 00:12:26.881 UTC [1] LOG:  starting PostgreSQL 13.4 on x86_64-pc-linux-musl, compiled by gcc (Alpine 10.3.1_git20210424) 10.3.1 20210424, 64-bit
trueblood-database_1 | 2022-01-14 00:12:26.881 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
trueblood-database_1 | 2022-01-14 00:12:26.881 UTC [1] LOG:  listening on IPv6 address "::", port 5432
trueblood-database_1 | 2022-01-14 00:12:26.885 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
trueblood-database_1 | 2022-01-14 00:12:26.889 UTC [22] LOG:  database system was interrupted; last known up at 2022-01-14 00:12:20 UTC
trueblood-database_1 | 2022-01-14 00:12:27.084 UTC [22] LOG:  database system was not properly shut down; automatic recovery in progress
trueblood-database_1 | 2022-01-14 00:12:27.086 UTC [22] LOG:  invalid record length at 0/190F688: wanted 24, got 0
trueblood-database_1 | 2022-01-14 00:12:27.086 UTC [22] LOG:  redo is not required
trueblood-database_1 | 2022-01-14 00:12:27.096 UTC [1] LOG:  database system is ready to accept connections
```

Slika 5.14: Pokretanje Docker spremnika

- Otvorite drugi terminal te u korijenskom direktoriju projekta upišite naredbu

```
$ psql -h localhost -p 5430 -d Trueblood -U admin
```

i password "admin"

```
[~ > psql -h localhost -p 5430 -d Trueblood -U admin
[Password for user admin:
psql (14.1, server 13.4)
Type "help" for help.

[Trueblood=# \d
              List of relations
 Schema |        Name         |   Type   | Owner
-----+----------------+-----+-----+
 public | bank_worker      | table  | admin
 public | blood_supply     | table  | admin
 public | donation_seq     | sequence | admin
 public | donation_try     | table  | admin
 public | donor            | table  | admin
 public | flyway_schema_history | table | admin
 public | secure_token     | table  | admin
 public | token_seq         | sequence | admin
 public | user_account      | table  | admin
 public | user_seq          | sequence | admin
(10 rows)

Trueblood=# ]
```

Slika 5.15: Spajanje na bazu pomoću alata PSQL

Postavljanje frontenda

- Preuzmite i instalirajte Node.js i [npm](#)
- Preuzmite yarn narebom

```
$ npm install --global yarn
```

- Pokretanjem naredbe

```
$ yarn install
```

instalirat će se potrebni paketi

- Razvojni poslužitelj (engl. *development server*) pokreće se naredbom

```
$ yarn start
```

```
PS C:\Users\doran\IdeaProjects\TrueBlood\IzvorniKod\frontend> yarn start
yarn run v1.22.17
$ webpack-dev-server --mode=development --open --hot
<i> [webpack-dev-server] Project is running at:
<i> [webpack-dev-server] Loopback: http://localhost:8080/
<i> [webpack-dev-server] On Your Network (IPv4): http://10.129.2.238:8080/
<i> [webpack-dev-server] Content not from webpack is served from 'C:\Users\doran\IdeaProjects\TrueBlood\IzvorniKod\frontend\public' directory
<i> [webpack-dev-server] 404s will fallback to '/index.html'
<i> [webpack-dev-middleware] wait until bundle finished: /
assets by path *.png 172 KiB
```

Slika 5.16: Pokretanje razvojnog poslužitelja

Javno puštanje aplikacije u pogon

- Napravite [Heroku korisnički račun](#)
- Instalirajte [Heroku CLI](#)
- Koristeći naredbu

```
$ heroku login
```

pokrenite postupak autorizacije

Primjeri korištenja Heroku alata

Navedeni primjeri mogu se koristiti i s trueblood-fe

- Za ponovno pokretanje aplikacije upišite naredbu

```
$ heroku restart --app trueblood-be
```

- Za uvid u zapisnik aplikacije

```
$ heroku logs --tail --app trueblood-be
```

- Pregled trenutnih build radnji

```
$ heroku builds --app trueblood-be
```

- Povezivanje s bazom pokreće se naredbom

```
$ heroku pg:psql --app trueblood-be
```

ova naredba nije dostupna za trueblood-fe

Status cjevovoda moguće je provjeriti na GitLabu. U slučaju podbacivanja cjevovoda te dobivanja poruke *Your account has reached its concurrent builds limit* u zapisniku aplikacije, potrebno je ponovno pokrenuti aplikaciju.

