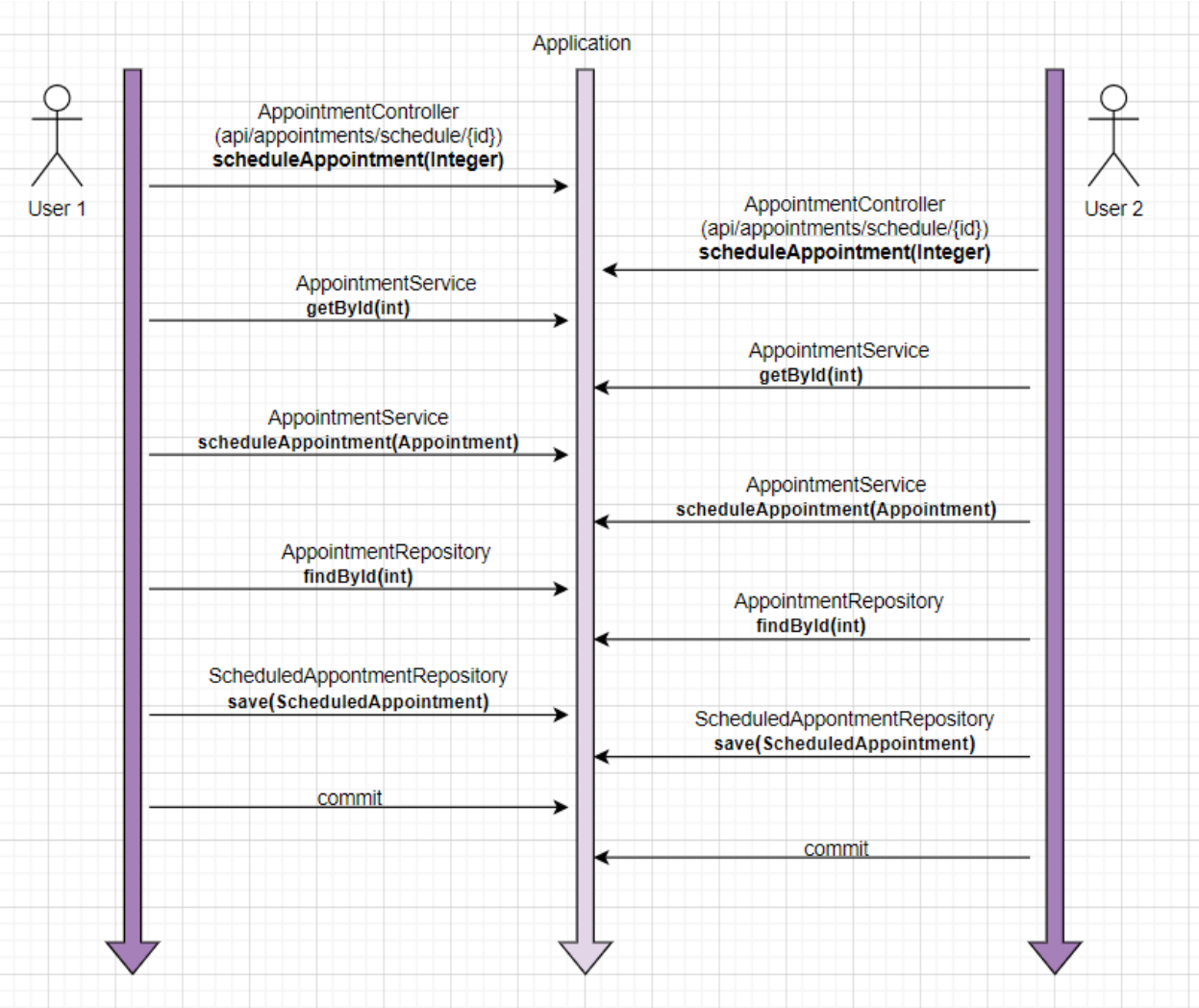
Konfliktne situacije-student 4 (Sara Jokic RA 75/2019)

1) Termini koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika

Opis konfliktne situacije: Korisnik na svom home page-u moze da vidi sve termine koji su slobodni u centrima I moze da rezervise neke od njih. Do konflikta moze doci ako 2 ili vise korisnika pokusaju istovremeno da zakazu isti termin. Tada bi se u bazu podataka upisalo vise objekata klase schaduled appointment koji se vezuju za razlicite korisnike, a iste predefinisane termine.



