

## Specifikacija projekta

### 1. Osnovne informacije o sistemu

**Naziv teme:** VitalFlow (Blood Bank App)

**Logo:**



**Naziv tima:** VitalForce

**Nastavna grupa:** Grupa7-TIM28

**Link na repozitorij tima:** <https://github.com/OOAD-2023-2024/OOAD-TIM28.git>

**Članovi tima:**

1. Benjamin Bandić, 19205
2. Muhamed Husić, 19106
3. Amer Mujalo, 19101
4. Bakir Šatara, 19004

**Namjena sistema:**

*Opisati sistem i njegovu namjenu sa maksimalno sedam rečenica. U okviru ovog polja potrebno je objasniti šta sistem treba raditi na apstraktном nivou, bez detaljnog objašnjavanja pojedinačnih funkcionalnosti i načina razlikovanja aktera sistema (što je predmet daljih poglavlja).*

VitalFlow je aplikacija osmišljena kako bi olakšala proces doniranja i distribucije krvi putem intuitivnog korisničkog interface-a. Glavna svrha sistema je pribavljanje i distribucija krvi između donora i klinika. Sistem funkcioniра као посредник који прикупља крвне донације и преноси их клиникама које су у потреби, чиме се pojednostављује процес и смањује административни терет. Интегрирајући разлиčите функционалности, VitalFlow доприноси побољшању доступности и ефикасности процеса донирања крви у јавнici.

## 2. Funkcionalnosti (poslovni procesi) sistema

*Opisati 6 do 8 najznačajnijih funkcionalnosti sistema (u zavisnosti od broja članova u timu). Funkcionalnosti sistema predstavljaju usluge koje sistem pruža korisnicima. Sve funkcionalnosti pripadaju nekoj od različitih vrsta:*

- *Usluga sistema - u svrhu ostvarivanja krajne usluge sistema,*
- *Perzistencija podataka (CRUD operacije)*
- *Asinhrona operacija - operacije koje koriste principe asinhrone obrade zahtjeva*
- *Operacija sa specifičnim algoritmom obrade - operacije koje koriste specifične algoritme obrade podataka,*
- *Korištenje vanjskog uređaja - operacije u kojima se vrši korištenje vanjskih uređaja.*

*Neophodno je navesti barem po jednu funkcionalnost svake od različitih vrsta.*

### 1) Naziv funkcionalnosti: Registracija i logiranje donora krvi

**Vrsta funkcionalnosti:** Usluga sistema

**Opis funkcionalnosti:** Omogućava novim korisnicima registraciju za donora krvi putem aplikacije

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Pri otvaranju aplikacije, korisnik će imati mogućnost registracije, ukoliko to već nije uradio. Radi se o formi u kojoj će morati popuniti svoje osnovne informacije poput imena, prezimena, e-mail adrese, broj telefona, krvne grupe... Zaposlenici će već imati dodijeljene username-e i password-e, te neće morati da se registriraju.

### 2) Naziv funkcionalnosti: Spremanje podataka o donorima

**Vrsta funkcionalnosti:** Perzistencija podataka (CRUD operacija)

**Opis funkcionalnosti:** Omogućuje unos, čitanje, ažuriranje i brisanje podataka o registriranim donorima u bazi podataka sistema.

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Donor kada se registruje i logira, imat će mogućnost korištenja R i U operacija za svoje personalne informacije. Zaposlenici će također imati pristup R i U operacijama, dok će Admin imati pristup kompletnom CRUD-u.

### 3) Naziv funkcionalnosti: Slanje notifikacija

**Vrsta funkcionalnosti:** Asinhrona operacija

**Opis funkcionalnosti:** Koristi princip asinhronne obrade zahtjeva kako bi automatski upozoravala donore o trenutnim potrebama za određenim tipovima krvi.

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Zavisno od zalihe određenih krvnih grupa, donori će dobijati notifikacije i potencijalno mail-ove u vezi manjka odredene krvne grupe. Ove notifikacije i mail-ovi će biti personalni, u smislu da će dolaziti notifikacije manjka odredene krvne grupe upravo onim donorima koji pripadaju toj krvnoj grupi. Kritična linija zalihe će kasnije biti određena.

- 4) **Naziv funkcionalnosti:** Provjera pada ispod kritične linije

**Vrsta funkcionalnosti:** Operacija sa specifičnim algoritmom obrade

**Opis funkcionalnosti:** Razvija se kompletni algoritam koji će provjeravati stanje zaliha u baci u vidu zadržavanja zalihe iznad kritične linije

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Kritičnu liniju je važno kvalitetno definisati, kako ne bi donorima dolazile nebitne notifikacije. Nakon što se definira kritična linija za svaku krvnu grupu, zavisno od uobičajene upotrebe, razvit ćemo određeni algoritam koji će pratiti da zalihe ne spadnu ispod te kritične linije, te ako spadnu, kao što smo već naveli, slat ćemo notifikacije i mail-ove donorima koji su relevantni za specifičan slučaj/krvnu grupu.

- 5) **Naziv funkcionalnosti:** Korištenje mail-a za slanje informacija

**Vrsta funkcionalnosti:** Korištenje vanjskog uređaja

**Opis funkcionalnosti:** Omogućuje slanje informacija klinici putem mail-a

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Klinika će imati opciju da popuni formu gdje će moći naglasiti koje su njene potrebe. Ta forma će se nalaziti u sklopu aplikacije. Poslije popunjavanja te forme, formu će provjeriti zaposlenici te poslati mail specifičnoj klinici sa odgovorom. Forma bi sadržavala biranje specifične klinike u padajućoj listi klinika sa kojim sarađujemo, zatim bi sadržavala polje za unos/biranje e-mail-a, te odabir potrebne krvne grupe i količine.

- 6) **Naziv funkcionalnosti:** Pregled profila donora

**Vrsta funkcionalnosti:** Usluga sistema

**Opis funkcionalnosti:** Omogućuje donorima da pregledaju svoje donorske profile kako bi pratili svoje donacije i personalne informacije.

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Donor nakon registracije i logovanja će imati mogućnost pregleda svog profila, na kojem će se nalaziti njegove personalne informacije, te neke dodatne informacije poput ostavljenog rekorda termina u kojem treba doći donirati krv.

7) **Naziv funkcionalnosti:** Hub

**Vrsta funkcionalnosti:** Usluga sistema

**Opis funkcionalnosti:** Omogućuje donorima izbor određenih termina u kojima mogu doći i donirati krv.

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

U pitanju je centralno područje ili hub koje će omogućiti donorima da se prijave za neki specifični termin za doniranje krvi. Taj termin će biti sačuvan u aplikaciji, te će biti prikazan i na njegovom personalnom profilu. Bit će u mogućnosti otkazati termin ukoliko bude naknadno nastao problem. Kako je ovo centralno područje, zaposlenici i admin će imati dostupan i nevidljivi dio/izbor gdje će moći provjeriti i trenutno stanje zaliha kroz sistem. Također, zaposlenici i admin će imati pristup pregledi svih termina izabranih, na isti način kao što imaju pristup zalihama.

8) **Naziv funkcionalnosti:** Stanje Zalihe

**Vrsta funkcionalnosti:** Perzistencija podataka (CRUD operacija)

**Opis funkcionalnosti:** Omogućuje korištenje CRUD operacija nad stanju zaliha kroz sistem

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Kada se uloguje admin ili neki zaposlenik, kroz hub će imati opciju provjere stanja zalihe krvi. Također, kada donor donira krv u određenom terminu, zaposlenik će dodati određenu količinu novo-donirane krvi ili će oduzeti od originalnog stanja određenu količinu koju prevoze u neku kliniku u saradnji sa bankom.

### 3. Akteri sistema

Potrebno je navesti najmanje tri aktera sistema.

Vrste aktera:

- Korisnik sistema
- Zaposlenik sistema
- Administrator

Neophodno je navesti barem po jednog aktera za svaku od različitih vrsta.

Korisnici usluga sistema

a) **Naziv aktera:** Donor krvi

**Vrsta aktera:** Korisnik usluge

**Funkcionalnosti u kojima akter učestvuje:**

Funkcionalnost sistema	Način učešća
Registracija donora krvi	Mogućnost uređivanja
Pregled vlastitog profila	Mogućnost pregleda
Izbor termina doniranja krvi	Mogućnost uređivanja

b) **Naziv aktera:** Tehničar

**Vrsta aktera:** Zaposlenik sistema

**Funkcionalnosti u kojima akter učestvuje:**

Funkcionalnost sistema	Način učešća
Čitanje i ažuriranje podataka donora	Mogućnost uređivanja
Pristup zalihamama	Mogućnost uređivanja
Pregled svih izabranih termina	Mogućnost uređivanja

c) **Naziv aktera:** Admin

**Vrsta aktera:** Administrator

**Funkcionalnosti u kojima akter učestvuje:**

*Način učešća:*

- *Mogućnost pregleda*
- *Mogućnost uređivanja*

Funkcionalnost sistema	Način učešća
CRUD operacije nad podacima donora i zaposlenika	Mogućnost uređivanja
Pristup zalihamama	Mogućnost uređivanja
Pregled svih izabranih termina	Mogućnost uređivanja

#### 4. Nefunkcionalni zahtjevi sistema

*Opisati najmanje tri najznačajnija nefunkcionalna zahtjeva sistema. Nefunkcionalni zahtjevi predstavljaju ograničenja koja sistem mora zadovoljiti kako bi mogao ispravno obavljati svoje funkcionalnosti. Validacije polja za unos vrijednosti ne predstavljaju nefunkcionalne zahtjeve.*

##### 1) Naziv nefunkcionalnog zahtjeva: Sigurnost

**Opis:** Sistem zahtjeva visok stupanj sigurnosti kako bi se osigurala zaštita korisničkih računa i privatnih podataka.

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

Korisničke šifre moraju biti duge najmanje 8 znakova i sadržavati barem jedno veliko slovo, jedan broj i jedan poseban znak. Ovo osigurava da su šifre dovoljno "teške" i otežava potencijalnim napadačima da ih dešifriraju ili hakiraju.

##### 2) Naziv nefunkcionalnog zahtjeva: Dostupnost

**Opis:** Sistem mora biti stalno dostupan kako bi korisnici mogli koristiti usluge u bilo koje vrijeme.

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

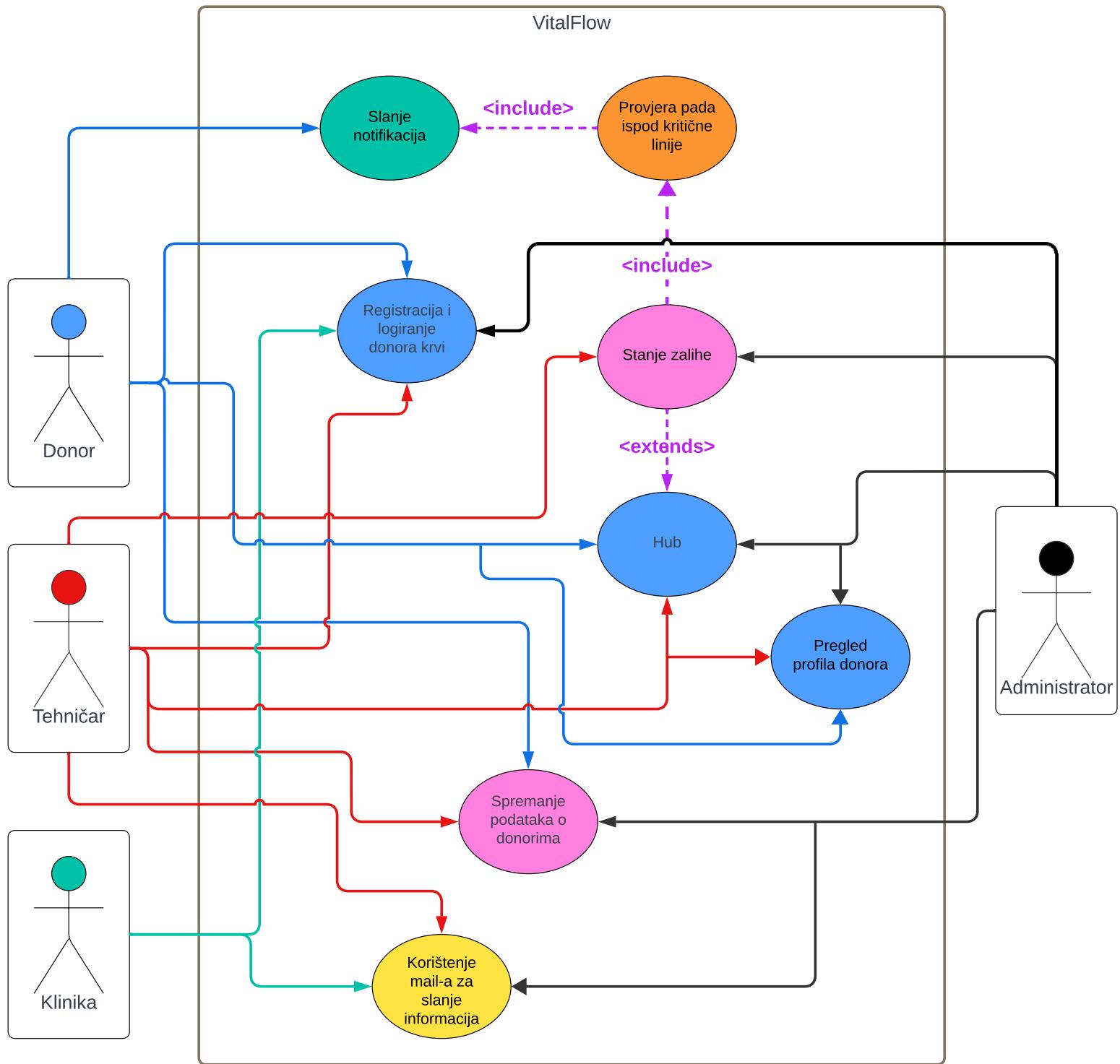
Downtime sistema treba biti minimalan, a planirani prekidi rada (npr. zbog održavanja) trebaju biti najavljeni korisnicima unaprijed.

##### 3) Naziv nefunkcionalnog zahtjeva: Performanse

**Opis:** Sistem mora biti sposoban brzo reagirati na korisničke zahtjeve kako bi osigurao učinkovitu interakciju.

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

Vrijeme odziva aplikacije na korisničke zahtjeve ne smije biti duže od 3 sekunde.



Objektno Orijentisana Analiza i Dizajn

Scenarij 1: Registracija i logiranje donora krvi

Naziv slučaja upotrebe	Registracija i logiranje donora krvi
Opis slučaja upotrebe	Donor pristupa aplikaciji i ponuđen je sa formom za registraciju/prijavu
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Donor je registrovan/prijavljen na sistem
Posljedice – neuspješan završetak	Donor nije registrovan/prijavljen na sistem – podaci nisu ispravni
Primarni akteri	Donor
Ostali akteri	Zaposlenik, Administrator
Glavni tok	Korisniku se otvara forma za upis personalnih informacija u sistem
Alternative/proširenja	/

Tok događaja 1. – Uspješan završetak “Registracija donora”

Donor	Sistem
1. Odabir opcije registracije profila	
2. Popunjavanje potrebnih podataka	
	3. Validacija
	4. Prihvatanje registracije
	5. Prijava

Tok događaja 2. – Nesuspješan završetak “Registracija donora”

Donor	Sistem
1. Odabir opcije registracije profila	
2. Popunjavanje potrebnih podataka	
	3. Validacija
	5. Isticanje pogrešnih podataka

Objektno Orientisana Analiza i Dizajn

Tok događaja 3. – Uspješan završetak “Prijava donora”

Donor	Sistem
1. Odabir opcije prijave donora	
2. Popunjavanje potrebnih podataka	3. Validacija
	4. Prihvatanje prijave
	5. Prijava

Tok događaja 4. – Nespuštan završetak “Prijava donora”

Donor	Sistem
1. Odabir opcije prijave donora	
2. Popunjavanje potrebnih podataka	3. Validacija
	4. Odbijanje prijave
	5. Isticanje pogrešnih podataka

Tok događaja 5. – Uspješan zaršetak „Prijava zaposlenika“

Zaposlenik	Sistem
1. Odabir opcije prijave zaposlenika	
2. Popunjavanje potrebnih podataka	3. Validacija
	4. Prihvatanje prijave
	5. Prijava

Tok događaja 6. – Neuspješan završetak „Prijava zaposlenika“

Zaposlenik	Sistem
1. Odabir opcije prijave zaposlenika	
2. Popunjavanje potrebnih podataka	3. Validacija
	4. Odbijanje prijave
	5. Isticanje pogrešnih podataka



Objektno Orientisana Analiza i Dizajn

Scenarij 2: Spremanje podataka o donorima

Naziv slučaja upotrebe	Spremanje podataka o donorima
Opis slučaja upotrebe	Omogućuje unos, čitanje, ažuriranje i brisanje podataka o registriranim donorima u bazi podataka sistema.
Vezani zahtjevi	/
Preduslovi	Registracija i logiranje donora krvi
Posljedice – uspješan završetak	Mogućnost izvršavanja CRUD operacija nad profilom donora
Posljedice – neuspješan završetak	/
Primarni akteri	Donor
Ostali akteri	Zaposlenik i Administrator
Glavni tok	Donor kada se registruje i logira, imat će mogućnost korištenja R i U operacija za svoje personalne informacije. Zaposlenici će također imati pristup R i U operacijama, dok će Admin imati pristup kompletnom CRUD-u.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Update podataka o donorima ”

Donor	Sistem
1. Izbor pregleda profila	
2. Prikaz profila	
3. Update nad podacima	
	4. Izmjena datih podataka
	5. Iстicanje izvršenih izmjena

*Objektno Orientisana Analiza i Dizajn*

Tok događaja 2. Uspješan završetak “ Update podataka o donorima ”

Zaposlenik	Sistem
1. Izbor pregleda profila	
2. Prikaz profila	
3. Update nad podacima	
	4. Izmjena datih podataka
	5. Iстicanje izvršenih izmjena

Tok događaja 3. Uspješan završetak “ CRUD nad podacima o donorima”

Administrator	Sistem
1. Izbor pregleda profila	
2. Prikaz profila	
3. Izvršavanje CRUD operacija	
	4. Izmjena datih podataka
	5. Isticanje izvršenih izmjena

Objektno Orientisana Analiza i Dizajn

Scenarij 3: Slanje notifikacija

Naziv slučaja upotrebe	Slanje notifikacija
Opis slučaja upotrebe	Koristi princip asinhronne obrade zahtjeva kako bi automatski upozoravala doneure o trenutnim potrebama za određenim tipovima krvi.
Vezani zahtjevi	/
Preduslovi	Registracija i logiranje donora krvi
Posljedice – uspješan završetak	Šalju se notifikacije donorima zavisno od stanja zalihe
Posljedice – neuspješan završetak	/
Primarni akteri	Donor
Ostali akteri	/
Glavni tok	Zavisno od zalihe određenih krvnih grupa, donori će dobijati notifikacije i potencijalno mail-ove u vezi manjka određene krvne grupe. Ove notifikacije i mail-ovi će biti personalni, u smislu da će dolaziti notifikacije manjka određene krvne grupe upravo onim donorima koji pripadaju toj krvnoj grupi.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “Slanje notifikacija”

Sistem	Donor
1. Stanje zalihe određene krvne grupe spalo ispod kritične linije	
2. Slanje notifikacija relevantnim donorima iste krvne grupe	
	3. Primanje datih notifikacija preko aplikacije, te na mail

Objektno Orientisana Analiza i Dizajn

Scenarij 4: Provjera pada ispod kritične linije

Naziv slučaja upotrebe	Provjera pada ispod kritične linije
Opis slučaja upotrebe	Razvija se kompletni algoritam koji će provjeravati stanje zalihe u baci u vidu zadržavanja zalihe iznad kritične linije
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Mogućnost efikasnog praćenja trenutne zalihe određene krvne grupe u bazi, te efikasno ažuriranje u vidu pada ispod granice
Posljedice – neuspješan završetak	/
Primarni akteri	Sistem
Ostali akteri	/
Glavni tok	Sistem uvijek prati stanje zalihe u baci, te ukoliko se desi da padne zaliha određene krvne grupe ispod kritične linije, šalje se notifikacija donorima iste krvne grupe, te se šalje i mail.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Provjera pada ispod kritične linije ”