6. Zaključak i budući rad

6.1 Analiza zadatka

Timski zadatak bio je analizirati traženi sustav, podijeliti se na podtimove koji će se baviti pojedinim aspektima sustava i napisjetku napraviti dogovorene zadatke.

6.2 Postignuti rezultati

Unatoč ne tako dugom vremenskom roku i vrlo ograničenom predznanju svih članova tima, iznimno smo zadovoljni postignutim rezultatom te smatramo da smo zadatak ispunili u potpunosti. Zadatke smo izvršavali redovito, a nismo se ustručavali ni tražiti pomoć od asistenta kada bismo negdje zapeli. Sastanke s asistentom koristili smo kao sastanke s naručiteljem projekta, na kojima bismo razjašnjavali nedoumice oko zadatka, tražili dodatne informacije i potpitanjima sortirali prioritete pri implementaciji.

Zadatak smo rješavali strukturirano, najprije razmatrajući obrasce uporabe sustava, a potom smo krenuli na konkretnu implementaciju. Tim smo podijelili u dio koji se bavio poslužiteljskom stranom (*backend*), dio koji se bavio klijentskom stranom (*frontend*) i dio koji se bavio izradom dokumentacije. Timovi nisu bili statični, već je u vremenu dolazilo do izmjena, kako zbog potrebe, tako i zbog želje članova tima da iskoriste rad na projektu kao priliku za učenje novih tehnologija koje će na budućim projektima moći koristiti kao korisno predznanje.

6.3 Izazovi u radu

Izazovi na koje smo nailazili najčešće su bili vezani uz nedostatak predznanja, a najčešće smo ih rješavali ili samostalno tražeći odgovore na internetu, ili u komunikaciji s ostalim članovima tima koji su nailazili na slične probleme. Kao što je već spomenuto, probleme koje nismo znali riješiti sami proslijedili smo asistentima kako bismo što prije mogli nastaviti dalje sa implementacijom. Osim toga, ponekad je problem znao predstavljati nedostatak komunikacije, točnije stajanje

određenih dijelova projekta zbog neadekvatnog prenošenja informacija. Ipak, do kraja projekta naučili smo iz pogrešaka i poboljšali aspekte koje smo mogli.

Izazov koji nas je pratio kroz cijelo trajanje projekta je nedostatak vremena, što zbog relativno kratkog vremenskog roka, to i zbog ostalih fakultetskih i poslovnih obaveza. Predmet *Programsko inženjerstvo* definira opterećenje nesrazmjerno stvarnom opterećenju, pogotovo s obzirom na udio bodova koji nosi izrada ovog projekta (što se izravno može vidjeti iz tablice sa satima utrošenima na projekt).

6.4 Mogućnosti za daljnji rad

Ipak, bez obzira na nesrazmjerno nastavno opterećenje, ovaj projekt jedna je od najkorisnijih stvari koje smo dosad radili na fakultetu, pružajući nam uvid u rad u timu na konkretnom projektnom zadatku, a to je posao s kojim ćemo se svi u timu jednog dana vrlo vjerojatno susresti. Ne samo da smo stekli puno konkretnog znanja, nego smo se poboljšali i u međuljudskim odnosima i bolje pripremili za buduće radno mjesto.

