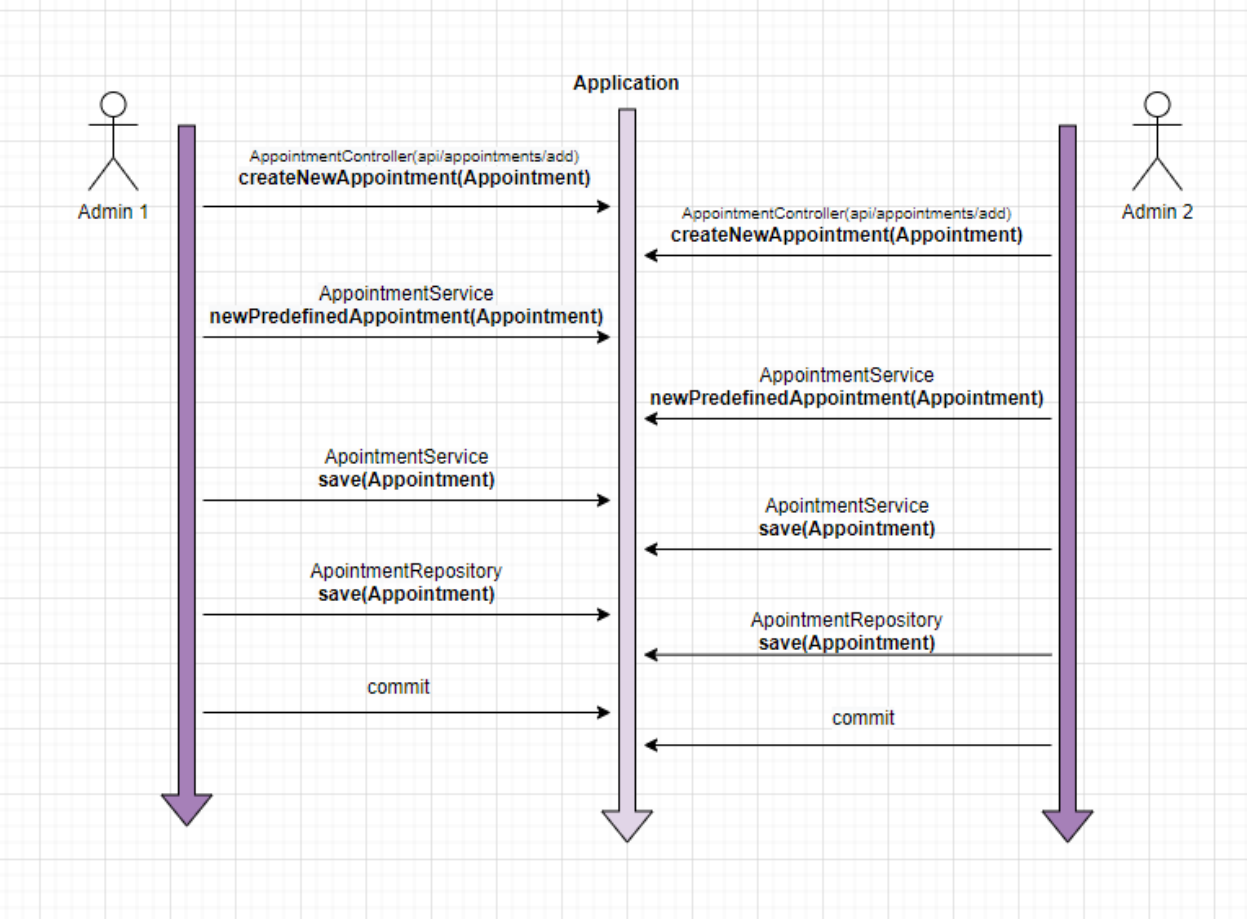
Konfliktne situacije Student 2 (Tamara Krgović RA54/2019)

1. Više administratora centra ne mogu unapred definisati termine u isto ili preklapajuće vreme

Opis konflikta: Administrator centra ima mogućnost da unapred definiše slobodne termine u svom centru. Može doći do konflikta ako dva administratora, sa različitih uređaja, istovremeno pokušaju da zakažu termine čiji se početak i vreme trajanja preklapaju. U bazi se tada sačuvaju dva termina sa preklapajućim početkom i vremenom trajanja.