Sistem
1. Provjera stanja zalihe u baci
2. Zaliha neke krvne grupe spada ispod kritične linije
3. Slanje povratnih podataka kroz sistem u vidu slanja notifikacija



*Objektno Orijentisana Analiza i Dizajn*

*Objektno Orientisana Analiza i Dizajn*

**Scenarij 5: Korištenje mail-a za slanje informacija**

Naziv slučaja upotrebe	Korištenje mail-a za slanje informacija
Opis slučaja upotrebe	Omogućuje slanje informacija klinici putem mail-a
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Uspješna komunikacija klinike i banke
Posljedice – neuspješan završetak	Neuspješna komunikacija klinike i banke
Primarni akteri	Klinika
Ostali akteri	Zaposlenici, Sistem
Glavni tok	Klinika će imati opciju da popuni formu gdje će moći naglasiti koje su njene potrebe. Ta forma će se nalaziti u sklopu aplikacije. Poslije popunjavanja te forme, formu će provjeriti zaposlenici te poslati mail specifičnoj klinici sa odgovorom. Forma bi odabir potrebne krvne grupe, količine i sl.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Korištenje mail-a za slanje informacija ”

Klinika	Sistem	Zaposlenik
1. Otvara formu u aplikaciji, te je popunjava odgovarajućim podacima		
2. Submit forme	3. Validacija	
	4. Šalje formu u sistem gdje je vidljiva zaposleniku	
		5. Provjerava popunjenu formu, te odgovara na dati zahtjev klinici preko mail-a



*Objektno Orijentisana Analiza i Dizajn*

Tok događaja 2. Nespešan završetak “ Korištenje mail-a za slanje informacija ”

Klinika	Sistem	Zaposlenik
1. Otvara formu u aplikaciji, te je popunjava odgovarajućim podacima		
2. Submit forme		
	3. Validacija	
	4. Iстicanje pogrešnih podataka	

Objektno Orientisana Analiza i Dizajn

Scenarij 6: Pregled profila donatora

Naziv slučaja upotrebe	Pregled profila donatora
Opis slučaja upotrebe	Omogućuje donorima da pregledaju svoje donorske profile kako bi pratili svoje donacije i personalne informacije.
Vezani zahtjevi	/
Preduslovi	Registracija i logiranje donora krvi
Posljedice – uspješan završetak	Mogućnost pregleda donorskog profila i svih dodatnih informacija, poput trenutno prijavljenih termina za doniranje
Posljedice – neuspješan završetak	/
Primarni akteri	Donor
Ostali akteri	Zaposlenik, Administrator
Glavni tok	Donor nakon registracije i logovanja ce imati mogućnost pregleda svog profila, na kojem će se nalaziti njegove personalne informacije, te neke dodatne informacije poput ostavljenog rekord-a termina u kojem treba doći donirati krv.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Pregled profila donatora ”

Donor	Sistem
1. Izbor pregleda donorskog profila	
	2. Otvara prozor za pregled profila sa svim relevantnim informacijama
3. Mogućnost pregleda profila uz mogućnost update-a nekih specifičnih podataka	

*Objektno Orientisana Analiza i Dizajn*

Tok događaja 2. Uspješan završetak “ Pregled profila donatora ”

Zaposlenik	Sistem
1. Izbor pregleda donorskog profila	
	2. Otvara prozor za pregled profila sa svim relevantnim informacijama
3. Mogućnost pregleda profila uz mogućnost update-a nekih specifičnih podataka	

Tok događaja 3. Uspješan završetak “ Pregled profila donatora ”

Administrator	Sistem
1. Izbor pregleda donorskog profila	
	2. Otvara prozor za pregled profila sa svim relevantnim informacijama
3. Mogućnost pregleda profila uz mogućnost izvršavanja CRUD operacija nad podacima	

*Objektno Orientisana Analiza i Dizajn*

**Scenarij 7: Hub**

Naziv slučaja upotrebe	Hub
Opis slučaja upotrebe	Omogućuje donorima izbor određenih termina u kojima mogu doći i donirati krv.
Vezani zahtjevi	/
Preduslovi	Registracija i logiranje donora krvi
Posljedice – uspješan završetak	Mogućnost izbora određenih termina u kojima mogu doći donori, te donirati krv. Mogućnost odustajanja od termina, te uvid u zahtjeve klinike.
Posljedice – neuspješan završetak	Nemogućnost izbora određenih termina u kojima mogu doći donori, te donirati krv. Nemogućnost odustajanja od termina, te uvida u zahtjeve klinike.
Primarni akteri	Donor
Ostali akteri	Zaposlenik, Administrator
Glavni tok	U pitanju je centralno područje ili hub koje će omogućiti donorima da se prijave za neki specifični termin za doniranje krvi. Taj termin će biti sačuvan u aplikaciji, te će biti prikazan i na njegovom personalnom profilu. Bit će u mogućnosti otkazati termin ukoliko bude naknadno nastao problem. Postoji mogućnost provjere zahtjeva klinike kroz Hub.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Prijava termina ”

Donor	Sistem
1. Izbor Hub-a	
	2. Otvara Hub
3. Izbor specifičnih termina	
4. Submit	
	5. Validacija
	6. Prihvatanje odabranog termina, te postavljanje istog kao record na personalni profil

*Objektno Orientisana Analiza i Dizajn*

Tok događaja 2. Neuspješan završetak “ Prijava termina ”

Donor	Sistem
1. Izbor Hub-a	
	2. Otvara Hub
3. Izbor specifičnih termina	
4. Submit	
	5. Validacija
	6. Iстicanje problema

Tok događaja 3. Uspješan završetak “ Odustajanje od termina kroz profil”

Donor	Sistem
1. Otvaranje donorskog profila	
	2. Otvara profil
3. Izbor odustajanja od termina	
	4. Validacija
	5. Prihvatanje odustajanja od termina

Tok događaja 4. Nespešan završetak “ Odustajanje od termina kroz profil ”

Donor	Sistem
1. Otvaranje donorskog profila	
	2. Otvara profil
3. Izbor odustajanja od termina	
	4. Validacija
	5. Odbijanje odustajnja od termina



*Objektno Orientisana Analiza i Dizajn*

Tok događaja 5. Uspješan završetak “ Odustajanje od termina kroz Hub”

Donor	Sistem
1. Otvaranje Hub-a	
	2. Otvara Hub
3. Izbor odustajanja od termina	
	4. Validacija
	5. Prihavatanje odustajanja od termina

Tok događaja 6. Uspješan završetak “ Pristup zahtjevima klinike ”

Zaposlenik	Sistem
1. Otvaranje Hub-a	
	2. Otvara Hub
3. Izbor zahtjeva klinike	
	4. Otvaranje svih zahtjeva
5. Mogućnost pregleda datih zahtjeva, te odgovora	

Tok događaja 7. Uspješan završetak “ Pristup zahtjevima klinike ”

Administrator	Sistem
1. Otvaranje Hub-a	
	2. Otvara Hub
3. Izbor zahtjeva klinike	
	4. Otvaranje svih zahtjeva
5. Mogućnost pregleda datih zahtjeva, te odgovora	

*Objektno Orientisana Analiza i Dizajn*

**Scenarij 8: Stanje Zalihe**

Naziv slučaja upotrebe	Stanje Zalihe
Opis slučaja upotrebe	Omogućuje korištenje CRUD operacija nad stanju zaliha kroz sistem
Vezani zahtjevi	/
Preduslovi	Logiranje
Posljedice – uspješan završetak	Mogućnost izbora određenih termina u kojima mogu doći donori, te donirati krv
Posljedice – neuspješan završetak	/
Primarni akteri	Donor
Ostali akteri	/
Glavni tok	Kada se uloguje admin ili neki zaposlenik, kroz hub će imati opciju provjere stanja zalihe krvi. Također, kada donor donira krv u određenom terminu, zaposlenik će dodati određenu količinu novo-donirane krvi ili će oduzeti od originalnog stanja određenu količinu koju prevoze u neku kliniku u saradnji sa bankom.
Alternative/proširenja	/

Tok događaja 1. Uspješan završetak “ Provjera stanja zalihe ”

Zaposlenik	Sistem
1. Izbor Hub-a	
	2. Otvara Hub
3. Izbor prozora za stanje zalihe	
	4. Otvara stanje zalihe
5. Pregled trenutnog stanja, uz moguću izmjenu	

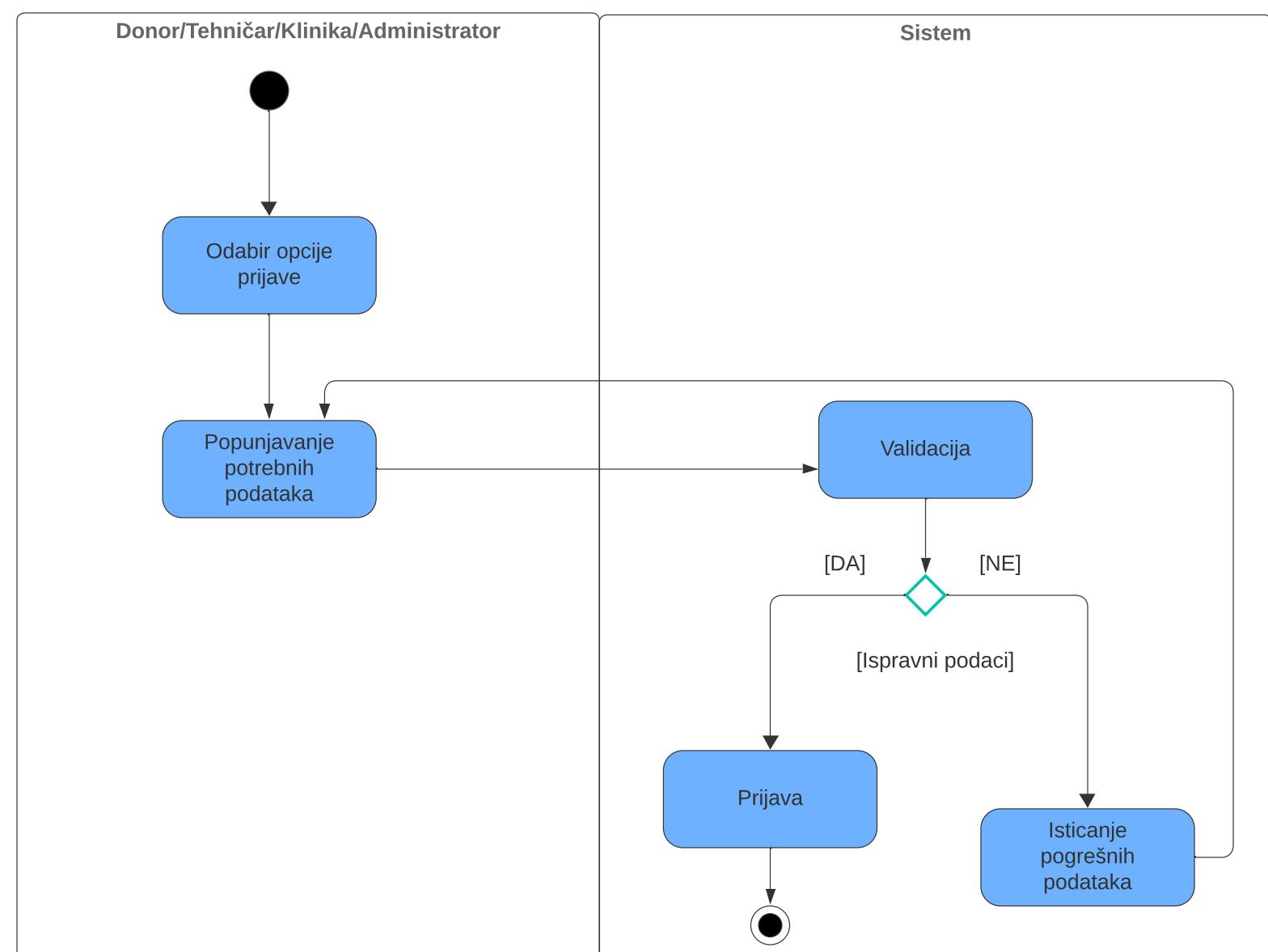
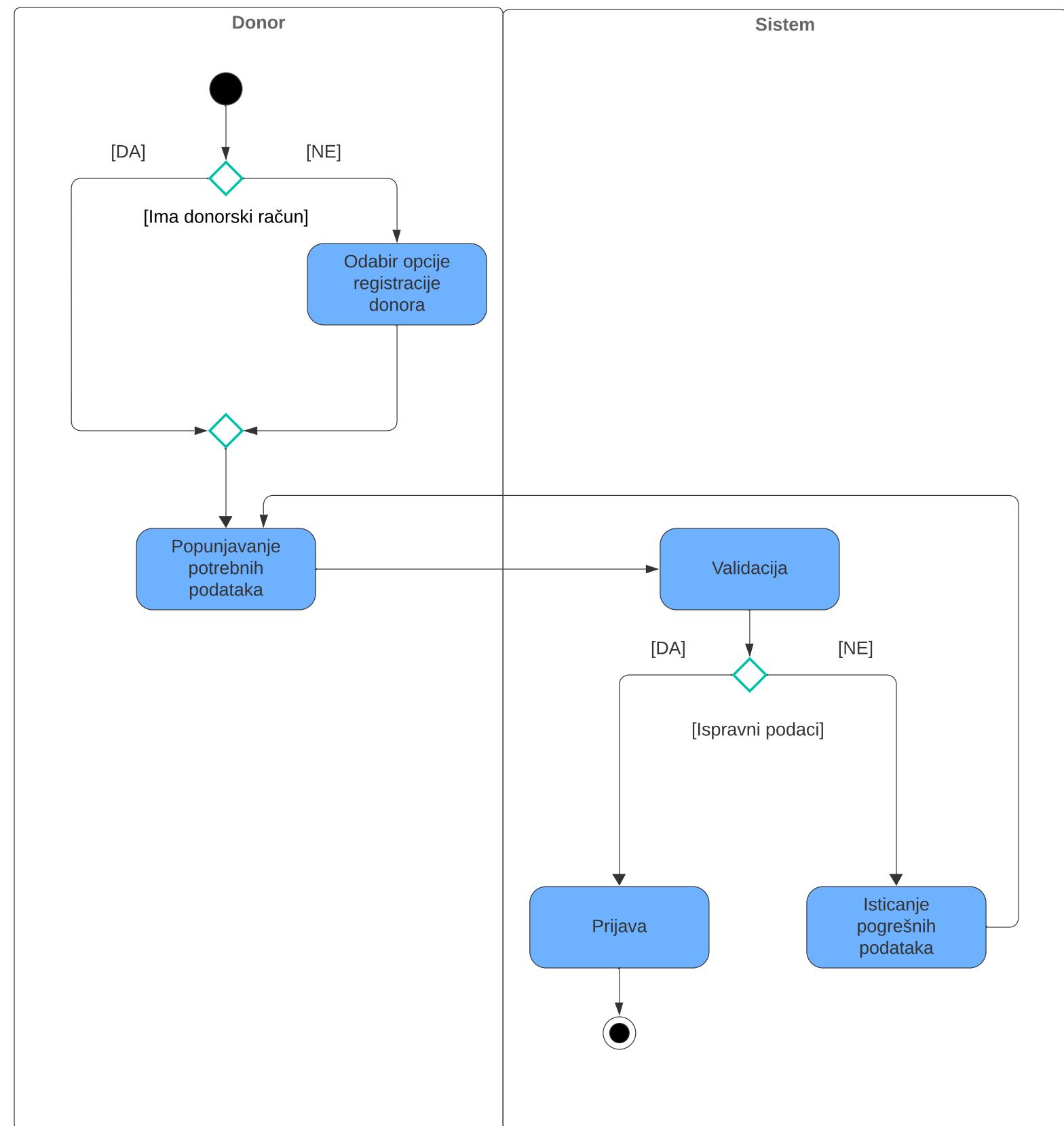
*Objektno Orijentisana Analiza i Dizajn*

Tok događaja 2. Uspješan završetak “ Provjera stanja zalihe ”

Administrator	Sistem
1. Izbor Hub-a	
	2. Otvara Hub
3. Izbor prozora za stanje zalihe	
	4. Otvara stanje zalihe
5. Pregled trenutnog stanja, uz moguću izmjenu	