Kako je opisano u poglavlju sa opisom projektnog zadatka, mnogo nam je dodatnih ideja padalo na pamet. Te ideje uglavnom nismo implementirali, ali su ostavljene kao mogućnost za budući rad. Iako se ovaj projekt koristi primarno za nastavu, daljnji rad i implementacija dodatno osmišljenih funkcionalnosti bio bi odlična vježba za daljnje usavršavanje članova tima. S obzirom na ugodnu radnu atmosferu i pokazanu odgovornost članova tima, ostaje otvorena mogućnost za daljnji rad na projektu.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
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. <https://www.techopedia.com/definition/24649/three-tier-architecture>
8. <https://www.w3.org/TR/wsdl.html>
9. <https://reactjs.org/>
10. <https://spring.io/projects/spring-boot>
11. <https://stackoverflow.com/>

Indeks slika i dijagrama

2.1 Postojeća stranica HZTM-a	6
3.1 Dijagram obrazaca uporabe 1 - Administracija računa	33
3.2 Dijagram obrazaca uporabe 2 - Proces aktivacije računa	34
3.3 Dijagram obrazaca uporabe 3 - Uređivanje postojećih računa	35
3.4 Dijagram obrazaca uporabe 4 - Proces doniranja	36
3.5 Dijagram obrazaca uporabe 5 - Javni web i mogućnosti nakon prijave	37
3.6 Dijagram obrazaca uporabe 6 - Potrošnja krvi i povezane obavijesti .	38
3.7 Sekvencijski dijagram 1 - Potrošnja krvi i povezane obavijesti	39
3.8 Sekvencijski dijagram 2 - Potrošnja krvi i povezane obavijesti	41
4.1 Arhitektura sustava	44
4.2 Dijagram baze podataka	49
4.3 Dijagram controller klasa	50
4.4 Dijagram model klasa	52
4.5 Dijagram DTO klasa	53
4.6 Dijagram Repository sučelja i Service klasa	54
4.7 DS1 Trueblood web aplikacija	55
4.8 DA1 Postupak doniranja	56
4.9 DA2 Djelatnik donor administracija računa	58
4.10 Dijagram komponenti	60
5.1 Testovi, prvi dio	63
5.2 Testovi, drugi dio	64
5.3 Testovi, treći dio	64
5.4 Testovi, četvrti dio	65
5.5 Testovi, peti dio	66
5.6 Rezultati testiranja	66
5.7 Rezultati prvog testa - profilna stranica prijavljenog korisnika . . .	67
5.8 Rezultati drugog testa - greška zbog postojećeg korisnika	68
5.9 Rezultati trećeg testa - uspješno pronađeni korisnik i započeta donacija	68

5.10 Rezultati četvrтog testa - greška	69
5.11 Rezultati četvrтog testa - greška	69
5.12 Dijagram razmještaja	70
5.13 Korisničko sučelje Dockera	71
5.14 Pokretanje Docker spremnika	71
5.15 Spajanje na bazu pomoću alata PSQL	72
5.16 Pokretanje razvojnog poslužitelja	73
6.1 Graf aktivnosti - 1. dio	89
6.2 Graf aktivnosti - 2. dio	90

Indeks tablica

1 Dnevnik promjena dokumentacije	3
1 Tablica <i>korisnik</i> u bazi podataka	46
2 Tablica <i>donor</i> u bazi podataka	47
3 Tablica <i>djelatnik banke</i> u bazi podataka	47
4 Tablica <i>zaliha krvi</i> u bazi podataka	48
5 Tablica <i>pokušaj donacije</i> u bazi podataka	49
1 Tablica aktivnosti po članovima tima	87

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 5. listopada 2021.
- Prisustvovali: Svi
- Teme sastanka:
 - Uspostava svih članova tima

2. sastanak

- Datum: 17. listopada 2021.