Crtež toka konfliktne situacije:



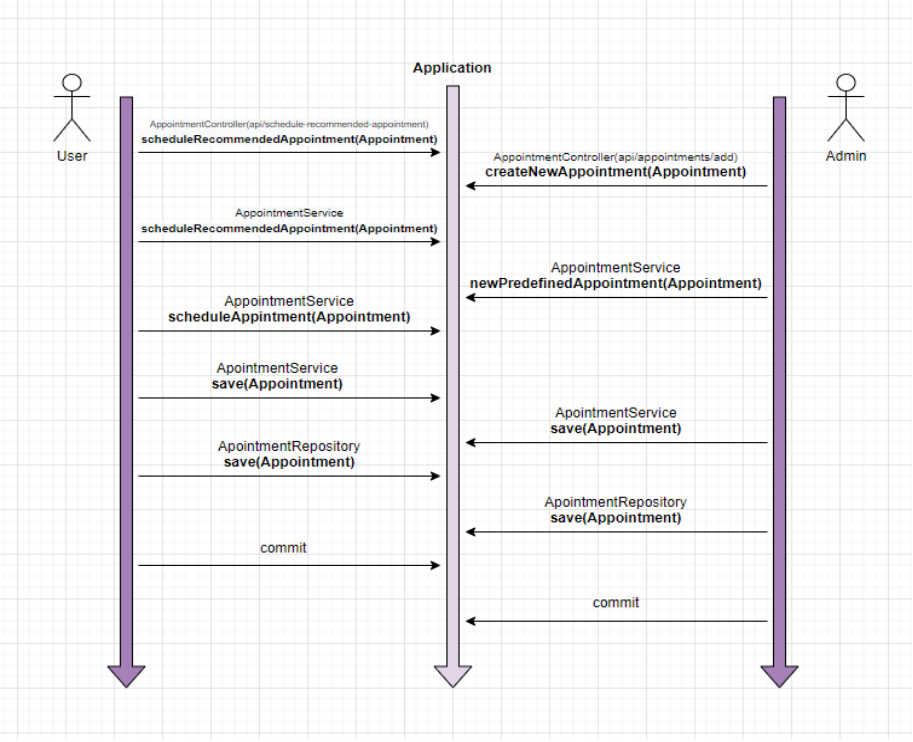
Opis rešenja: Nivo izolacije transakcije se podiže na najviši nivo *SERIALIZABLE* dodavanjem parametra isolation = Isolation.SERIALIZABLE u anotaciju @Transactional iznad metode newPredefinedAppointment(Appointment appointment). Ovaj nivo izolacije osigurava da će se

transakcije izvršavati jedna za drugom, nijedna transakcije neće moći da menja podatke koje neka transakcija trenutno koristi, sve dok se ona u potpunosti ne izvrši. Sada kada admini pokušaju da definišu nove predefinisane termine, onaj prvi će uspešno odradititi operaciju i sačuvati novi termin u bazu dok će drugom biti zabranjen pristup bazi dok se prva operacija ne izvrši. Kada pokuša opet da zakaže taj termin, dobiće grešku da termin u tom periodu već postoji.
Ova konflikta situacije je testirana pomoću testova. Imamo dva thread-a koja predstavljaju admine. Oba threada pokušavaju da izvrše operaciju kreiranja predefinisanog termina. Onaj Thread koji prvi krene sa izvršavanjem, blokiraće ovaj drugi. Zbog toga thread koji je blokiran uspavljujemo na par sekundi dok se ne izvrši prethodna transakcija. Nakon tih par sekindi, thread opet pokušava da izvrši operaciju dodavanja novog predefinisanog termina ali javiće mu se exception(OverlappingAppointmentException) koji kaže da termin sa zadatim početkom i trajanjem već postoji u bazi.

2. Administrator centra ne može unapred definisati termin u isto ili preklapajuće vreme za koje i korisnik kreira rezervaciju termina

Opis konflikta: Kao što je pomenuto u prethodnoj situaciji, admin ima mogućnost da unapred definiše slobodne termine. Takođe korisnik može, navođenjem vremena početka i trajanja termina, da iz ponuđenih centara izabere jedan u kom želi da kreira rezervaciju termina. Do konflikta može doći kada korisnik i administrator pokušaju istovremeno da obave svoje željene operacije. Tada se u bazi čuvaju dva preklapajuća termina što nije poželjno.