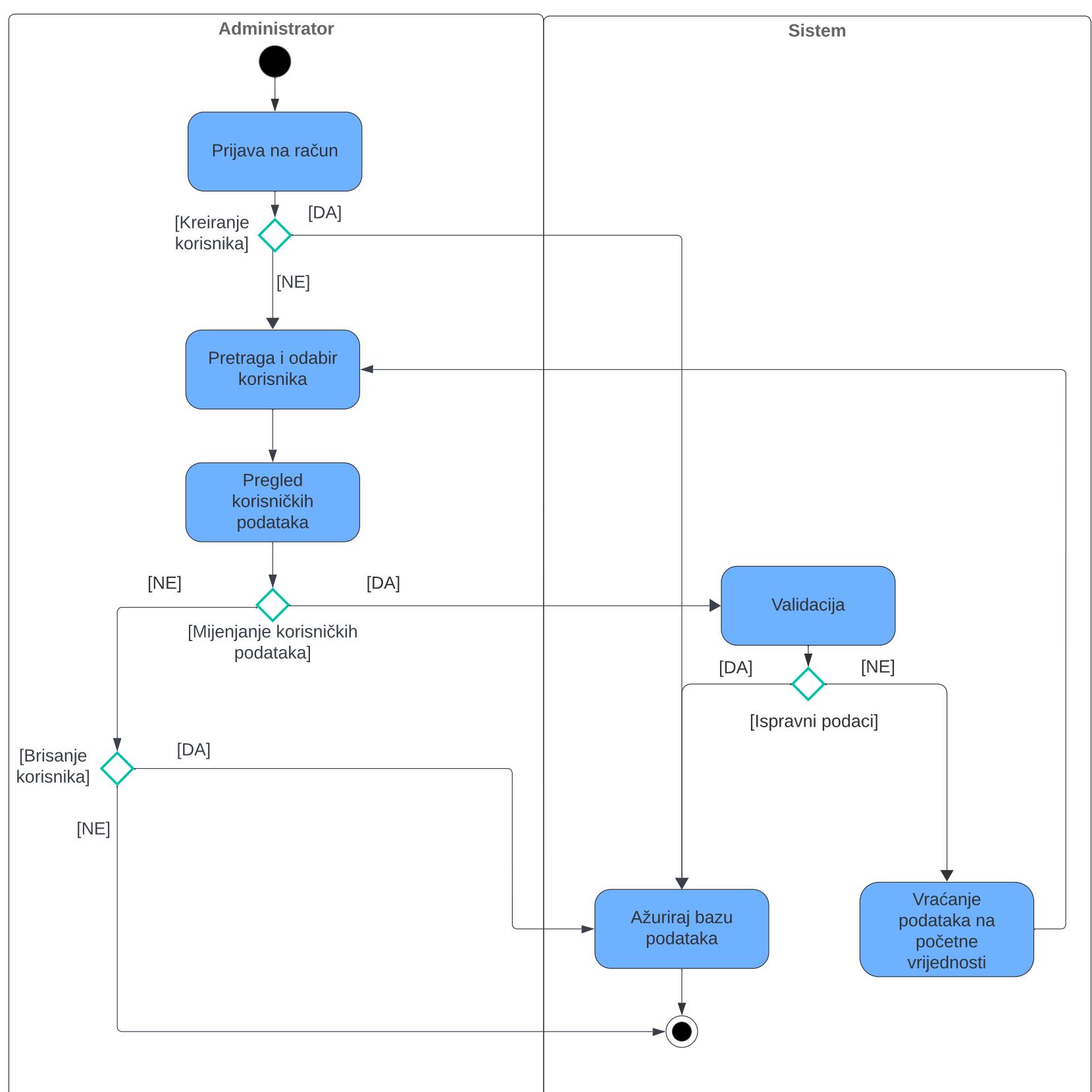
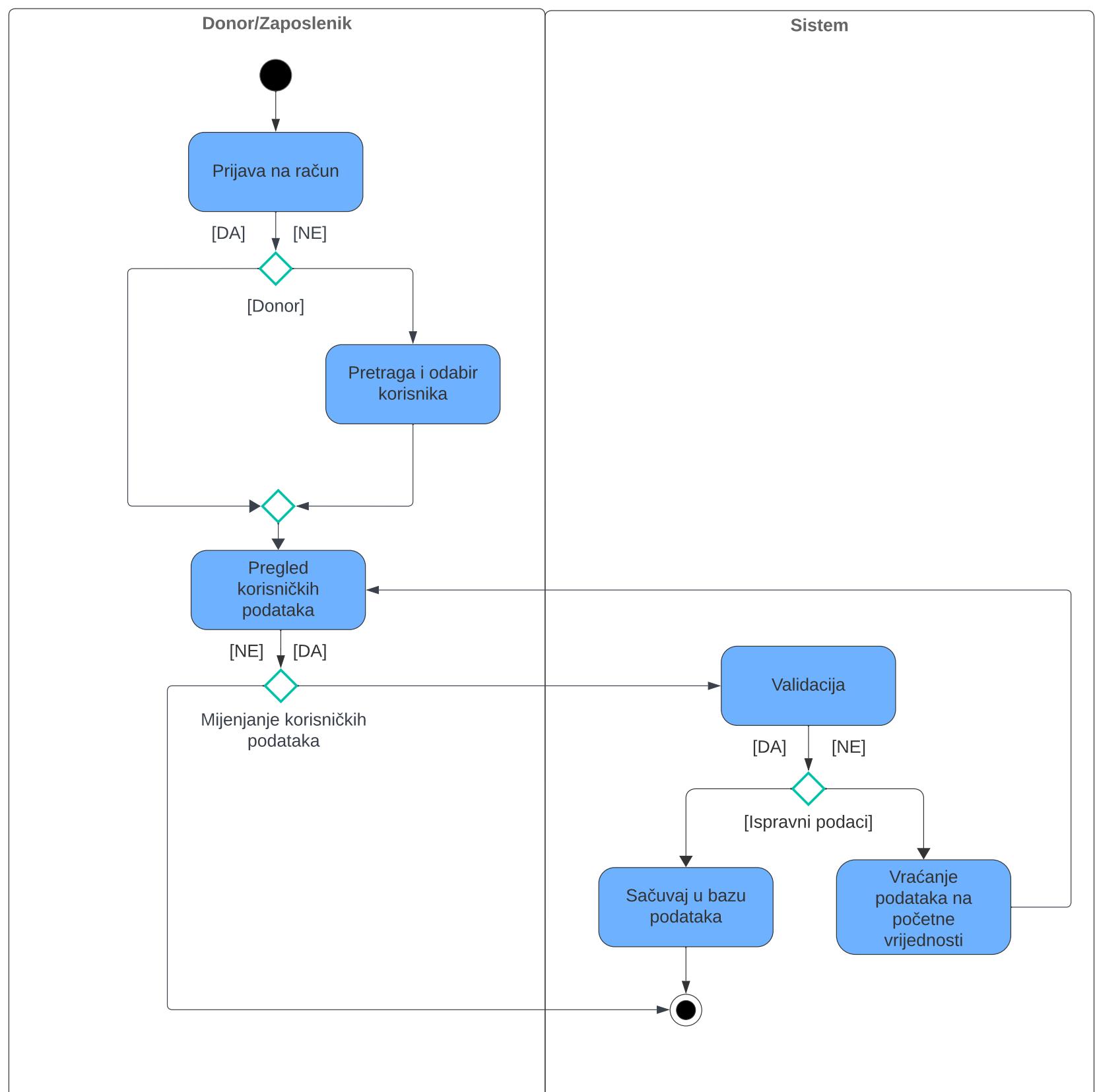
#### Activity diagram shapes

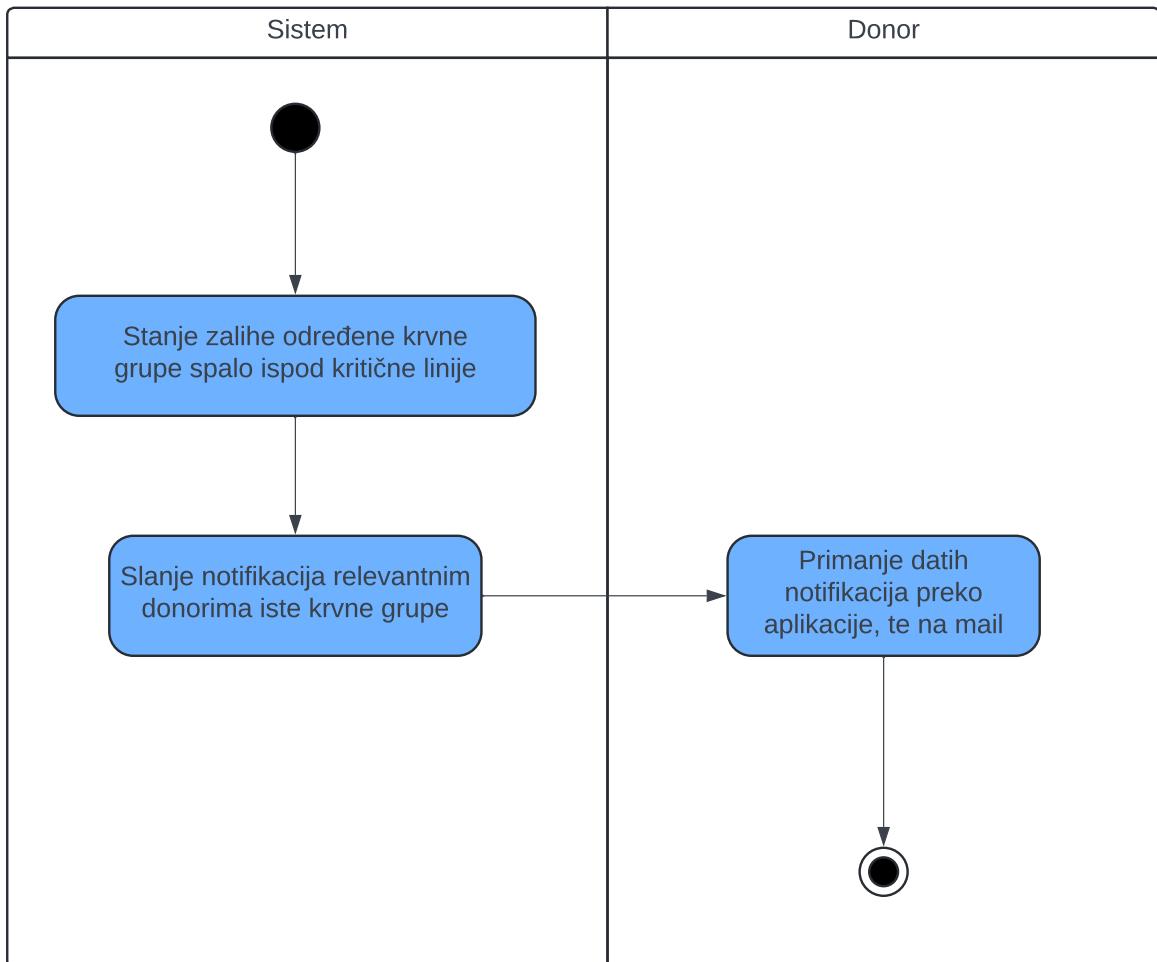
- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End



### Activity diagram shapes

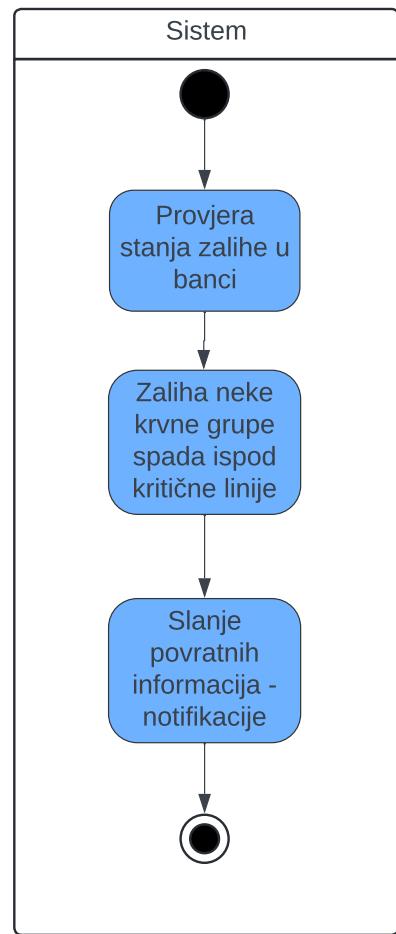
- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End





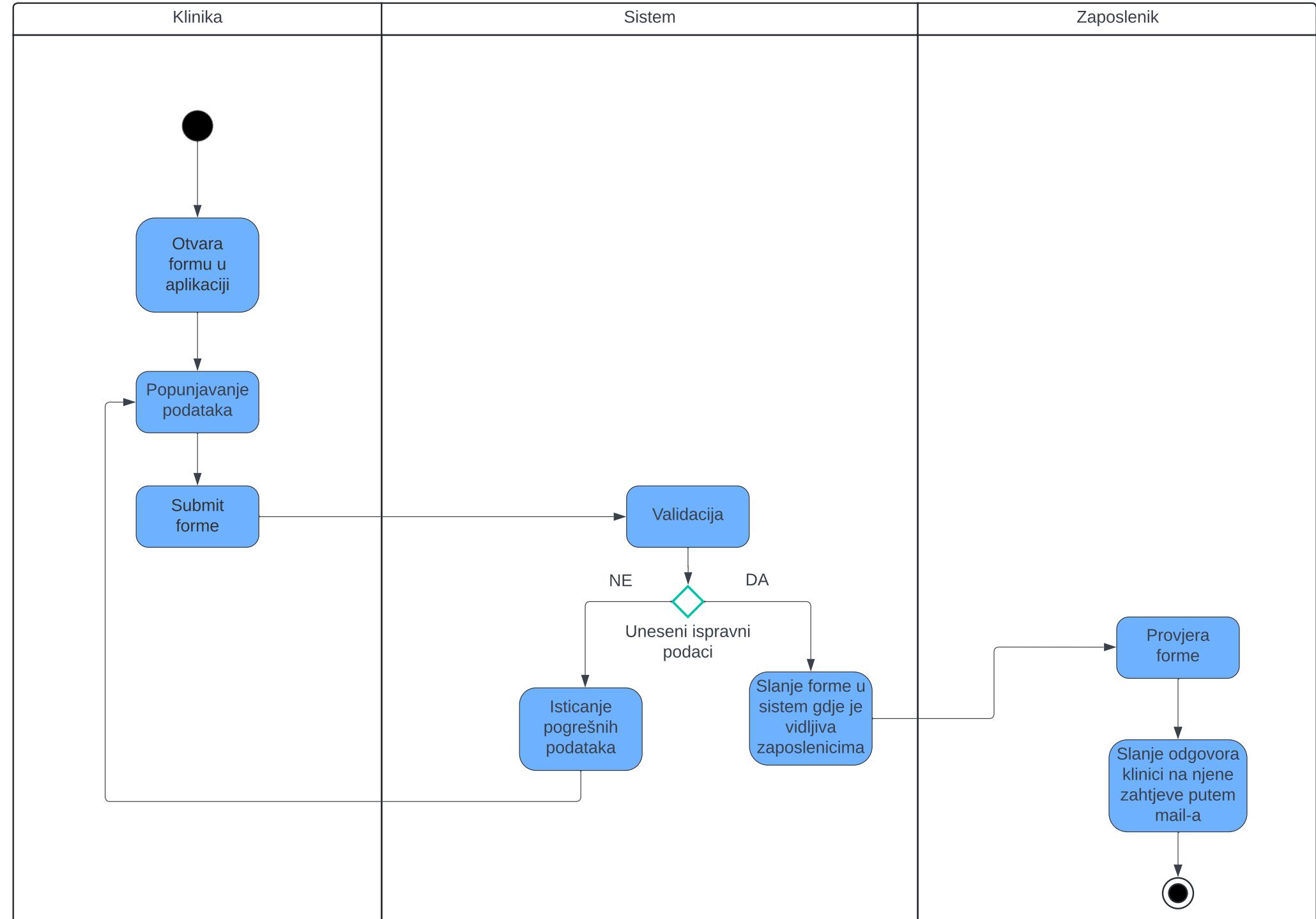
### Activity diagram shapes

- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End



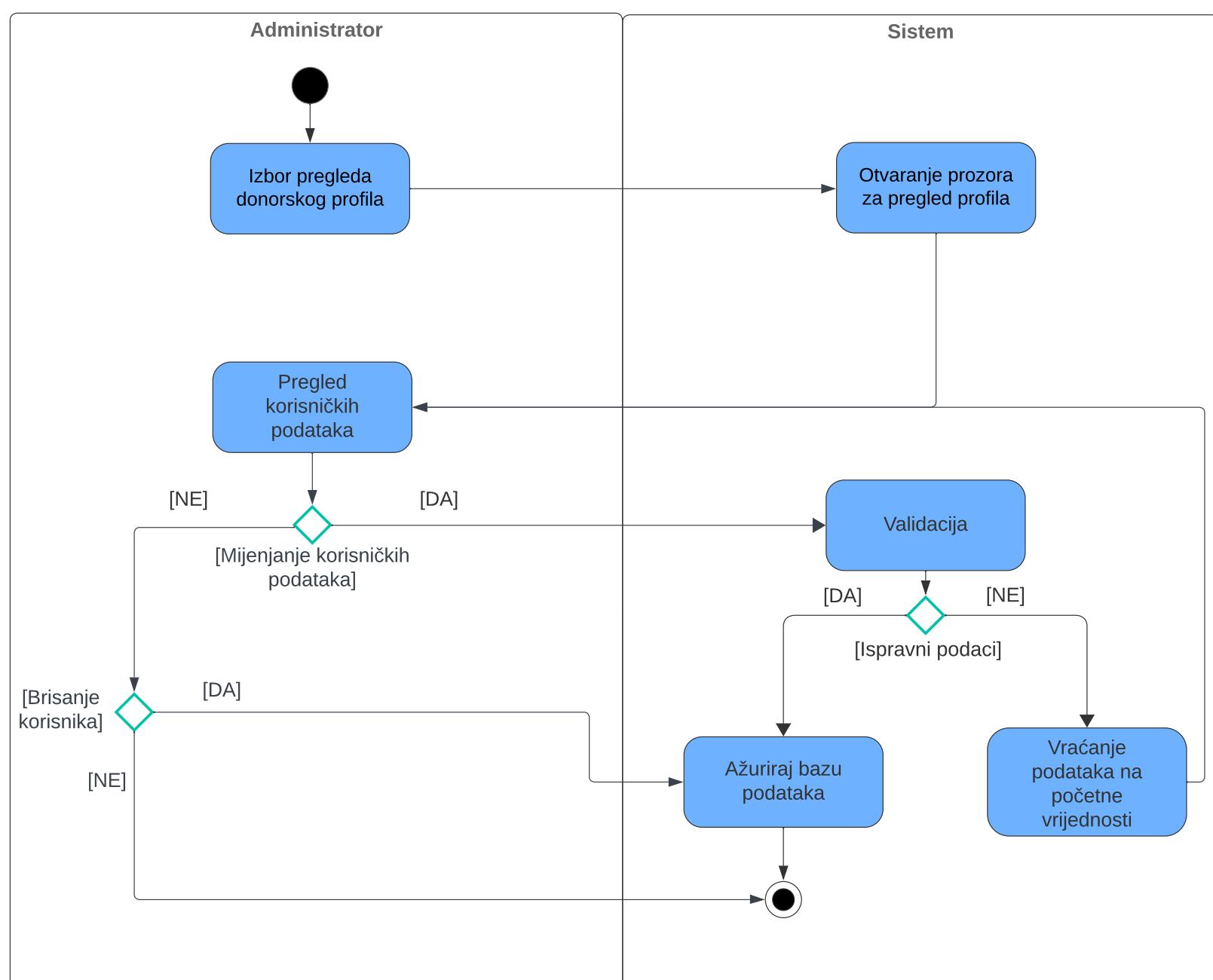
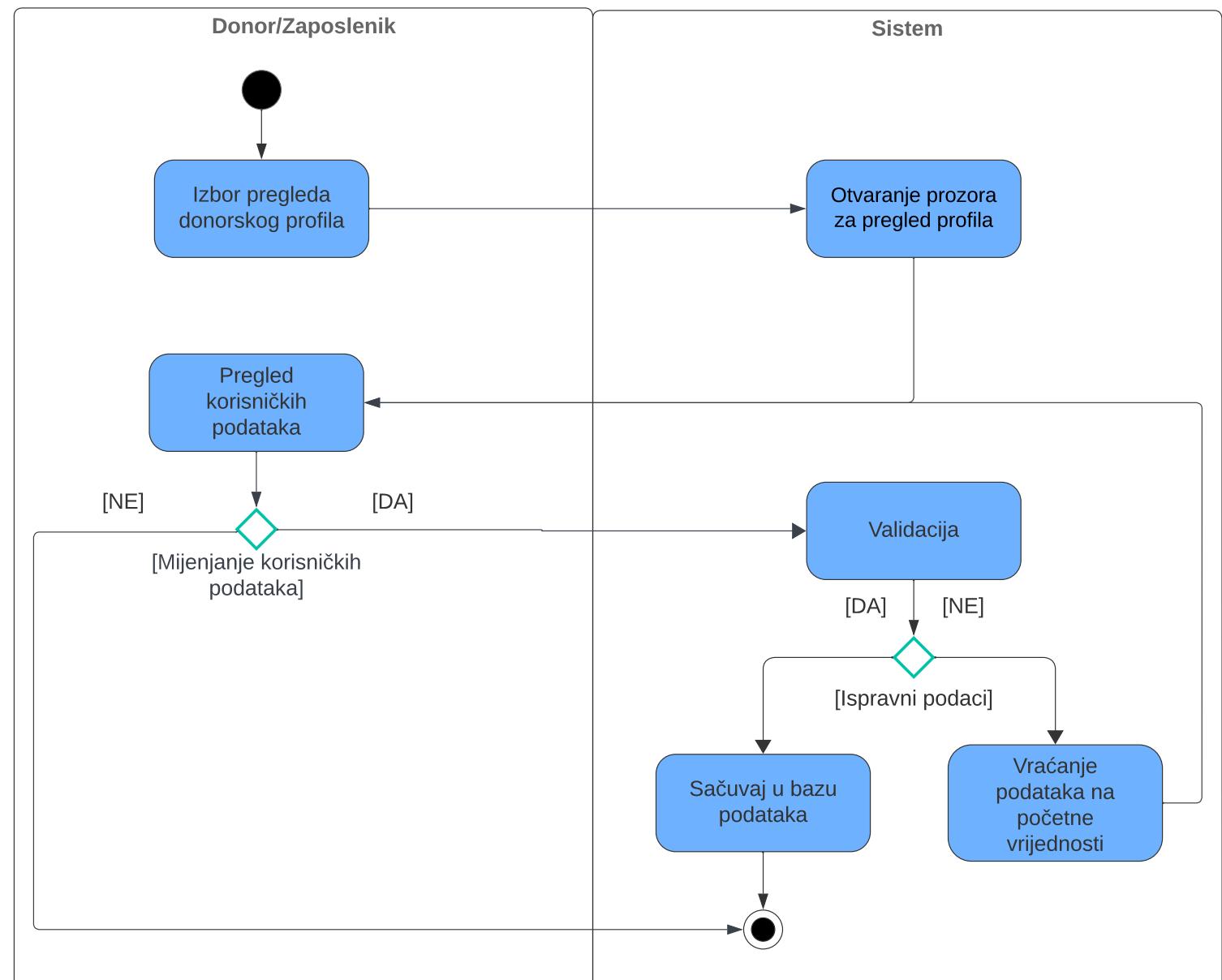
**Activity diagram shapes**

- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End



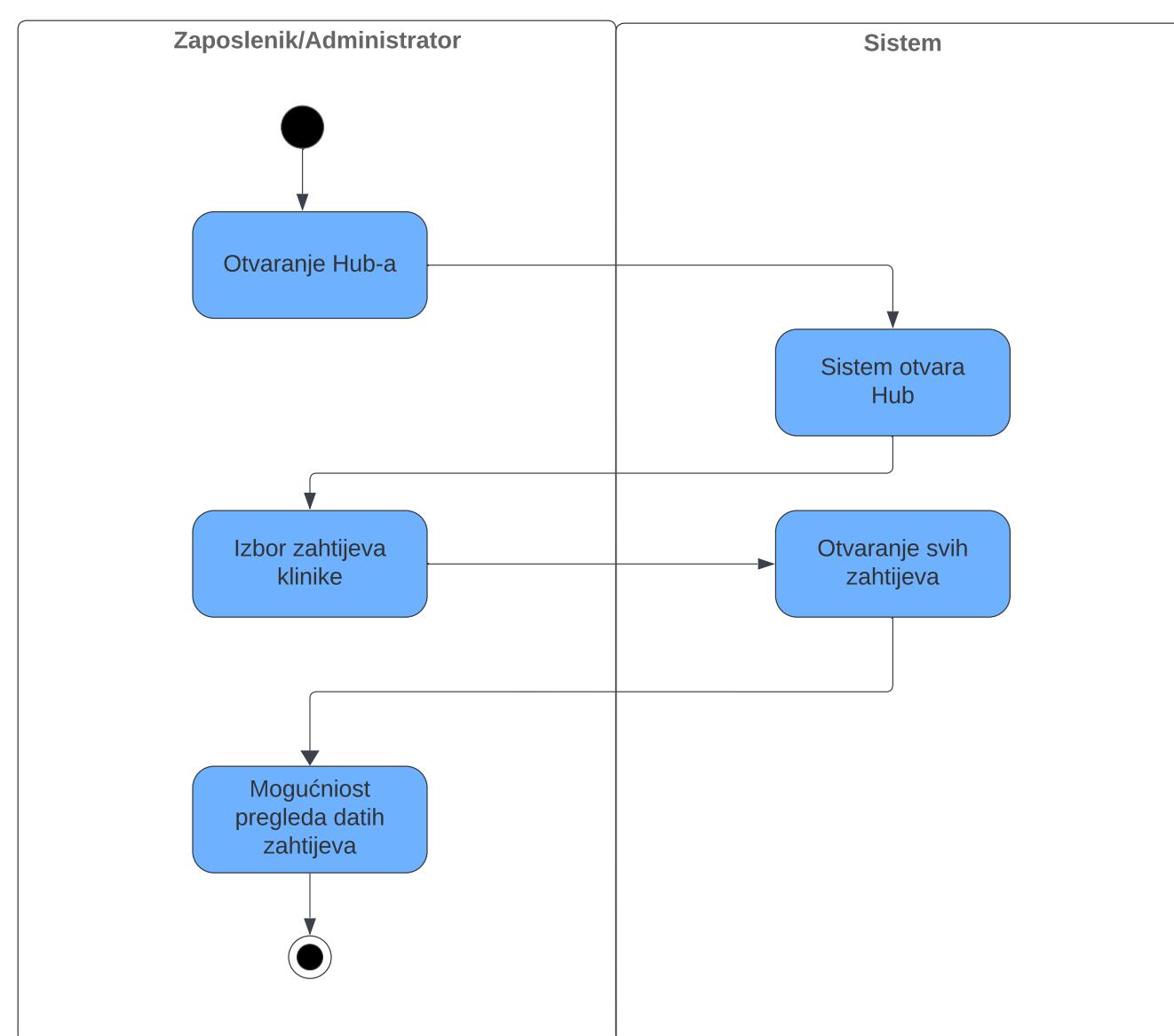
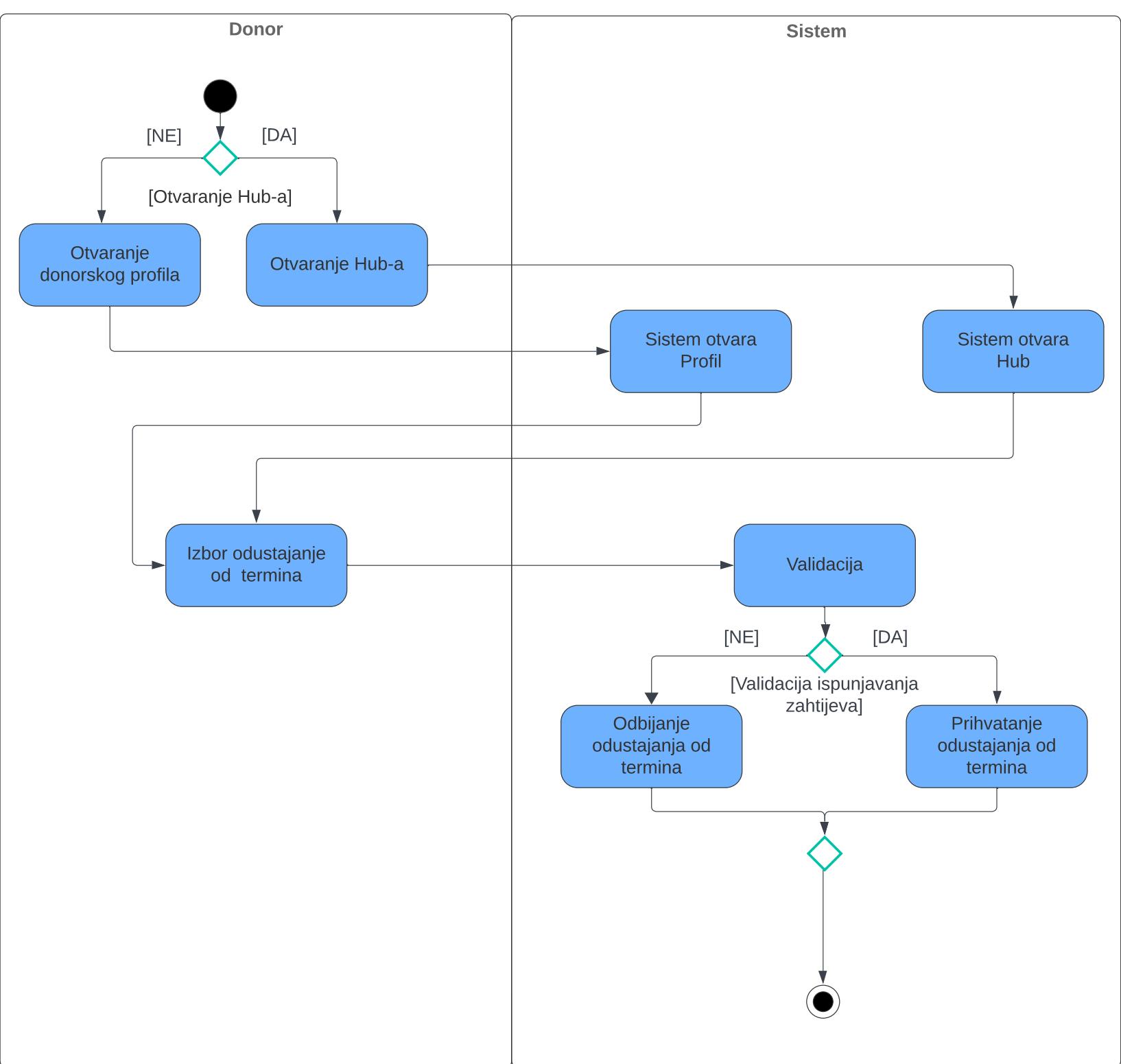
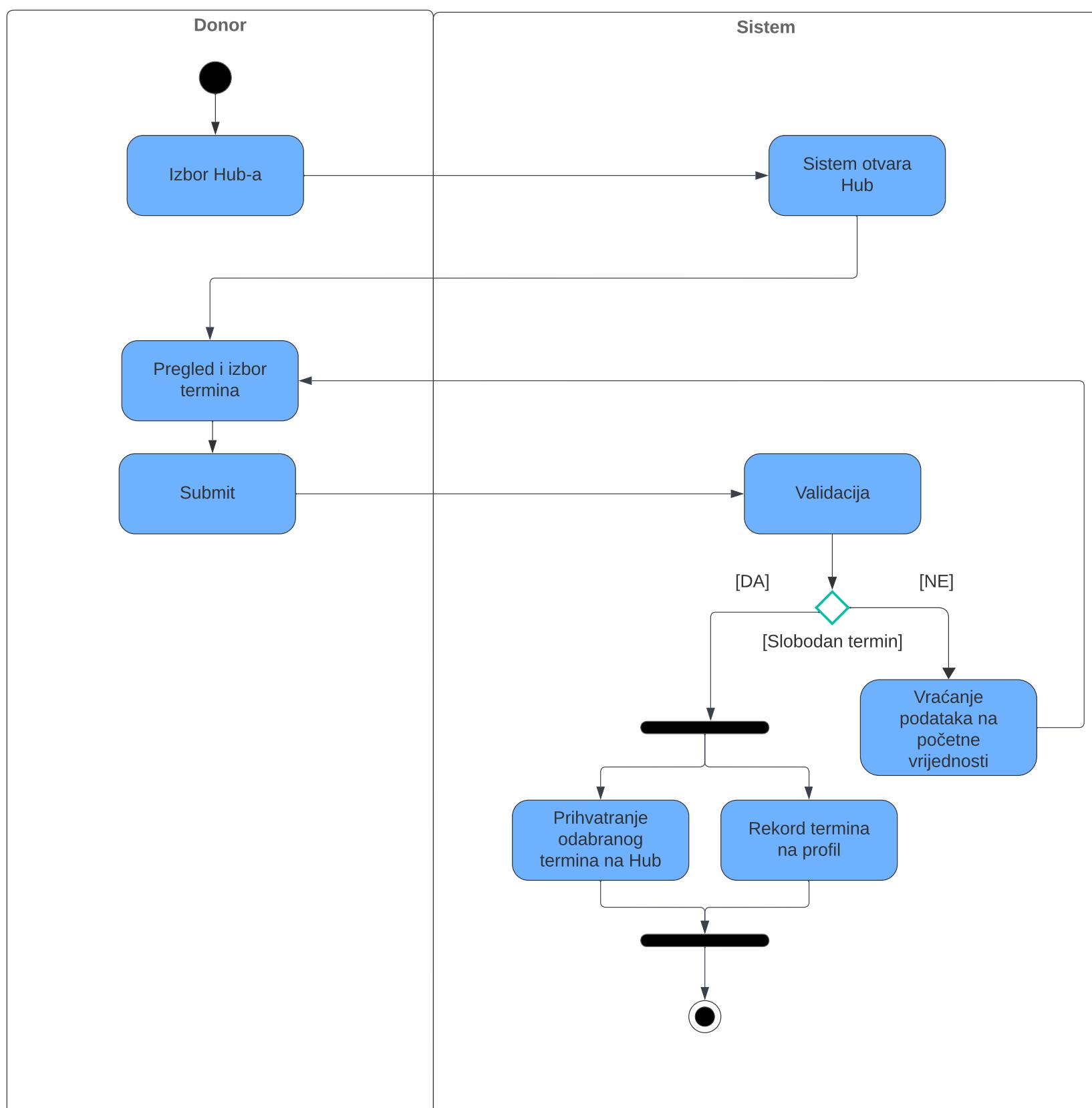
### Activity diagram shapes

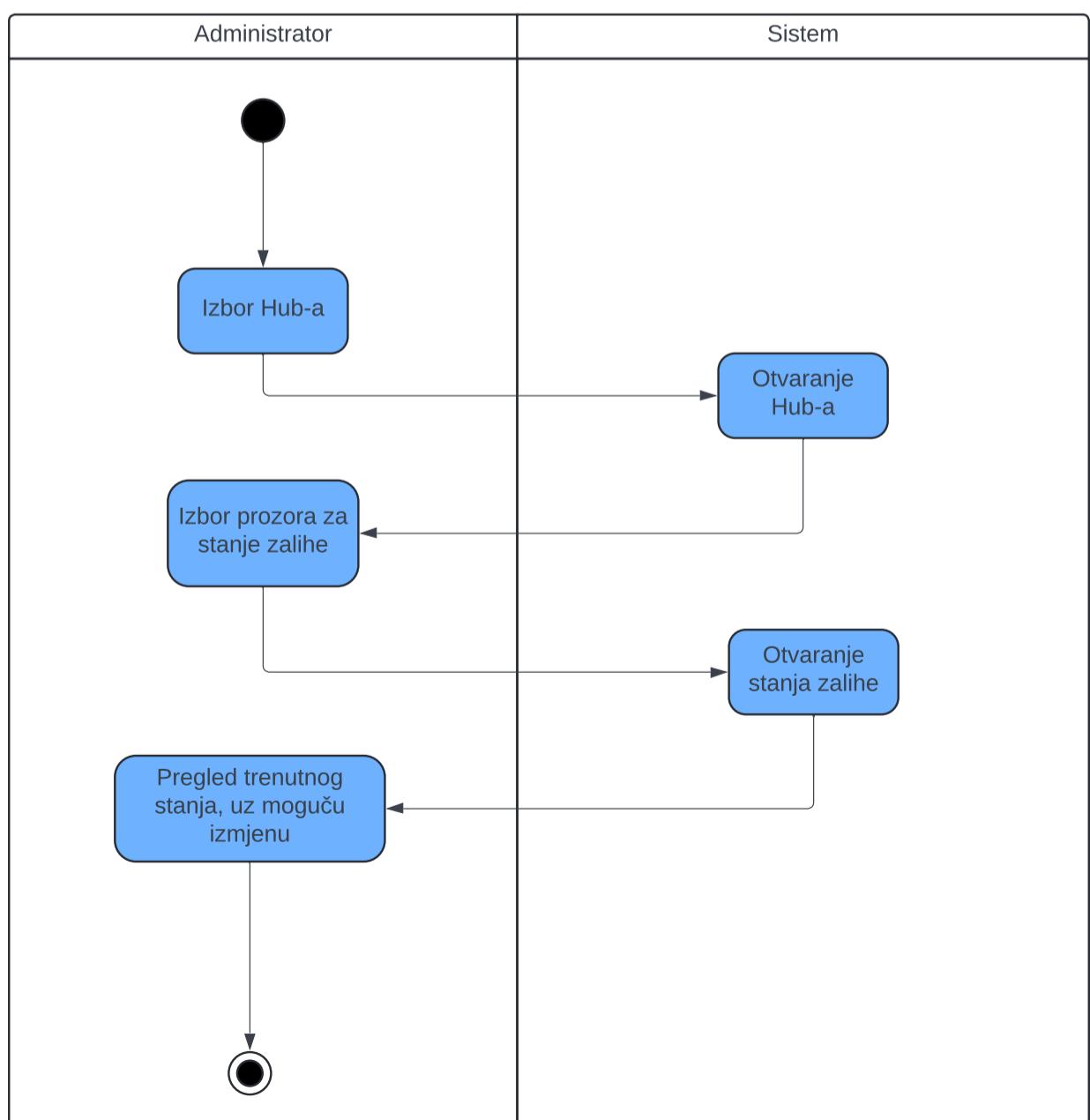
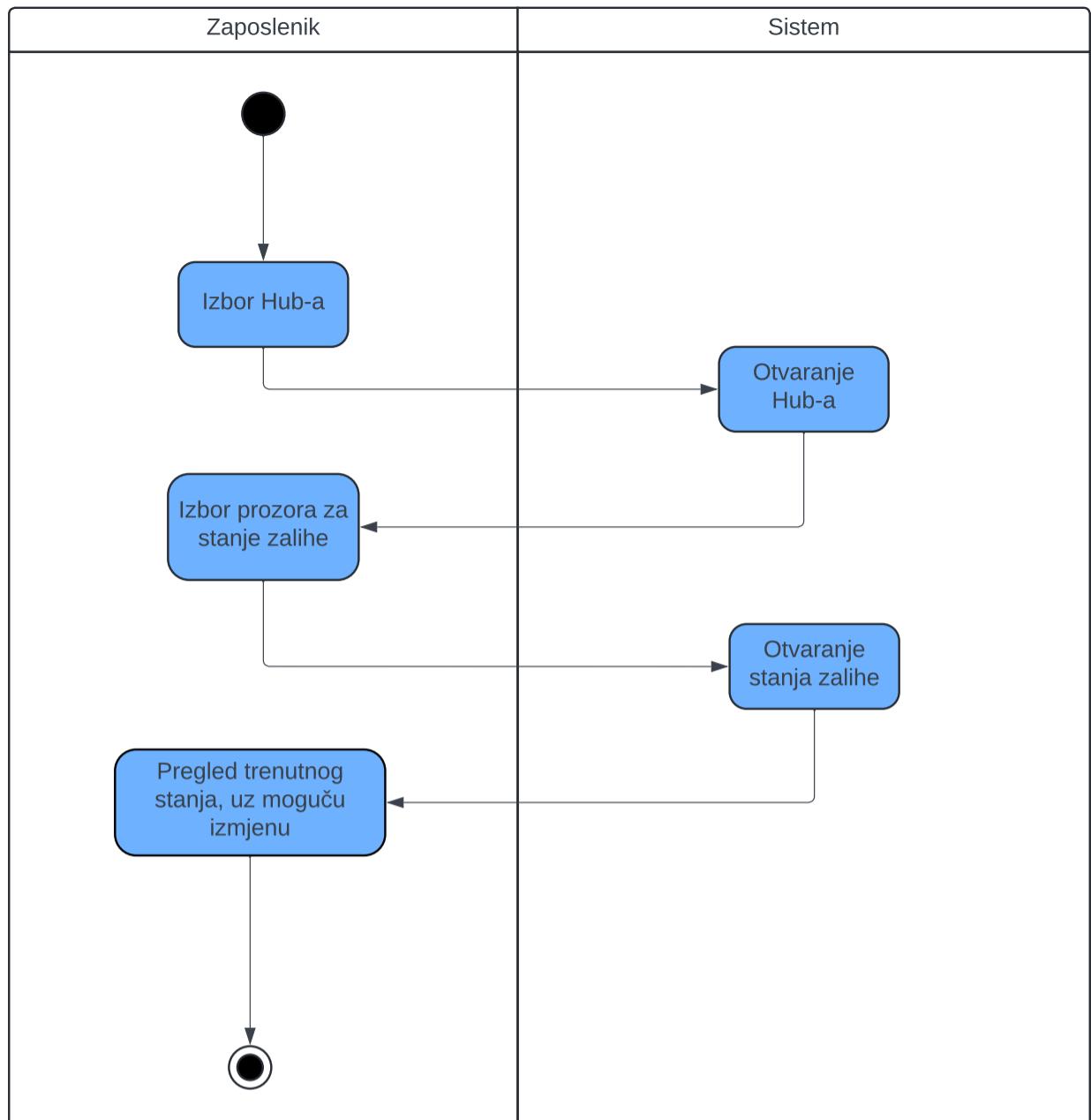
- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End



**Activity diagram shapes**

- Start
- Horizontal Fork/Join
- ◇ Branch/Merge
- End





https://VitalFlow/login

The login page features a background image of red blood cells and a network of veins against a dark blue and purple gradient. At the top left is a small red flame icon. The top navigation bar includes links for Početna, O nama, Kontakt, Registruj Se, and Login, with Login being the active tab. A search icon is at the top right.

**Login**

Email

Password

Zapamti me

Zaboravljena lozinka?