- Prisustvovali: Svi
- Uspostava GitLab repozitorija
- Uspostava SSH ključeva
- Uspostavljena platforma za komunikaciju (Slack)
- Prva okvirna podjela poslova:
 - Ana - backend (Spring)
 - Vukota - backend (Spring)
 - Jakov - backend, testovi
 - Dora - backend
 - Marko - full-stack
 - Toni - full-stack, organizacija
 - Borna - frontend (UI)
- Zadatak za prvi tjedan:
 - Do četvrtka:
 - * Upoznati se sa radnom okolinom, gitom, softverom koji se koristi
 - * Pročitati zadatak s razumijevanjem
 - * Zabilježiti sva pitanja koja se pojave
 - Četvrtak:
 - * Sastanak s asistentima (labos) u 11:00

* Sastanak u 13:00

3. sastanak

- Datum: 21. listopada 2021.
- Sastanak s asistentom i demosom: odgovorili na pitanja
 - Prisutni: Toni, Ana, Jakov, Marko
 - u aplikaciji napraviti formular s pitanjima. (sve su da ne pitanja, nema opisivanja)
 - 1 tablica sa svim userima u sustavu - ime prezime username pass mail role id(općeniti)
 - aktivacijski mail ako na doniranju kreira račun
 - *pazi ima jos jedan nacin pada
 - DonorId je primarni ključ, a OIB alternativni ključ
- Sastanak grupni:
 - Prisutni: Ana, Vukota, Jakov, Toni, Marko, Dora
 - crtanje UC dijagrama
 - backend brainstorming oko funkcionalnosti
 - userId=donorId
 - za account postoje 2 bool varijable 1) account inicijalno aktiviran (pokreće se otvaranjem linka u mailu) i 2) account trajno deaktiviran (od strane admina)
- Zadatci do idućeg sastanka:
 - Ana - radi bazu (edrplus)
 - Marko - poboljšati organizaciju gita, inicijalizacija direktorija za kod
 - Dora - UC dijagrami, sekvencijski dijagrami
 - Toni - UC dijagrami, sekvencijski dijagrami
 - Vukota - postavljanje LaTeX-a i inicijalizacija projektne dokumentacije
 - Jakov - Uvođenje u Spring
 - Borna - Uvođenje u React
- Idući sastanak: Četvrtak(28.10.) - sastanak sa asistentima i grupni sastanak (11h, 13h)

4. sastanak

- Datum: 28. listopada 2021.
- Sastanak s asistentom i demosom: odgovori na pitanja, demonstracija inicijalnih UC, demonstracija dizajna stranice

- Prisustvovali: Toni, Ana, Dora, Borna, Jakov, Luka
- odgovorena pitanja iz maila
- prokomentirani UC dijagrami – treba reducirati
- predstavljen model baze podataka, spomenute nove mogućnosti platforme
- predstavljen idejni dizajn frontenda
- predloženi alati Overleaf (Latex), Dbdiagram.io, Hibernate
- Sastanak grupni:
 - Prisustvovali: svi
 - dogovor oko dalnjeg frontend dizajna
 - dogovoren sastanak backend podtima (subota) radi inicijalizacije dockera
- Zadatci do idućeg sastanka:
 - Toni - Popravak i nadogradnja UC dijagrama i crtanje sekvencijskih; inicijalizacija dokumentacije
 - Dora, Jakov - Upoznavanje sa Springom
 - Ana - dorada baze podataka, istražiti opcije predstavljene na sastanku; docker
 - Vukota - docker, pisanje inicijalnog readme.md
 - Borna - nastavak idejnih dizajna stranice, inicijalna konstrukcija frontend dijela stranice
 - Marko - održavanje gita, rad s Bornom na frontendu
- Idući sastanak: Backend - subota(30.10.), ostali - četvrtak (4.11.)

4.1. sastanak (backend)

- Datum: 30. listopada 2021.
- Prisustvovali: Ana, Luka, Jakov, Dora
- uspostava Dockera (baza)
- početak rada na useru (Ana)
- početak rada na bloodSupplyju (Dora i Jakov)
- pomaganje (Luka)
- Idući sastanak: svi - četvrtak (4.11.)

5. sastanak

- Datum: 4. studenog 2021.