Resenje konfliktne situacije: Konfliktna situacija je resena koriscenjem optimistickog zakljucavanja klase Appointment. Uvodi se novi atribut Version sa anotacijom @Version. Prilikom izmene available atributa klase Appointment, inkrementuje se Version atribut. Komitovanje ce biti spreceno ukoliko se atribut version u bazi I transakciji ne poklapaju. Provera verzije radi se pesimistickim zakljucavanjem.

Ova konfliktna situacija je testirana koristeci 2 niti. Nit 1 ucitava appointment koji treba da se zakaze, nakon toga nit1 se uspavljuje na nekoliko sekundi. Za to vreme nit 2 ucitava isti appointment I zakazuje ga izmenom available atributa i komituje izmenu. Verzija u bazi je inkementovana za 1. Nakon toga nit 1 se budi I pokusava da zakaze appoitnemnt koji je prethodno ucitala ali kada poredi verziju ucitanog appointment-a I onog u bazi nece se poklopiti I javice ObjectOptimisticFailureExeption.

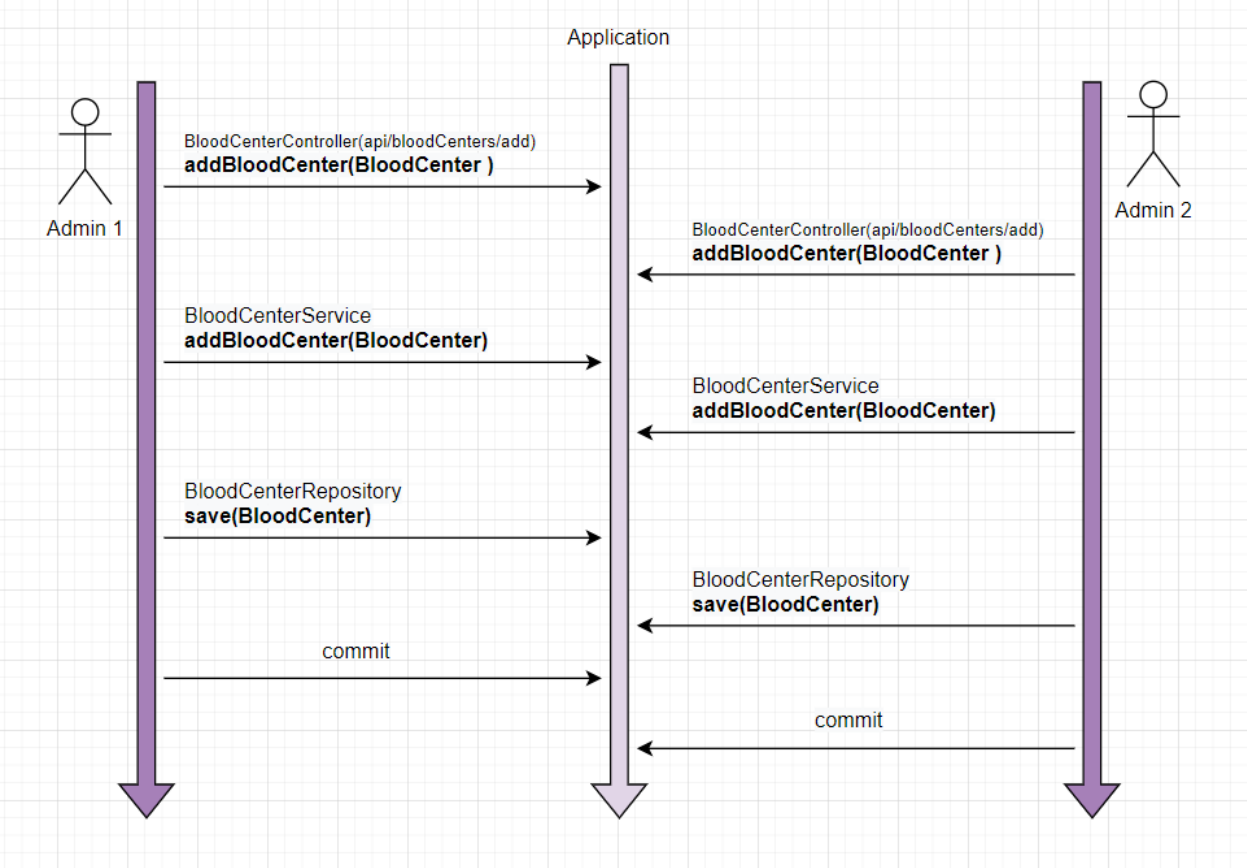
2) Ista banka krvi ne sme biti registrovana od strane vise razlicitih admina