Prijavi se

Nemate racun? Registruj se

Za klinike: Pristupi formi za zatraživanje krvi

https://VitalFlow/zahtjevi\_klinike

Klinički zahtjev krvi

Email

Password

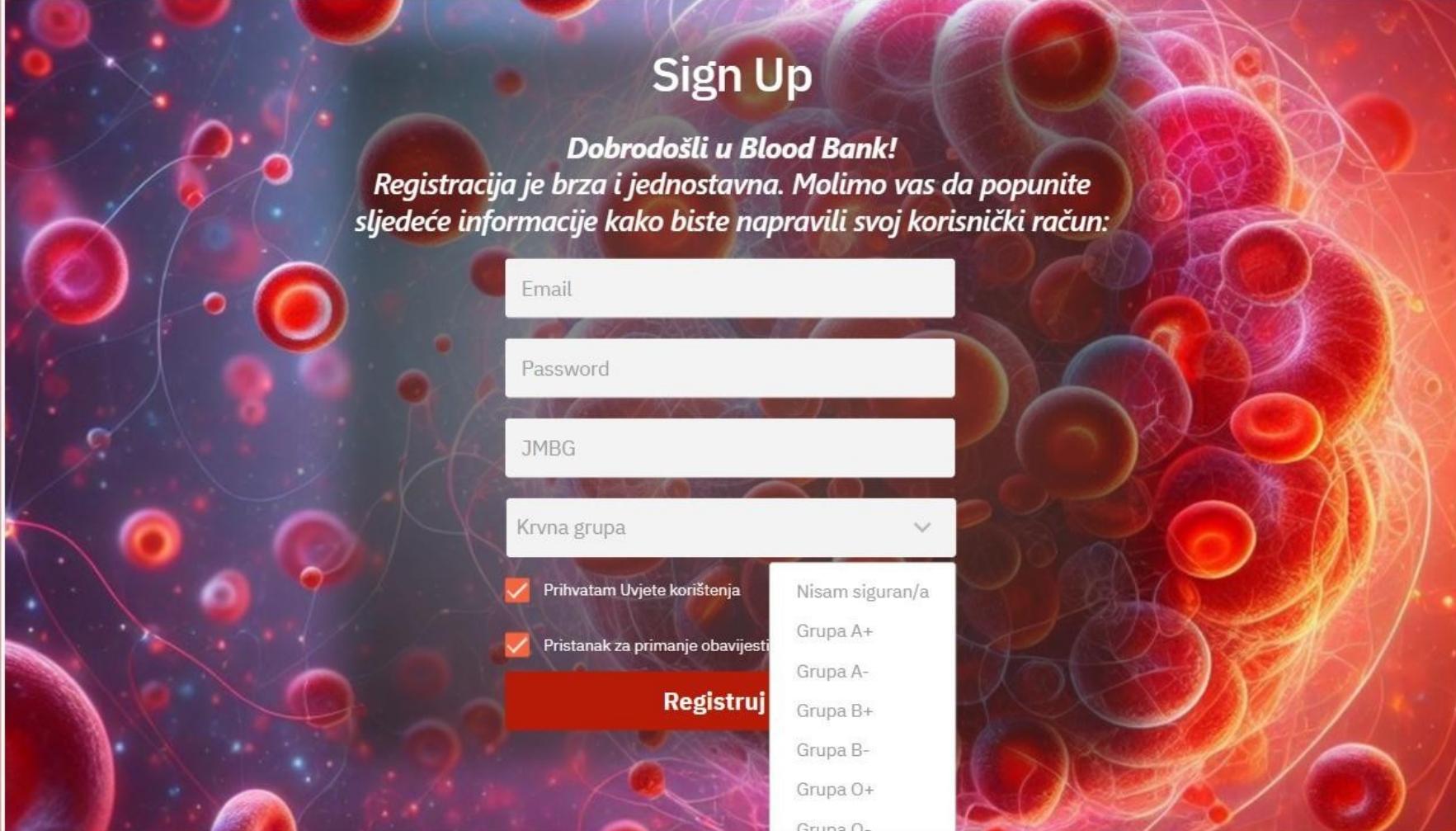
Količina

Krvna grupa

Specijalne napomene

Potvrди zahtjev

https://VitalFlow/signup



Početna O nama Kontakt Registruj Se Login Q

# Sign Up

**Dobrodošli u Blood Bank!**

*Registracija je brza i jednostavna. Molimo vas da popunite sljedeće informacije kako biste napravili svoj korisnički račun:*

Email

Password

JMBG

Krvna grupa

Prihvatom Uvjete korištenja

Pristanak za primanje obavijesti

**Registruj**

Nisam siguran/a  
Grupa A+  
Grupa A-  
Grupa B+  
Grupa B-  
Grupa O+  
Grupa O-



Početna

O nama

Kontakt

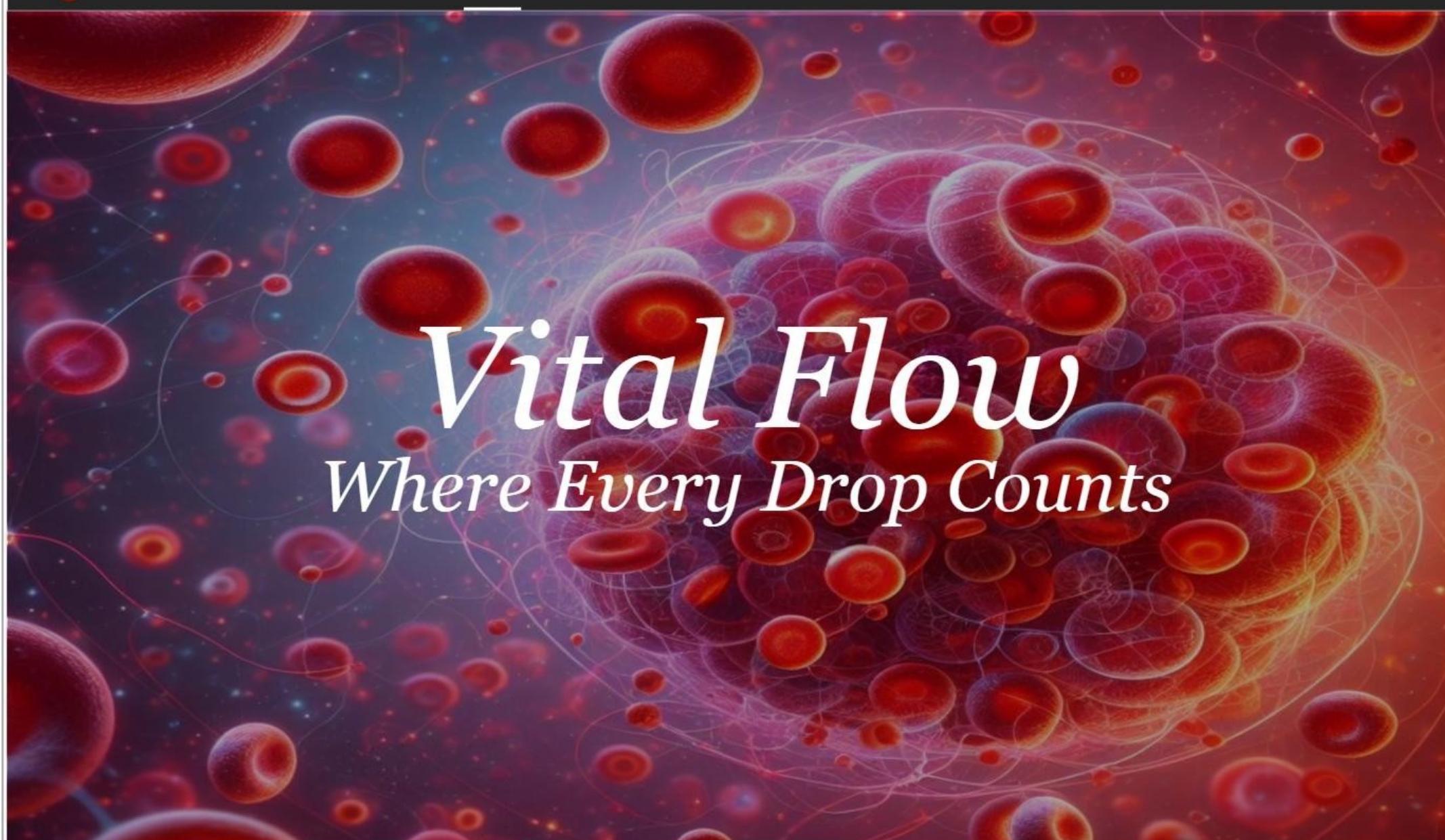
Hub

Profil



# Vital Flow

## *Where Every Drop Counts*



https://VitalFlow/Početna

**Vital Flow**

*Where Every Drop Counts*

*Uredite profil računa*

*Prijavite termin za doniranje*

Početna O nama Kontakt Hub Profil

Search icon

Red flame logo

Background image: A dense cluster of red blood cells against a dark blue background, with a network of glowing white lines forming a complex web across the scene.



https://VitalFlow/onama

**Vital Flow**

Dobrodošli u VitalFlow - vašu sigurnu luku za zdravlje i humanost. Mi smo tim posvećen pružanju inovativnih rješenja u svijetu krvnih banaka, omogućujući brzu i učinkovitu razmjenu životno važnih resursa. Sa strašću za zdravlje i brigom za zajednicu, naša misija je osigurati da svaki kapljica krvi ima moć promijeniti nečiji život nabolje. VitalFlow je više od aplikacije - to je zajednica koja povezuje ljudе i spašava živote. Kroz našu platformu, osnažujemo krvne banke i donatore da rade zajedno u stvaranju pozitivne promjene. Naša vizija je svijet u kojem svaka osoba ima pristup sigurnoj i kvalitetnoj krvi kad im je najpotrebnija. Mi smo VitalFlow - inovatori u području medicinskog humanizma. Naša platforma spaja tehnologiju i humanost kako bi olakšala pristup krvi u hitnim situacijama. S ponosom podržavamo zdravlje zajednice i aktivno radimo na stvaranju boljeg sutra za sve nas. VitalFlow je rezultat strasti i predanosti prema spašavanju života. U našoj priči, svaki donor, svaka transfuzija, svaki korak naprijed je priča o nadi i humanosti.



Početna

O nama

Kontakt

Hub

Profil



## Naš Tim



Benjamin Bandić

bbandic1@etf.unsa.ba

Tekst



Muhamed Husić

mhusic1@etf.unsa.ba

Tekst



Amer Mujalo

amujalo1@etf.unsa.ba

Tekst



Bakir Šatara

bsatara1@etf.unsa.ba

Tekst



Početna

O nama

Kontakt

Hub

Profil



## Kontakt



VitalFlow

VitalFlow@thecompany.com  
+387 60 000 000

Adresa:

Pošalji E-mail

Ime i Prezime

E-Mail

Vaša poruka...

Pošalji



Početna

O nama

Kontakt

Hub

Profil



Prijavljeni termini:

Prijava termin:

Ime i Prezime

Krvna Grupa

JMBG

Sala

Sat 4.May.2024

**Prijavi Termin**

Zahtjevi Klinike

Stanje Zalihe



Početna

O nama

Kontakt

Hub

Profil



## Zahtjevi Klinika

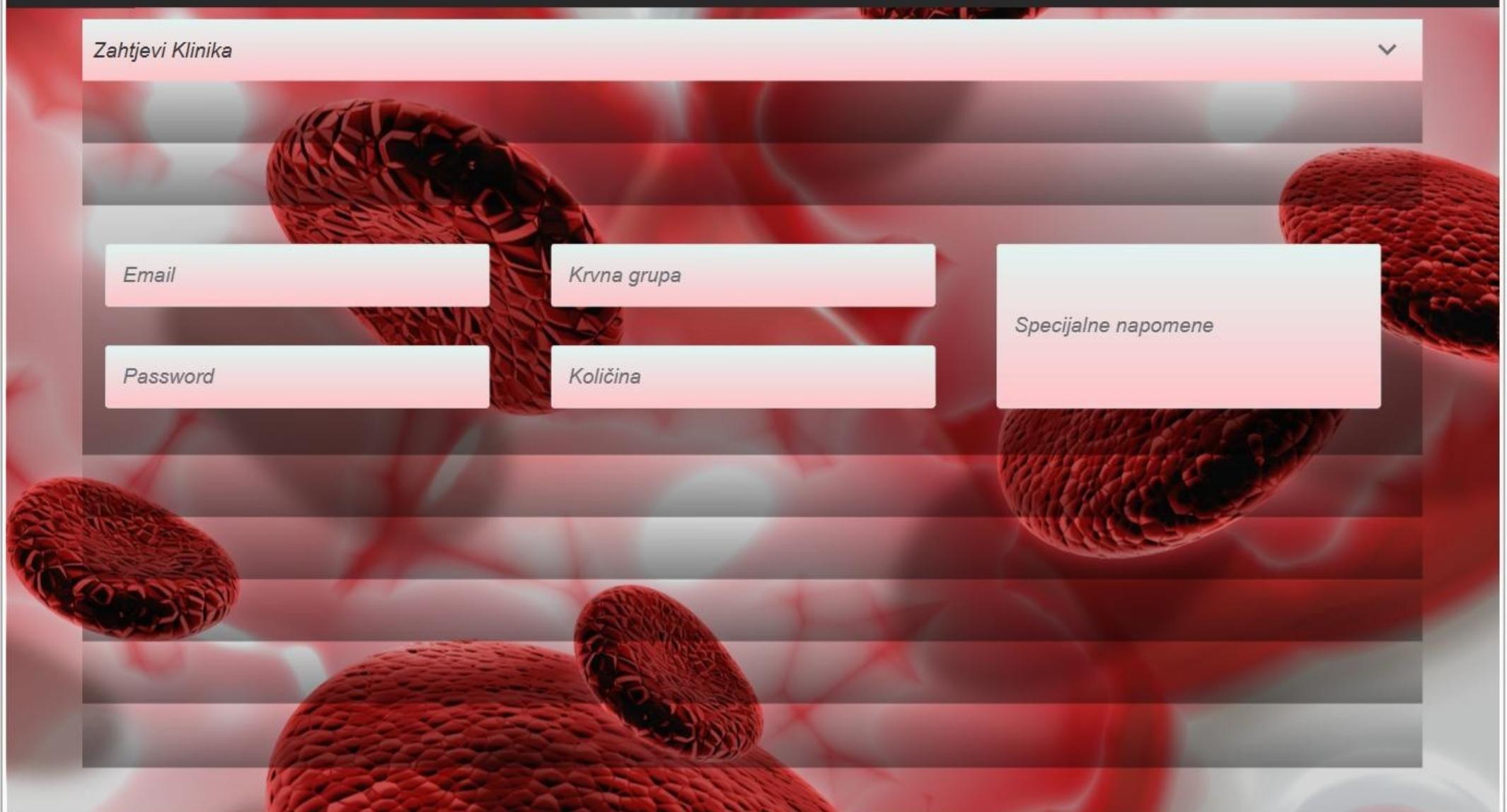
Email

Krvna grupa

Specijalne napomene

Password

Količina



https://VitalFlow/admin/zahtjevi?page=1

Klinički zahtjev krvi

▶ **Klinika X** **Datum: 04.08.2023**

**Stanje Zaliha**

<https://VitalFlow/admin/zahtjevi?page=1>

Klinički zahtjev krvi

**Klinika X** **Datum: 04.08.2023**

Email: klinikaX@KLINIKA.com Krvna grupa: -- Količina: 4l

Specijalni zahtjevi klinike:

.....  
.....  
.....  
.....

**Klinika X** **Datum: 04.08.2023**

**Klinika X** **Datum: 04.08.2023**

**Klinika X** **Datum: 04.08.2023**

**Stanje Zaliha**

https://VitalFlow/hub?sta=donor/hub\_termini

Prijava Termina

<input checked="" type="checkbox"/> <b>Prijavljeni</b>	<b>Sala A</b>	<b>Datum: 04.08.2023 12h</b>
<input type="checkbox"/>	<b>Sala A</b>	<b>Datum: 04.08.2023 12h</b>
<input type="checkbox"/>	<b>Sala A</b>	<b>Datum: 04.08.2023 12h</b>
<input type="checkbox"/>	<b>Sala A</b>	<b>Datum: 04.08.2023 12h</b>
<input type="checkbox"/>	<b>Sala A</b>	<b>Datum: 04.08.2023 12h</b>

**Stanje Zaliha**

https://VitalFlow/hub?sta=admin/pretraga\_donora

The screenshot shows a web browser window with the URL https://VitalFlow/hub?sta=admin/pretraga\_donora. The page has a dark theme with a red header and footer featuring a blood vessel and red blood cells. The header includes a logo, navigation links (Početna, O nama, Kontakt, Hub, Profil), a search bar, and a user icon.

**Pretraga Donora**

**Donor: Ime Prezime**

**Datum prijave: 04.08.2023**

Email: imePrezimeX@VitalFlow.com      Krvna grupa: --      JMBG:

Broj donacija: X      Broj Telefona: xxx/yyy-zzz      Datum Rođenja: XXXXX

Broj Otkazivanja: X      Prijavljen na termin: -----      Napomene:

Ažuriraj

▶ **Donor: Ime Prezime**      **Datum prijave: 04.08.2023**

▶ **Donor: Ime Prezime**      **Datum prijave: 04.08.2023**

▶ **Donor: Ime Prezime**      **Datum prijave: 04.08.2023**

Stanje Zaliha

The main content area displays three donor entries, each with a disclosure arrow icon and a bolded "Donor: Ime Prezime" label. To the right of each entry is a bolded "Datum prijave: 04.08.2023". Below the first two entries are "Email" and "Krvna grupa" fields, and below the third is a "JMBG" field. Further down are "Broj donacija" and "Broj Telefona" fields, followed by "Datum Rođenja" and "Napomene" fields. A large orange "Ažuriraj" button is positioned between the middle and bottom sections. The bottom section contains three more entries with the same structure. A red "Stanje Zaliha" button is located at the very bottom right.