- Sastanak s asistentima:

- Prisustvovali: Toni, Ana, Dora, Borna, Jakov, Vukota
- maknuti field brojDonacija iz donora
- maknuti NOT NULL sa bloodType
- glavni sudionik ne pisati pod sudionici u opisima UC-jeva
- UC dodavanje računa moguće razdvojiti na tri (da imamo u svakom po jednog glavnog sudionika)
- za generičku funkcionalnost treba pokazati da imamo session, držimo session, i možemo kreirati usera (U tablici 3 prikazano što treba)
- Grupni sastanak:
 - Prisustvovali: Svi
 - edukacija o gitu
 - dogovor da idući tjedan bude gotov login page i registracija
 - oformljen zajednički account za uređivanje dokumentacije
 - pri stvaranju accounta, na mail se šalje link za aktivaciju I RANDOM LOZINKA, a postojat će opcija "Promijeni lozinku"
 - za generičku funkcionalnost, lozinka se ispisuje u terminal
- Zadatci za idući tjedan:
 - Vukota: završi sve modele (klase kao relacije u bazi)
 - Ana: Endpoint (create user)
 - Dora: Endpoint (create donor)
 - Jakov: Istražiti Heroku (deploy aplikacije), Testovi (backend)
 - Borna: Dovršiti login page (popravci) i dizajn za stranicu registracije
 - Marko: spajanje frontenda i backenda
 - Toni: Popravak UC dijagrama, pisanje dokumentacije

6. sastanak

- Datum: 10. studenog 2021.
- Prisustvovali: svi
- rad na web securityju
- brainstorm o mergeu frontenda i backenda kao priprema za deploy
- podjela posla oko dokumentacije
- Zadatci za dalje:
 - Borna: Napisati ukratko dio vezan za frontend u dokumentaciju - arhitektura sustava
 - Marko, Ana, Vukota: Web security (omogućiti funkcionalnost logina)

- Toni, Jakov: Rad na dokumentaciji

6.1. sastanak

- Datum: 11. studenog 2021.
- Prisustvovali: Toni, Vukota, Marko
- Na idućem sastanku:
 - prezentacija generičke funkcionalnosti
 - postaviti sva pitanja koja još imamo za finalni upload u petak u 23:59
 - generička funkcionalnost treba pokazati session (refresh čuva session)
 - sami napravimo neki mail za slanje aktivacijskih mailova
 - idući sastanak u srijedu u 11:00 - svi su obavezni

7. sastanak

- Datum: 17. studenog 2021.
- Prisustvovali: svi
- potrebno ispraviti UC opise na puno strože definiranje akcija i reakcija u sustavu
- potrebno proširiti opis sustava (poglavlje 2)
- osigurati pouzdanost sustava do krajnje predaje u petak
- napraviti neke quality-of-life promjene
- prijavljenim korisnicima onemogućiti ponovnu prijavu
- redizajnirati endpointe
- promijeniti login na AWT-token
- ispraviti i dovršiti poglavlje o arhitekturi sustava

8. sastanak - Koordinacija prije kolokviranja

- Datum: 5. prosinca 2021.
- Prisustvovali: svi
- Prezentirani dijelovi projekta kako bi se svi upoznali sa bitnim funkcionalnostima
- Dogovor o preraspodjeli odgovornosti na projektu
 - Borna: frontend
 - Toni: frontend + koordinacija
 - Marko: fullstack
 - Ana: backend

- Vukota: backend
- Dora: dokumentacija + backend
- Jakov: testovi + backend

9. sastanak

- Datum: 8. prosinca 2021.
- Prisustvovali: svi
- Dogovor oko demokratske raspodjele bodova na prvoj predaji
- Dogovor o prioritetima u nastavku projekta
- Oformljen dokument s napretkom na pojedinim stavkama projekta

10. sastanak

- Datum: 16. prosinca 2021.
- Prisustvovali: svi
- Sastanak s assitentom
 - Komentiran prostor za razvoj nakon prve predaje (dostupno u dokumentu na slacku)
 - Dogovorena mogućnost povratnih informacija tijekom praznika
- Međusobno izvještavanje o napretku
- Raspodijeljen posao, dogovoreni prioriteti

11. sastanak

- Datum: 4. siječnja 2022.