Opis konfliktne situacije: Admin moze na svom profilu da registurje novu banku krvi unoseci njene informacije, do konflikta dolazi ukoliko vise administratora sistema istovremeno sa razlicitih uredjaja pokusa da kreira banku krvi sa istim kredencijalima. U bazu ce se upisati dve banke krvi sa istim informacijama, sto je nepozeljno.

Resenje konfliktne situacije: Podizemo nivo izolacije transakcije na serializable. Iznad metode addBloodcenter u servisu dodajemo anotaciju @Transactional sa atributom isolation=isolation.SERIALIZABLE.

Ovaj atribut ce osigurati da se transakcije izvrsavaju sekvencijalno. Dok se jedna transakcija izvrsava druge ne mogu da modifikuju ni citaju podatke koje koristi prva transakcija.

Ovu konfliktnu situaciju smo testirali koriscenjem 2 niti gde ce nit koja prva dodje na izvrsenje metode addBloodCentar uspesno izvrsiti metodu dok ce druga nit ostati blokirana. Zbog toga drugu nit uspavljujemo na nekoliko sekundi I potom je pozivamo da izvrsi metodu addBloodCnetar, to nece moci da uradi jer je blood centar sa tim imenom vec upisan u bazu podataka I javice BloodCenterAlreadyExtists exeption.