Početna

O nama

Kontakt

Hub

Profil



*Grupa A+*

+

*Količina:*

-

*Kapacitet:*

+

*Količina:*

-

*Kapacitet:*

*Grupa B+*

+

*Količina:*

-

*Kapacitet:*

+

*Količina:*

-

*Kapacitet:*

*Grupa AB+*

+

*Količina:*

-

*Kapacitet:*

+

*Količina:*

-

*Kapacitet:*

*Grupa O+*

+

*Količina:*

-

*Kapacitet:*

+

*Količina:*

-

*Kapacitet:*

*Grupa A-*

*Grupa B-*

*Grupa AB-*

*Grupa O-*



Početna

O nama

Kontakt

Hub

Profil



# Ime i Prezime

Datum registracije:

01.01.2024

*Prijavljeni termini:*

## Email

nešto@vitalflow.com

## JMBG

0402000160003

## Krvna Grupa

A+

## Opis

Tekst:

https://VitalFlow/hub?sta=zaposlenik/profil&id=253



**Ime i Prezime**

Datum registracije:  
01.01.2024

**Email:**  
email@VitalFlow.com

**JMBG**  
75287658725784

**ID:**  
253

**Broj telefona:**  
218/230-948

Ažuriraj

## Analiza i dizajn sistema

*U nastavku je potrebno definisati sve potencijalne klase koje će se koristiti u sistemu. Za određivanje klasa koje će biti neophodne za rad sistema potrebno je koristiti specifikaciju sistema i prethodno kreirane dijagrame.*

*Template za jednu klasu potrebno je iskopirati onoliko puta koliko je neophodno da bi se definisale sve klase u sistemu.*

### Definicija klasa u sistemu

**Naziv klase:** Korisnik

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

- (FZ br. 01: Registracija i logiranje)
- (FZ br. 02: Spremanje podataka - CRUD)
- (FZ br. 03: Pregled profila)
- (FZ br. 04: Hub)

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
imeIPrezime	String	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration
email	String	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration
password	String	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration
brojTelefona	String	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration
datumRođenja	String	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration
jmbg	Int	<input type="checkbox"/> Atribut je staticki <input type="checkbox"/> Atribut je enumeration

**Naziv klase:** Donor : Korisnik

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
krvnaGrupa	KrvneGrupe	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
brojOtkazivanja	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Zaposlenik : Korisnik

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

(FZ br. 01: Korištenje mail-a za slanje informacija)  
(FZ br. 02: Stanje Zalihe)

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
zaposlenikID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Admin : Korisnik

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

(FZ br. 01: Korištenje mail-a za slanje informacija)  
(FZ br. 02: Stanje Zalihe)

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
adminID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Hub

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
terminID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
klinikaID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Termin

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
jmbg	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
datum	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
sala	Sale	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kapacitet	Int	<input checked="" type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Zahtjev

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
zahtjevID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
email	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kolicina	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
krvnaGrupa	KrvneGrupe	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

opis	String	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
------	--------	---

**Naziv klase:** Sale

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
A1	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
A2	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
A3	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
B1	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
B2	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
B3	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Zaliha

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
krvnaGrupa	KrvneGrupe	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
kriticnaLinija	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kolicina	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** KrvneGrupe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
A+	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
A-	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
B+	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
B-	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
AB+	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
AB-	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
O+	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
O-	Click or tap here to enter text.	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** PregledZahtjevaKrozHub

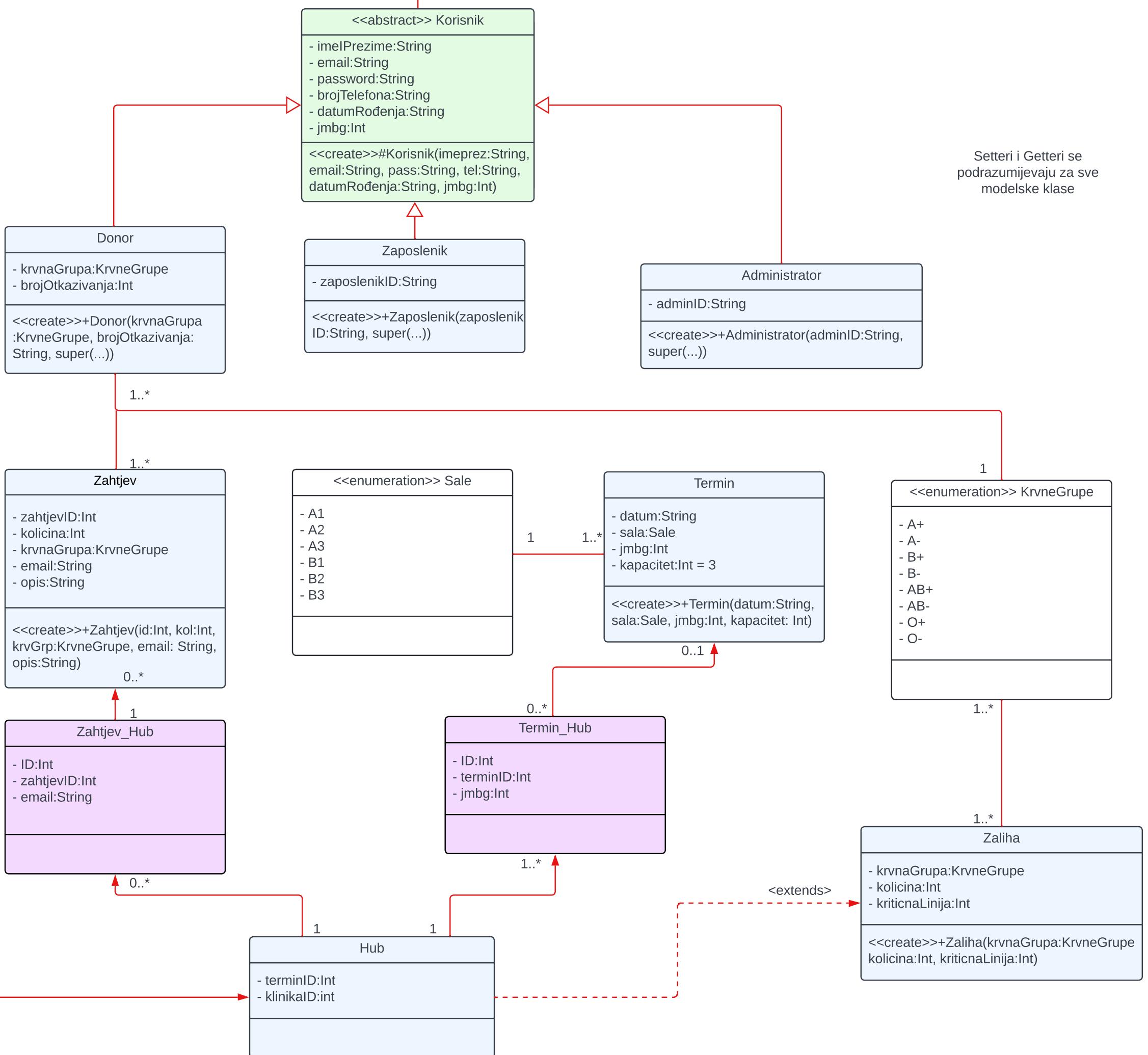
**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
id	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
zahtjevID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
email	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** PregledTerminaKrozHub

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
id	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
terminID	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
jmbg	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>



# SOLID PRINCIPLES

## Single-Responsibility Principle (SRP)

Single-Responsibility Principle (SRP) je prvi SOLID princip koji navodi da bi svaka klasa trebala imati samo jednu odgovornost. To znači da bi svaka klasa trebala biti zadužena za samo jedan dio funkcionalnosti i ta odgovornost treba biti potpuno enkapsulirana u klasi.

Na osnovu našeg dijagrama klasa, SRP princip se primjenjuje na sljedeći način:

- **Apstraktna klasa ‘Korisnik’:** Klasa ‘Korisnik’ čuva osnovne podatke o korisniku, dok su izvedene klase ‘Donor’, ‘Zaposlenik’, i ‘Administrator’ zadužene za čuvanje podataka specifičnih za svaku od ovih uloga.
- **Klase ‘Zahtjev’, ‘Termin’, i ‘Zaliha’:** Svaka od ovih klasa ima tačno jednu odgovornost. Klasa ‘Zahtjev’ je zadužena za upravljanje zahtjevima, klasa ‘Termin’ za upravljanje terminima, a klasa ‘Zaliha’ za upravljanje zalihama.

Svaka klasa u dijagramu ima svoj skup odgovornosti bez preklapanja funkcionalnosti, što ukazuje na to da se Single-Responsibility Principle poštuje u cjelokupnom dijagramu klasa.

## Open-Closed Principle (ORP)

Open-Closed Principle (OCP) je drugi SOLID princip i on navodi da bi “softverski entiteti (klase, module, funkcije, itd.) trebali biti otvoreni za proširenje, ali zatvoreni za modifikaciju”. Ovo znači da bi trebalo biti moguće dodati nove funkcionalnosti bez mijenjanja postojećeg koda.

Na osnovu našeg dijagrama klasa, OCP princip se primjenjuje na sljedeći način:

- **Apstraktna klasa ‘Korisnik’:** Ova klasa je “zatvorena za modifikaciju” jer definira osnovne atribute kao što su ‘korisnickoIme’, ‘email’, i ‘lozinka’ koje su zajedničke za sve korisnike. Međutim, ona je “otvorena za proširenje” jer omogućava izvođenje novih klasa (‘Donor’, ‘Zaposlenik’, ‘Administrator’) koje mogu dodati specifične atribute i metode relevantne za svoje uloge.
- **Klase ‘Zahtjev’, ‘Termin’, i ‘Zaliha’:** Svaka od ovih klasa je zatvorena za modifikaciju jer imaju jasno definisane odgovornosti. Međutim, one su otvorene za proširenje jer se mogu dodati nove metode ili atributi koji proširuju njihove funkcionalnosti bez mijenjanja postojećeg koda.

Dakle, na osnovu dijagrama klasa, vrijedi Open-Closed Principle.

## Liskov Substitution Principle (LSP)

Liskov Substitution Principle (LSP) je treći SOLID princip koji navodi da bi podklase trebale biti zamjenjive za svoje nadklase bez utjecaja na ispravnost programa. To znači da bi svaka klasa koja koristi određenu nadklasu trebala moći koristiti bilo koju od njenih podklasa bez da to utječe na ispravnost programa.

Na osnovu našeg dijagrama klase, LSP princip se primjenjuje na sljedeći način:

- **Apstraktna klasa ‘Korisnik’:** Klasa ‘Korisnik’ je nadklasa za klase ‘Donor’, ‘Zaposlenik’, i ‘Administrator’. Svaka od ovih podklasa može biti zamijenjena klasom ‘Korisnik’ bez utjecaja na ispravnost programa. Na primjer, ako postoji funkcija koja koristi klasu ‘Korisnik’, ta funkcija bi trebala raditi ispravno bez obzira koristi li se instanca klase ‘Donor’, ‘Zaposlenik’, ili ‘Administrator’.

Primjena Liskov-og Substitution Principle-a osigurava da dizajn softvera ostane čist, sa jasno definiranim odnosima između klasa. Zaključujemo da za naš dijagram klasa vrijedi Liskov Substitution Principle.

## Interface Segregation Principle (ISP)

Interface Segregation Principle (ISP) je četvrti SOLID princip koji navodi da klijenti ne bi trebali biti prisiljeni ovisiti o interfejsima koje ne koriste. Ovaj princip promoviše upotrebu više specifičnih interfejsa umjesto jednog općeg interfejsa.

Na onovu našeg dijagrama klase, ISP princip se primjenjuje na sljedeći način:

- **Interfejs ‘Korisnik’:** Umjesto da imamo jedan opći interfejs ‘Korisnik’ koji koriste sve klase (‘Donor’, ‘Zaposlenik’, ‘Administrator’), možemo imati više specifičnih interfejsa. Na primjer, možemo imati interfejs ‘IDonor’ za klasu ‘Donor’, interfejs ‘IZaposlenik’ za klasu ‘Zaposlenik’, i interfejs ‘IAdministrator’ za klasu ‘Administrator’. Svaki interfejs bi imao metode koje su specifične za datu klasu. Trenutno nije implementiran niti jedan interfejs.

Primjena Interface Segregation Principle-a pomaže u održavanju čistog dizajna softvera, smanjuje zavisnosti između klasa i olakšava održavanje koda. Na osnovu navedenog objašnjenja, Interface Segregation Principle bi vrijedio za naš konkretni projekat.

## Dependency Inversion Principle (DIP)

Dependency Inversion Principle (DIP) je peti SOLID princip koji predstavlja specifičnu metodologiju za slabo povezane softverske klase. Kada se ovaj princip primjenjuje, konvencionalni odnosi zavisnosti između visokih klasa koji postavljaju politike i niskih klasa na kojima ovi prvi zavise se preokreću, čime se visoke klase čine nezavisnim od detalja implementacije niskih klasa.

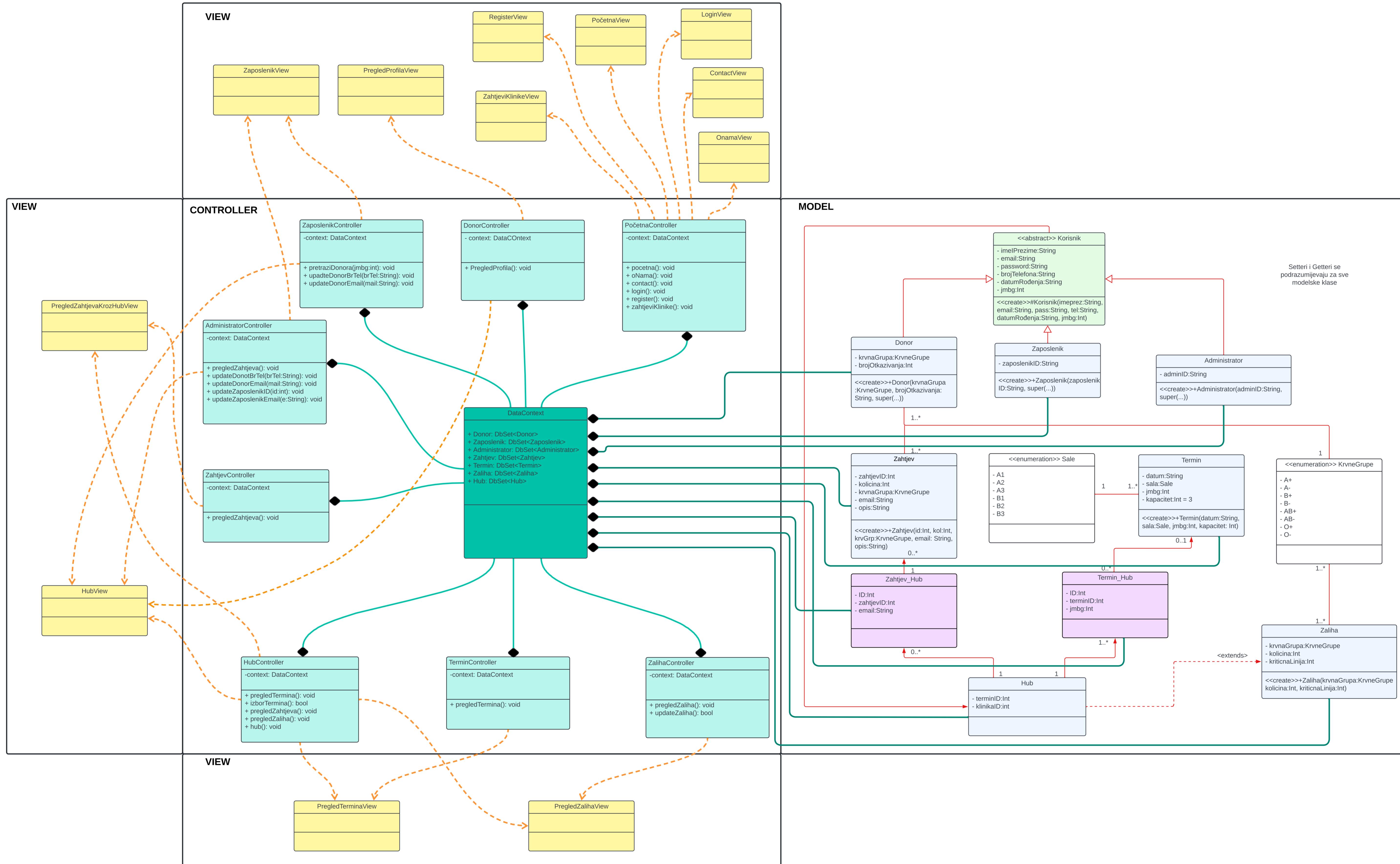
Princip navodi:

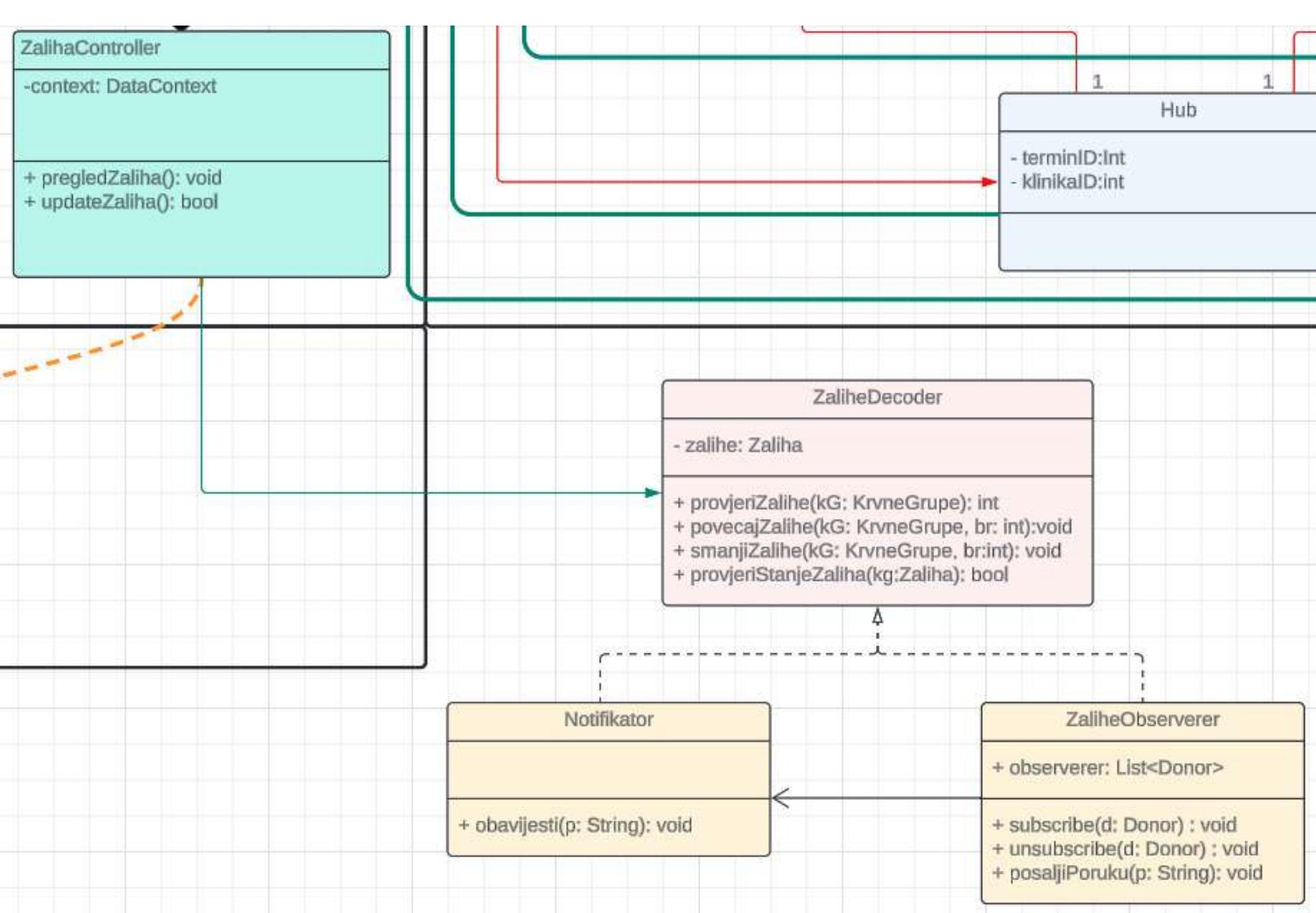
- Visoke klase ne bi trebale zavisiti o niskim klasama. Oba bi trebala zavisiti o apstrakcijama (npr. interfejsima).
- Apstrakcije ne bi trebale zavisiti o detaljima. Detalji (konkretnе implementacije) bi trebali zavisiti o apstrakcijama.