- Prisustvovali: Toni, Ana, Vukota, Marko, Jakov, Borna
- Prezentacija dosadašnjeg napretka
- Rješavanje problema oko autorizacije zahtjeva s frontend-a
- Rješavanje problema vezanih za slanje maila i validaciju računa

12. sastanak

- Datum: 4. siječnja 2022.
- Prisustvovali: Ana, Vukota, Toni, Borna, Jakov
- Koordinacija zadataka
- Pokušaj popravka autorizacije na backendu

13. sastanak - sastanak s asistentom

- Datum: 5. siječnja 2022.
- Prisustvovali: Dora, Marko, Toni, Jakov

- Razgovor o dosadašnjem napretku i očekivanim komponentama pri predaji
- Rasprava o mogućnostima popravka dijagrama
- Popravci:
 - Urediti kontakt i FAQ
 - Informacije o krvnim grupama koje nedostaju da bude svaka u novi red
 - Tražilica treba po defaultu tražiti samo aktivne korisnike
 - Upute za puštanje u pogon bi bile dosta dobro napisane kao naš readme pa možemo to iskoristiti

14. sastanak

- Datum: 12. siječnja 2022.
- Prisustvovali: Ana, Vukota, Borna, Toni
- Prolazak kroz funkcionalnosti po tablici use-caseova
- Bilježenje preostalog posla i podjela po osobama (rezultat na slacku)

15. sastanak - sastanak s asistentom