Crtež toka konfliktne situacije:

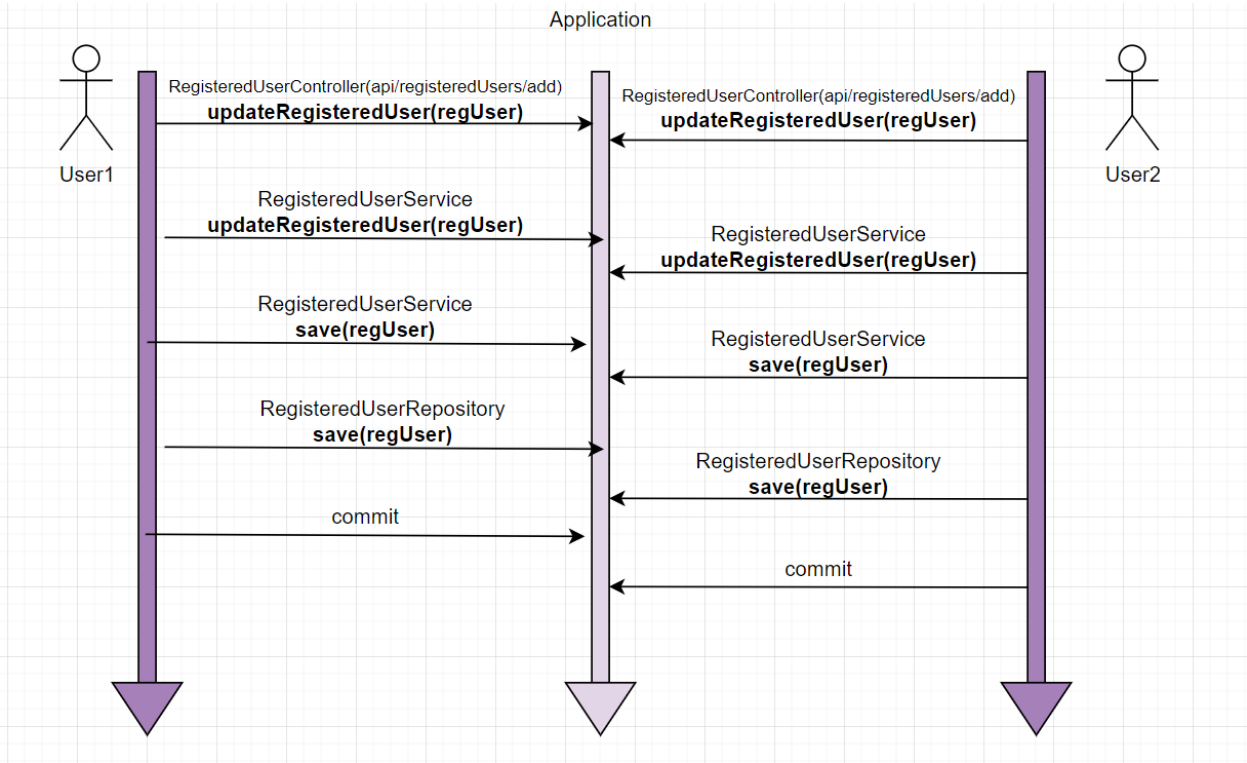


Opis rešenja: Nivo izolacije transakcije se isto podiže na najviši nivo *SERIALIZABLE* dodavanjem parametra isolation = Isolation.SERIALIZABLE u anotaciju @Transactional iznad metode scheduleAppointment(Appointment appointment). Ovaj nivo izolacije osigurava da će se transakcije izvršavati jedna za drugom, nijedna transakcije neće moći da menja podatke koje neka transakcija trenutno koristi, sve dok se ona u potpunosti ne izvrši.
Ova konfliktna situacija je takođe testirana pomoću dva thread-a, kao i prethodna.

3. Više korisnika istovremeno pristupa istom nalogu i pokušava da ga ažurira

Opis konfliktne situacije: Korisnik ima mogućnost da pregleda svoj profil i ažurira svoje lične informacije. Do konfliktne situacije može doći ako više korisnika istovremeno, sa različitih uređaja, pristupaju istom nalogu i pokušaju da ga ažuriraju. U ovom slučaju, sve izmene koje je User 1 napravio nad objektom RegisteredUser će biti izgubljene i u bazi će biti sačuvane izmene koje je User 2 napravio.

Crtež toka konflitne situacije:



Opis rešenja: Koristi se optimističko zaključavanje. U klasi RegisteredUser dodato je polje Version sa anotacijom @Version, koje se svakom promenom tog objekta za koji je vezan inkrementuje. Na taj način, svaki put pre nego što se user ažurira, porede se verzije učitanog objekta i onog u bazi da bi se proverilo da li se podaci nisu menjali u međuvremenu.
Ova konfliktna situacija je testirana pomoću testova. Pokrenuta su dva thread-a, Thread 1 učita user-a koji se ažurira, i onda se uspava na 3 sekunde da bi Thread 2 učitao tog istog user-a, izmenio podatke i odradio operaciju ažuriranja (i tada promenio verziju tog objekta za 1). Nakon 3 sekunde, kada prvi thread pokuša da odradi operaciju baciće se ObjectOptimisticLockingFailureException jer se verzija koju je Thread 1 učitao i ona koja je u bazi ne poklapaju.