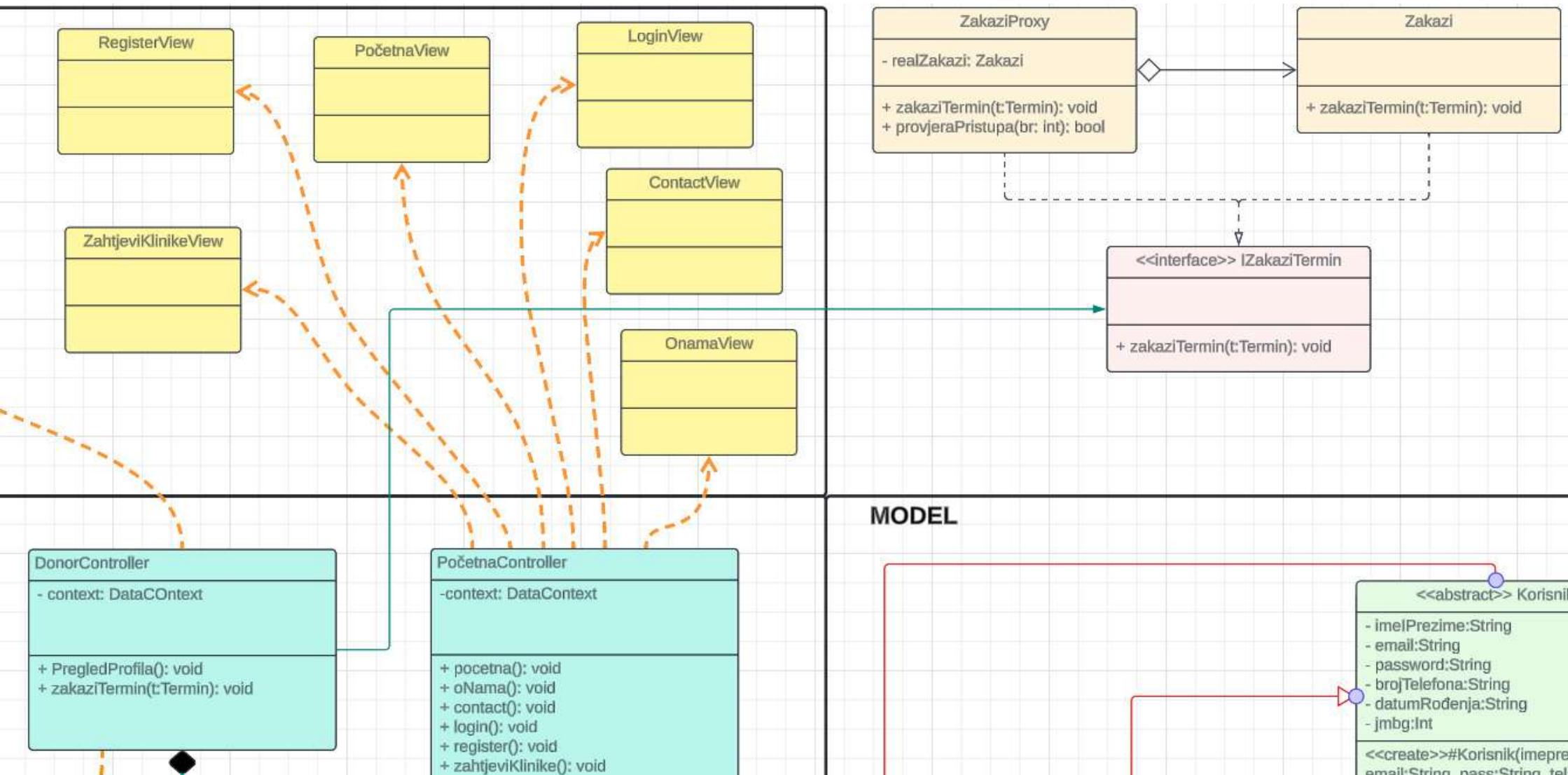
Na onovu našeg dijagrama klasa, DIP princip se primjenjuje na sljedeći način:

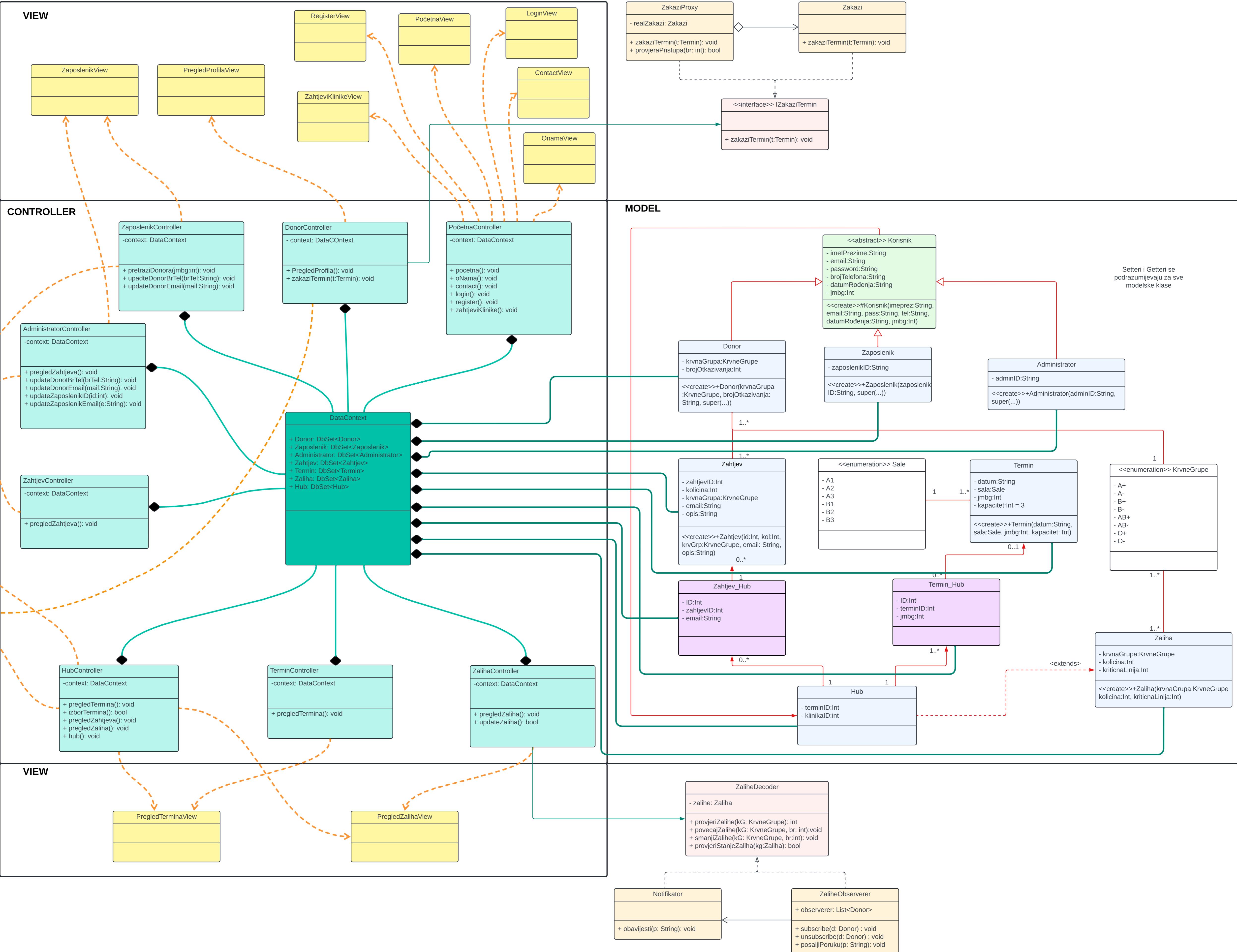
- **Apstraktna klasa ‘Korisnik’:** Kako su klase ‘Donor’, ‘Zaposlenik’ i ‘Administrator’ izvedene iz apstraktne klase ‘Korisnik’, navedene klase zapravo zavise od apstrakcije.

Kao i kod ISP-a, primjena Dependency Inversion Principle-a pomaže u održavanju čistog dizajna softvera, smanjuje zavisnosti između klasa i olakšava održavanje koda. Dakle, za naš dijagram klasa vrijedi Dependency Inversion Principle.









## **Proxy – Patern Strukture**

### **Opis**

Proxy objekat radi kao surrogat za stvarne objekte. Koristi se za omogućavanje pristupa i kontrolu pristupa stvarnim objektima. Manipulacija nad nekim objektom je dozvoljena ukoliko je neki uslov ispunjen ili ako korisnik npr. nema pravo pristupa traženom objektu.

### **Problemi koje rješava**

Kontrola pristupa: Proxy omogućava kontrolu pristupa stvarnim objektima, što znači da se određeni objekti mogu koristiti samo ako korisnik ispunjava određene uvjete ili ako ima odgovarajuće ovlasti.

### **Primjena**

Proxy uzorak koristi se u aplikaciji VitalFlow za kontrolu pristupa funkciji zakazivanja termina donacija, tj. pruža dodatnu logiku za upravljanje situacijama u kojima donor uzastopno propušta zakazane termine. Dakle ukoliko donor uzastopno propusti dva ili više zakazana termina, onda on u određenom vremenskom periodu neće biti u stanju da zakaže bilo kakav termin.

## **Facade – Patern Strukture**

### **Opis**

Facade je strukturalni patern dizajna koji pruža jednostavan interfejs za kompleksan sistem klasa ili podsistema. Omogućuje klijentima da jednostavno koriste složene operacije ili funkcionalnosti bez potrebe za detaljnim poznavanjem unutaršnjih procesa.

### **Problemi koje rješava**

Jednostavno obavještavanje donora na osnovu stanja zaliha: Donori primaju obavijesti o nedostatku krvi na jednostavan i učinkovit način.

### **Primjena**

U kontekstu aplikacije VitalFlow, Facade patern bi mogao biti primijenjen za pojednostavljinjanje procesa obavještavanja donora o nedostatku određene krvne grupe. Umjesto da svaki zaposlenik direktno pristupa logici provjere stanja zaliha i slanja notifikacija, Fasadu bi mogli koristiti kao jednostavnu tačku za pristup tim operacijama. Fasada bi se brinula o provjeri stanja zaliha za

određenu krvnu grupu i slanju notifikacije ako je potrebno, omogućavajući tako lakši i čistiji pristup ovim funkcionalnostima.

## **Decorator – Patern Strukture**

### **Opis**

Decorator patern je strukturalni patern dizajna koji omogućuje dinamičko dodavanje dodatnih funkcionalnosti ili ponašanja postojećim objektima bez mijenjanja njihove osnovne strukture. Ovaj patern omogućuje fleksibilno proširenje funkcionalnosti sistema tako da se nove funkcionalnosti mogu dodati ili ukloniti neovisno o kodu.

### **Problemi koje rješava**

Dinamičko dodavanje funkcionalnosti provjere zaliha: Decorator patern omogućava dodavanje funkcionalnosti provjere nivoa zaliha krvi bez menjanja osnovne strukture klase.

Automatsko obaviještanje donora: Možemo dodati funkcionalnost koja automatski šalje obaviještenja donorima kada zalihe određene krvne grupe spadnu ispod kritičnog nivoa. Ovo obaviještanje može biti putem e-maila, SMS-a ili drugih komunikacionih kanala.

### **Primjena**

Decorator se koristi za dinamičko dodavanje funkcionalnosti provjere zaliha krvi, kako bi se automatski obavijestili donori kada zalihe određene krvne grupe spadnu ispod kritične linije.

## **Adapter – Patern Strukture**

### **Opis**

Adapter patern se koristi kada treba omogućiti nekompatibilne interfejse da rade jedan sa drugim posredstvom Adapter klase koja implementira klijent interfejs i služi kao omotač za servis koji je potrebno adaptirati.

## **Problemi koje rješava**

Migracija podataka: Kada aplikacija treba migrirati s jedne baze podataka na drugu, npr. kada bi naša aplikacija trebala migrirati s SQL Servera/MySQL-a na MongoDB, suočili bismo se s različitim interfejsima i načinima komunikacije s bazom podataka.

## **Primjena**

Adapter patern rješava gore navedeni problem omogućujući prilagodbu operacija i zahtjeva aplikacije na interfejs koji podržava nova baza podataka, čime se olakšava migracija i održavanje aplikacije.

# **Bridge – Patern Strukture**

## **Opis**

Bridge pattern omogućuje odvajanje apstrakcije (klijentske logike) od implementacije (komunikacija s vanjskim sistemom klinike).

## **Problemi koje rješava**

Ako aplikacija komunicira s vanjskim sistemom klinike ili sličnim vanjskim servisom, važno je odvojiti klijentsku logiku (kako se aplikacija koristi) od detalja implementacije (kako se komunicira s vanjskim sistemom). Bridge pattern rješava ovaj problem omogućujući da se apstrakcija (Bridge) koristi za komunikaciju s vanjskim sistemom, dok konkretnе implementacije bridge-a rade s detaljima komunikacije.

## **Primjena**

Korištenjem Bridge Patterna u aplikaciji VitalFlow, omogućava se odvajanje klijentske logike od implementacijskih detalja komunikacije s vanjskim sistemima. Ovo povećava fleksibilnost, omogućava jednostavnije prilagođavanje i proširenje funkcionalnosti bez promjene osnovne logike aplikacije, te poboljšava održavanje koda.

# **Composite – Patern Strukture**

## **Opis**

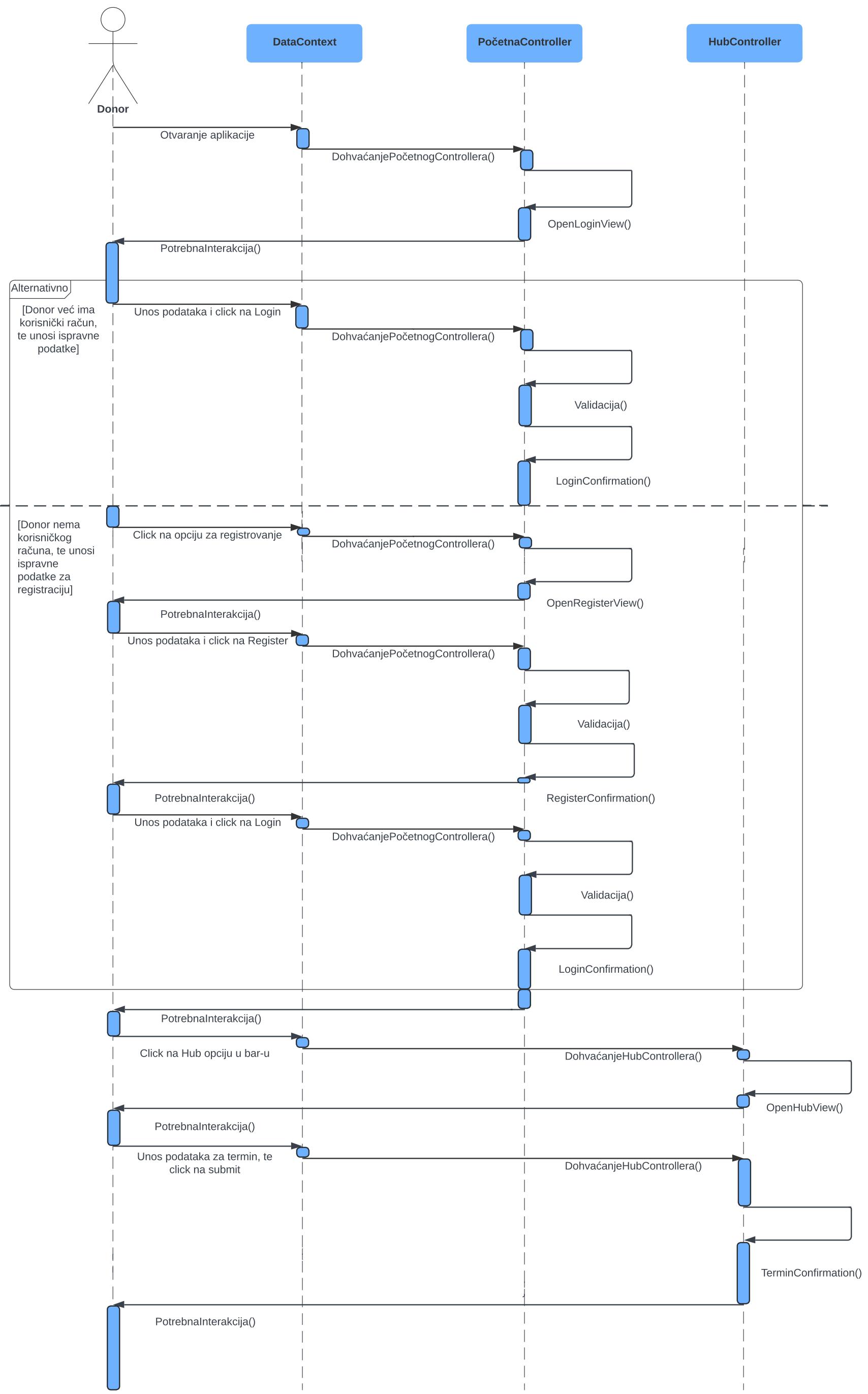
Composite pattern se često koristi kada treba raditi s objektima koji predstavljaju hijerarhijske strukture, poput stabla ili strukture odjeljenja u organizaciji.

## **Problemi koje rješava**

Jednostavno praćenje grupnih donacija: Kada bi omogućili grupne donacije, ukoliko želimo da lakše računamo koje sale su slobodne u kojim terminima, te koji donor pripada kojoj krvnoj grupi.

## **Primjena**

Ovaj pattern bi mogli implementirati ukoliko omogućimo grupne donacije, odnosno ukoliko bismo omogućili da donor pored toga što prijavi sebe ima mogućnost da prijavi i svoje prijatelje. Na ovaj način bi imali Composite klasu koja sadrži listu korisnika i leaf klasu kao pojedine korisnike različitih vrsta. Composite i leaf klase bi implementirale interfejs sa operacijama koje vršimo nad pojedinim objektima.



## **Factory – Kreacijski**

### **Opis**

Factory patern je kreacijski patern koji omogućava kreiranje objekata bez navođenja tačnih klasa objekata koji se kreiraju. Koristimo ga ukoliko nismo sigurni tačno sa kakvim tipom objekata će naš kod raditi. On definiše interfejs za stvaranje objekata, ali dozvoljava podklasama da promijene tip objekta koji će biti kreiran.

### **Problemi koje rješava**

U aplikaciji imamo različite vrste korisnika (Donor, Zaposlenik, Admin) koji dijele zajedničke osobine, ali također imaju specifične atribute i ponašanja. Bez Factory Patterna, instanciranje ovih različitih tipova korisnika može postati kompleksno i neorganizirano, naručito kada se doda potreba za inicijalizacijom specifičnih atributa.

### **Primjena**

Korištenjem Factory Patterna za kreiranje korisničkih objekata u aplikaciji VitalFlow, postiže se centralizacija i enkapsulacija logike za instanciranje različitih tipova korisnika. Ovo čini kôd čistijim, lakšim za održavanje i proširenje, te smanjuje mogućnost grešaka pri kreiranju objekata.

## **Abstract Factory – Kreacijski Patern**

### **Opis**

Abstract Factory Pattern je kreacijski patern koji pruža interfejs za kreiranje familija povezanih ili zavisnih objekata bez navođenja njihovih konkretnih klasa. Omogućava grupisanje logike za kreiranje srodnih objekata u okviru jedne fabrike. Ovaj pattern prati open-closed i single-responsibility principe. Slično kao i prethodni pattern, ovaj metod također centralizira kreiranje objekata na jedno mjesto u aplikaciji.

### **Problemi koje rješava**

U aplikaciji VitalFlow potrebno je upravljati raznovrsnim resursima kao što su zalihe krvi i notifikacije za donore i klinike. Bez odgovarajuće strukture, kreiranje i upravljanje ovim resursima može postati kompleksno i teško za održavanje.

### **Primjena**

Korištenjem ovog paterna u aplikaciji VitalFlow za upravljanje resursima kao što su zalihe krvi i notifikacije, postižemo bolju organizaciju i strukturu koda. Ovo omogućava lakše održavanje i proširenje aplikacije, osigurava dosljednost u kreiranju objekata i smanjuje mogućnost grešaka. Abstract Factory Pattern je idealan kada imamo potrebu za kreiranjem kompleksnih struktura

povezanih objekata, što je čest slučaj u složenim aplikacijama kao što je ova.

## Builder – Kreacijski Patern

### Opis

Builder Pattern je kreacijski patern koji omogućava konstruisanje složenih objekata korak po korak. Omogućava kreiranje različitih reprezentacija objekta koristeći istu konstrukciju.

### Problemi koje rješava

U aplikaciji VitalFlow imamo različite tipove korisnika (Admin, Donor, Zaposlenik) s mnogo atributa. Kreiranje ovih objekata korištenjem velikih konstruktora može dovesti do:

- **Nepreglednog koda:** Gigantsko dugi konstruktori s mnogo parametara postaju teško čitljivi i održivi.
- **Visoke vjerovatnoće grešaka:** Povećana mogućnost grešaka zbog nepravilnog redoslijeda ili zaboravljenih parametara prilikom instanciranja objekata.
- **Teškog održavanja:** Dodavanje ili izmjena parametara zahtijeva promjenu svih poziva konstruktora.