- Datum: 13. siječnja 2022.
- Prisustvovali: Ana Dora, Toni, Jakov, Borna
- Završni komentari na dijagrame (dovršiti dijagram komponenti, u dijagramu stanja izmjene oko ujednačenosti jezika)
- Prezentacija funkcionalnosti
- Komentirano ograničenje schedulera vezano za Heroku timeout i zaključak da nad time nemamo utjecaja i da je to OK

Tablica aktivnosti

Doprinosi u aktivnostima navedeni su u satima po članovima grupe po aktivnosti.

Table 6.1: Tablica aktivnosti po članovima tima

	Toni Ivanković	Ana Gršković	Borna Mahović	Jakov Matošić	Dora Nevidal	Marko Opačić	Luka Vukota
Upravljanje projektom	15	2					
Opis projektnog zadatka	5.5						
Funkcionalni zahtjevi	6						
Opis pojedinih obrazaca	15						
Dijagram obrazaca	7.5				1		
Sekvencijski dijagrami	3				0.5		
Opis ostalih zahtjeva	2						
Arhitektura i dizajn sustava		4		20		2	
Dijagram razreda				7			
Dijagram stanja					4		
Dijagram aktivnosti					4		
Dijagram komponenti	2				0.5		
Korištene tehnologije i alati	3.5	1.5	1.5	1.5	1.5	1.5	1.5
Ispitivanje programskog rješenja				12			
Dijagram razmještaja					1		