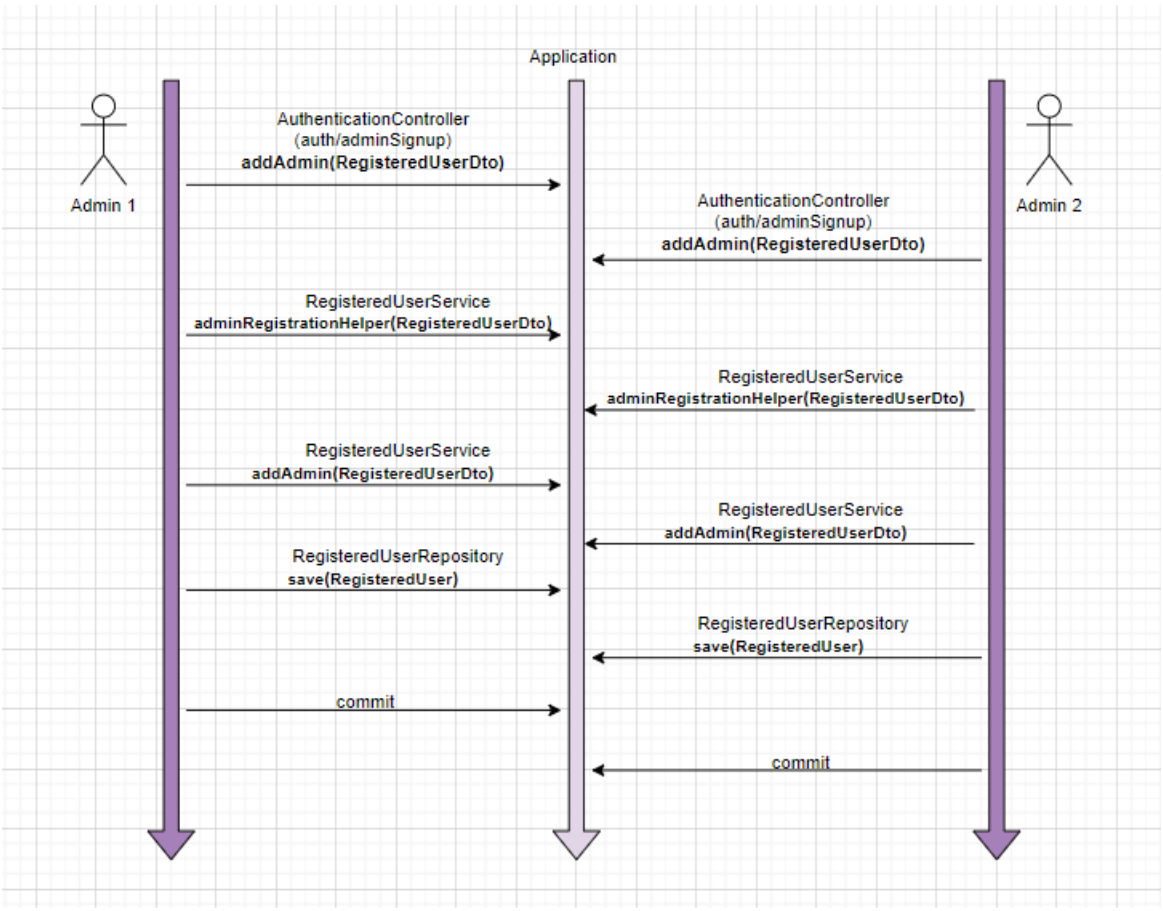


3) Vise administratora istovremeno pokusava da registruje novog administratora sistema

Opis konfliktne situacije: Admin sistema ima mogucnost da registruj novog admina sistema. Moze doci do konfliktne situacije ukoliko istovremeno vise admina sa razliitih uredjaja pokusa da registruje novog admina.

Resenje konfliktne situacije: Takodje kao u prethodnoj konfliktoj situaciji, podigli smo nivo izolacije transakcije uvodjenjem @Transactional sa atributom isolation=isolation.SERIALIZABLE. Ova anotacija postavljena je iznad addAdmin metode. Tako smo osigurali sekvencijalno izvrsavanje transakcija.

Ovu konfliktnu situaciju testirali smo na slican nacin kao prethodno navedenu , koriscenjem dve niti koje se “utrkuju” , ona koja bude brza izvrsice metodu addAdmin, dok druga ostaje blokirana. Nakon toga pozivamo nit 2 nakon nekoliko sekundi i navedemo je da pokusa da upise admina sa istim kredencijalima. Kako druga nit nije uspesno izvrsila zadatak test prolazi javljajuci AdminAlreadyExists exeption.


Application
Admin 1
Admin 2
AuthenticationController
(auth/adminSignup)
addAdmin(RegisteredUserDto)
AuthenticationController
(auth/adminSignup)
addAdmin(RegisteredUserDto)
RegisteredUserService
adminRegistrationHelper(RegisteredUserDto)
RegisteredUserService
adminRegistrationHelper(RegisteredUserDto)
RegisteredUserService
addAdmin(RegisteredUserDto)
RegisteredUserService
addAdmin(RegisteredUserDto)
RegisteredUserRepository
save(RegisteredUser)
RegisteredUserRepository
save(RegisteredUser)
commit
commit