### Primjena

Koristićemo Builder Pattern za kreiranje različitih tipova korisnika (Admin, Donor, Zaposlenik) u našoj aplikaciji. Korištenjem Builder Patterna u aplikaciji VitalFlow za kreiranje korisničkih objekata, postižemo bolju organizaciju i strukturu koda. Ovo omogućava lakše kreiranje složenih objekata korak po korak, poboljšava čitljivost i održavanje koda, te smanjuje mogućnost grešaka prilikom instanciranja objekata. Builder Pattern je posebno koristan kada imate objekte sa mnogo optionalnih parametara ili kompleksnim inicijalizacijama.

## Prototype – Kreacijski Patern

### Opis

Prototype Pattern je kreacijski patern koji omogućava kloniranje postojećih objekata umjesto kreiranja novih objekata od nule. Omogućava kreiranje novih objekata na osnovu šablonu, čime se smanjuje kompleksnost i vrijeme potrebno za instanciranje.

### Problemi koje rješava

U aplikaciji VitalFlow, Prototype Pattern bi se mogao primjeniti za kloniranje instanci korisnika kao što su Donor, Zaposlenik ili Administrator, posebno ukoliko ove instance sadrže više kompleksnih atributa i trebaju se često kreirati sa sličnim početnim stanjima.

## Primjena

Iako su naše trenutne klase dovoljno jednostavne i ne zahtijevaju hitnu primjenu ovog paterna, Prototype Pattern može postati koristan kako se aplikacija bude razvijala i kako se budu dodavali složeniji atributi i ponašanja korisničkim objektima. Korištenje Prototype Patterna omogućilo bi jednostavnije i brže kreiranje kopija postojećih objekata što bi poboljšalo efikasnost i održavanje koda.

## Singleton – Kreacijski Patern

### Opis

Singleton Pattern je kreacijski patern koji osigurava da klasa ima samo jednu instancu i pruža globalnu tačku pristupa toj instanci. Ovo je korisno za upravljanje resursima koje je potrebno centralizovati.

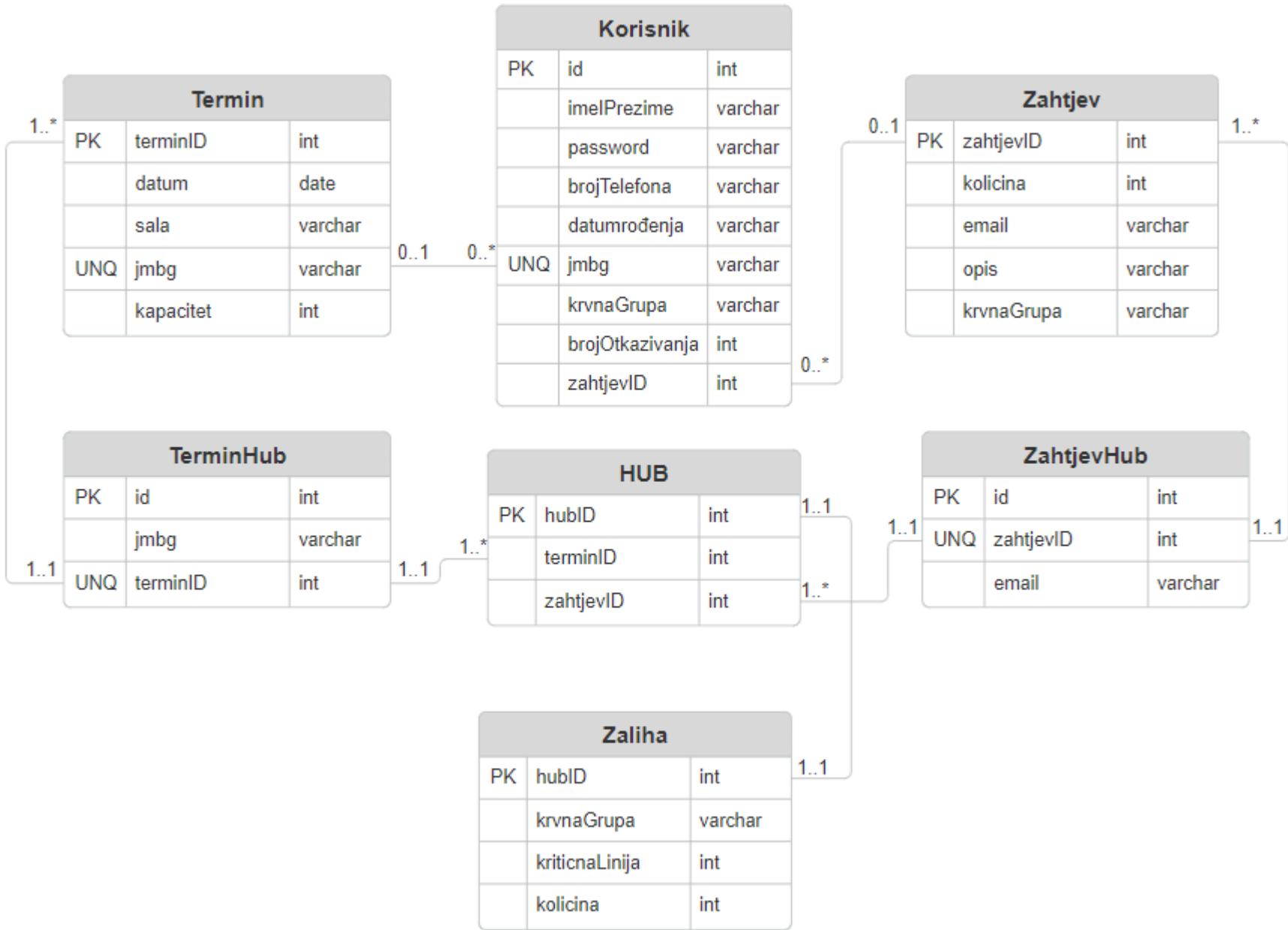
### Problemi koje rješava

U aplikaciji VitalFlow postoje resursi i stanja koja treba da budu jedinstvena kroz cijelu aplikaciju kao što su:

- **Stanje zaliha krvi:** Praćenje zaliha krvi mora biti centralizovano kako bi se osigurala tačnost podataka i efikasno upravljanje donacijama i potrebama.
- **Sistem za notifikacije:** Slanje notifikacija korisnicima treba biti centralizovano kako bi se osiguralo konzistentno obavještavanje svih korisnika.

## Primjena

Koristićemo Singleton Pattern za upravljanje stanjem zaliha krvi i sistemom za notifikacije. Korištenjem Singleton Patterna u aplikaciji za upravljanje globalnim stanjima i resursima kao što su stanje zaliha krvi i sistem za notifikacije, postižemo centralizovanu kontrolu, smanjujemo mogućnost grešaka i poboljšavate efikasnost koda. Singleton Pattern je idealan za resurse koji trebaju biti jedinstveni kroz cijelu aplikaciju, omogućavajući jednostavan i kontrolisan pristup.



# Paterni Ponašanja

## 1. Observer (Promatrač)

Observer pattern omogućava obavljanje donora krvi o potrebama klinika za određenim tipovima krvi. Kada zaliha krvi padne ispod kritične linije, sistem (subjekat) šalje obavijest svim donorima koji su promatrači (observers) i pripadaju odgovarajućoj krvnoj grupi. Donori tada mogu odlučiti da li žele donirati krv. Prednosti ovog pristupa uključuju rasterećenje sistema od direktnog kontaktiranja svakog donora i omogućavanje donorima da samostalno odluče kako će reagirati na obavijesti. Ovo također povećava fleksibilnost i modularnost sistema jer se novi promatrači mogu dodavati ili uklanjati bez utjecaja na ostatak sistema.

## 2. Strategy (Strategija)

Strategy pattern može se koristiti za definisanje različitih strategija obrade podataka i interakcije s korisnicima. Na primjer, za izračunavanje kritične linije zaliha krvi možemo imati različite strategije: jednu koja se bazira na historijskim podacima o potrošnji krvi, drugu koja koristi trenutne statističke modele i treću koja se oslanja na sezonske varijacije. Svaka strategija će biti implementirana kao zasebna klasa koja implementira zajednički interfejs za izračunavanje kritične linije. Ovo omogućava lako mijenjanje strategija bez potrebe za izmjenom osnovne logike aplikacije.

## 3. Iterator (Iterator)

Iterator pattern omogućava korisnicima prolazak kroz listu donora, zaliha krvi ili rezerviranih termina bez otkrivanja unutrašnje strukture tih kolekcija. Na primjer, donori mogu koristiti iterator za pregled svojih prošlih donacija, dok zaposlenici mogu koristiti iterator za prolazak kroz listu trenutnih zaliha krvi ili predstojećih termina za donaciju. Iterator omogućava standardiziran način prolaska kroz podatke, čineći aplikaciju fleksibilnijom i lakšom za održavanje.

## 4. Memento (Memento)

Memento pattern se može koristiti za snimanje trenutnog stanja profila donora ili stanja zaliha krvi i vraćanje tog stanja u budućnosti. Na primjer, prilikom promjene korisničkih podataka, možemo snimiti trenutno stanje korisničkog profila kao memento objekat. Ako korisnik kasnije poželi vratiti prethodno stanje, sistem može koristiti memento objekat za vraćanje podataka. Ovo omogućava korisnicima sigurnost da mogu vratiti prethodno stanje u slučaju greške ili neželjenih promjena.

## 5. Chain of Responsibility (Lanac odgovornosti)

Chain of Responsibility patern može se koristiti za validaciju podataka prilikom registracije donora ili prilikom obrade zahtjeva za donaciju. Svaki korak u procesu validacije (npr. provjera ispravnosti formata broja telefona, provjera kompatibilnosti krvne grupe s trenutnim potrebama, provjera ispravnosti unesenih osobnih podataka) može biti zaseban handler u lancu odgovornosti. Ako jedan handler ne može obraditi zahtjev, on ga proslijeđuje sljedećem handleru u lancu. Ovo omogućava modularnu i fleksibilnu obradu zahtjeva.

## **6. Mediator (Posrednik)**

Mediator pattern može se koristiti za olakšavanje komunikacije između različitih komponenti sistema, kao što su modul za registraciju donora, modul za upravljanje zalihamama krvi i modul za slanje notifikacija. Posrednik upravlja interakcijama između ovih komponenti, smanjujući direktne zavisnosti među njima. Na primjer, kada se novi donor registruje, modul za registraciju obavještava posrednika, koji zatim obavještava modul za slanje notifikacija i modul za upravljanje zalihamama kako bi ažurirali svoje podatke i poslali relevantne obavijesti.

## **7. Template Method (Template metoda)**

Template Method pattern može se koristiti za definiranje koraka u procesu doniranja krvi, gdje se zajednička struktura algoritma nalazi u apstraktnoj klasi, a specifični koraci se implementiraju u podklasama. Na primjer, proces rezervacije termina za donaciju krvi može imati sljedeće korake: izbor termina, unos podataka, potvrda rezervacije i slanje obavijesti. Apstraktna klasa može definirati template metodu RezervacijaTermina(), dok će specifične metode kao što su IzborTermina(), UnosPodataka() i SlanjeObavijesti() biti implementirane u podklasama. Ovo omogućava lako prilagođavanje i proširenje procesa rezervacije bez mijenjanja osnovne strukture algoritma.

Navedeni Paterni Ponašanja omogućavaju efikasno i modularno upravljanje funkcionalnostima unutar sistema "VitalFlow", poboljšavajući njegovu fleksibilnost, održivost i korisničko iskustvo.

## **8. Interpreter Pattern (Tumač)**

Interpreter pattern se može koristiti za tumačenje ili analiziranje jezika specifičnog za domenu unutar aplikacije. U slučaju aplikacije VitalFlow, ovaj patern bi se mogao koristiti za tumačenje i izvršavanje složenih pravila ili upita koji se odnose na upravljanje zalihamama krvi, filtriranje donacija, ili generisanje izvještaja. Na primjer, možemo imati jednostavan jezik koji omogućava administratorima da definišu pravila za filtriranje donacija krvi na osnovu različitih kriterija kao što su krvna grupa, datum donacije, ili status obrade donacije. Ovaj jezik može imati sintaksu kao što je: "krvna\_grupa = 'A+' AND datum > '2023-01-01' AND status = 'prihvaćeno'".

## **9. Command (Komanda)**

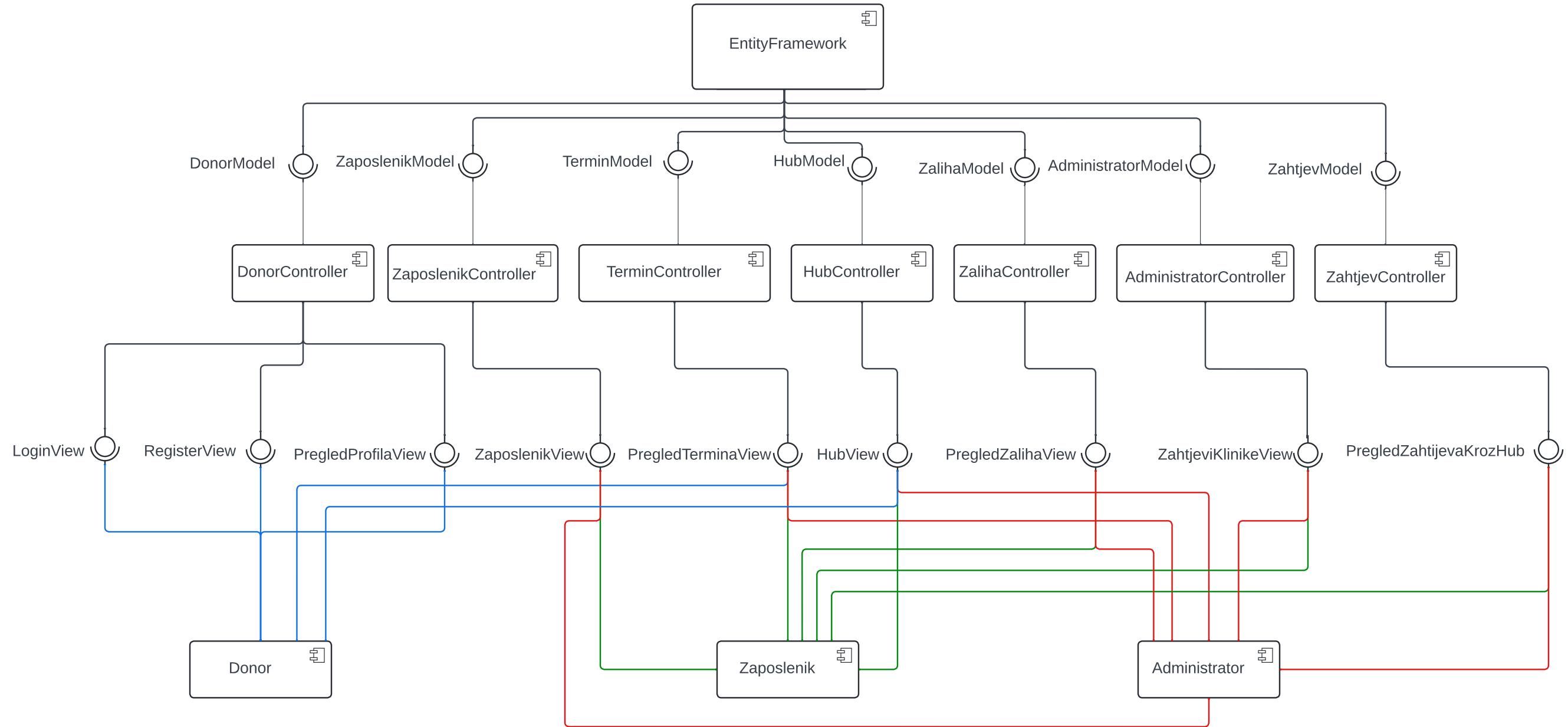
Command pattern je pattern ponašanja koji pretvara zahtjeve ili jednostavne operacije u objekte. Ovaj pattern omogućava parametrizaciju klijenata sa različitim zahtjevima, redoslijedima ili zapisima operacija i podršku za operacije poništavanja. U kontekstu aplikacije VitalFlow, Command pattern se može koristiti za implementaciju operacija kao što su registracija donora, rezervacija termina za donaciju, ili obavještavanje korisnika. Na primjer, svaka operacija može biti enkapsulirana kao objekat komande, koji sadrži sve potrebne informacije za izvršenje te operacije.

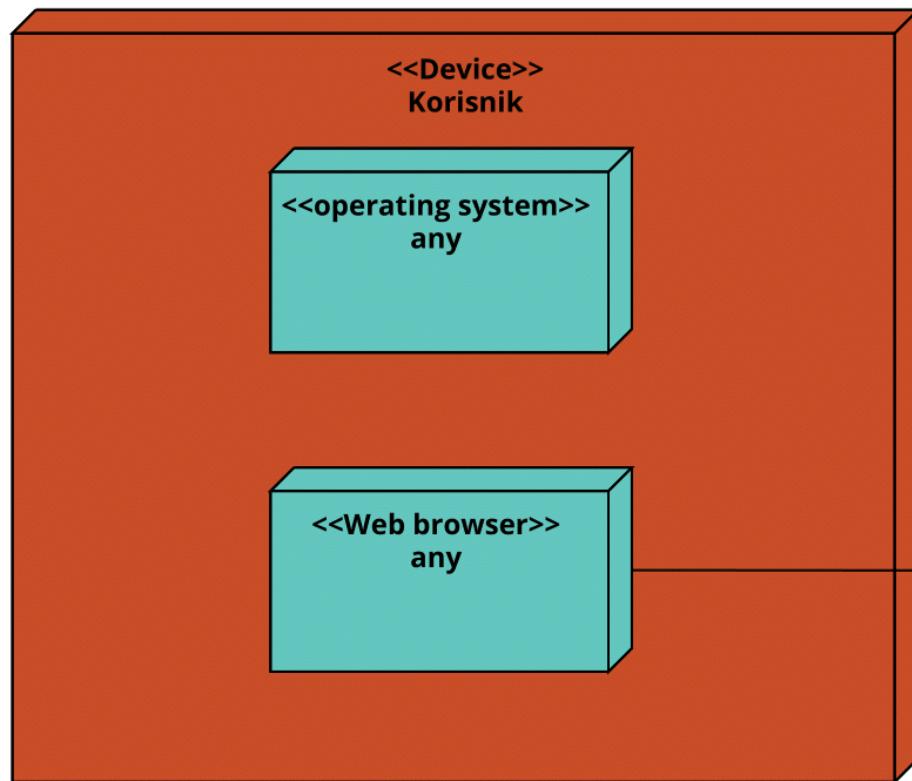
## **10. Visitor (Posjetioc)**

Visitor pattern je pattern ponašanja koji omogućava dodavanje novih operacija na objekte bez promjene tih objekata. To se postiže definisanjem operacija u zasebnoj klasi koja se naziva Visitor (posjetioc). Objekti koje treba posjetiti (elementi) prihvataju posjetioca i omogućavaju mu da izvrši operacije na njima. U kontekstu aplikacije VitalFlow, Visitor pattern se može koristiti za izvršavanje različitih operacija nad podacima donora krvi bez promjene klase donora. Na primjer, možemo imati različite posjetioce za operacije kao što su generisanje izveštaja, verifikacija podataka ili ažuriranje statistika.

## **11. State (Stanje)**

State pattern je pattern ponašanja koji omogućava objektu da promijeni svoje ponašanje kada mu se promijeni unutrašnje stanje. Objekt će izgledati kao da je promijenio svoju klasu. Ovo je korisno kada objekt treba da se ponaša različito u zavisnosti od svog stanja. U kontekstu aplikacije VitalFlow, State pattern se može koristiti za upravljanje različitim stanjima donacija krvi, kao što su "primljena", "obrađena", "odobrena" i "odbijena". Svako od ovih stanja može imati različito ponašanje i reakcije na događaje.





**<<HTTPS>>**