Upute za puštanje u pogon					1	1	
Proučavanje tehnologija	5	6	6	11	8	5	5
Dnevnik sastajanja	2.5				2.5		

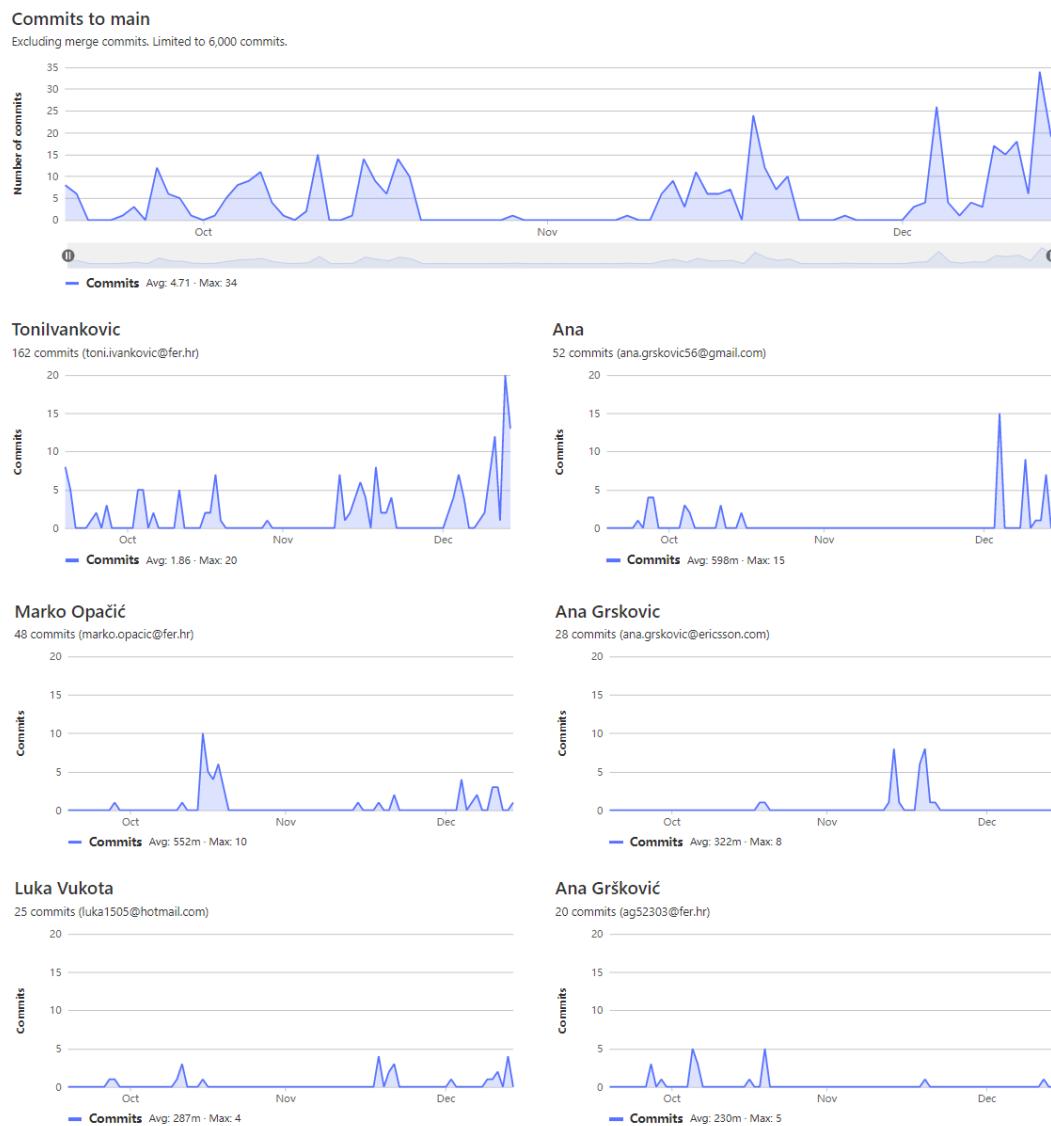
Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

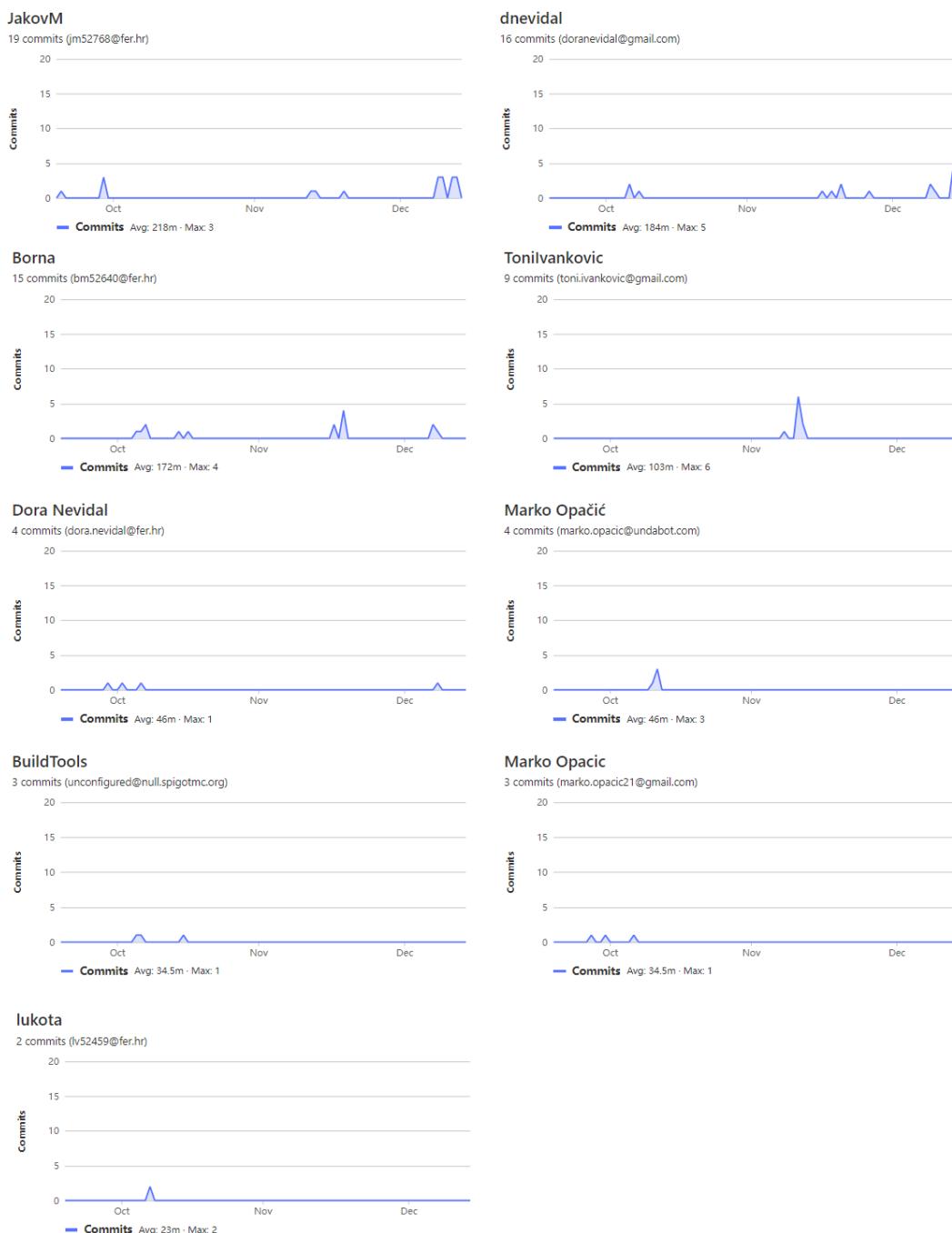
		Toni Ivanković	Ana Gršković	Borna Mahović	Jakov Matosić	Dora Nevidal	Marko Opačić	Luka Vukota
Zaključak i budući rad	1							
Popis literature	0.25							
<i>izrada početne stranice</i>			11			7		
<i>izrada baze podataka</i>	0.5	4					1	
<i>spajanje s bazom podataka</i>		3					2	
<i>back end (poslužiteljska strana)</i>	12	57		11	12	24	39	
<i>front end (klijentska strana)</i>	55		10			8		
<i>održavanje cjevovoda</i>						1		

Dijagrami pregleda promjena

S obzirom da je Gitlab za neke članove tima bilježio različite mail adrese ovisno o računalu s kojeg su pristupali, neki članovi tima imaju po više grafova. Napominjemo da brojevi *commitova* (u slučaju našeg projekta) nisu relevantno mjerilo količine odrađenog posla zbog različite prirode zaduženja i navika *commitanja* pojedinih članova.



Slika 6.1: Graf aktivnosti - 1. dio



Slika 6.2: Graf aktivnosti - 2. dio