Konkurentni pristup – Student 3

Konfliktna situacija 1:

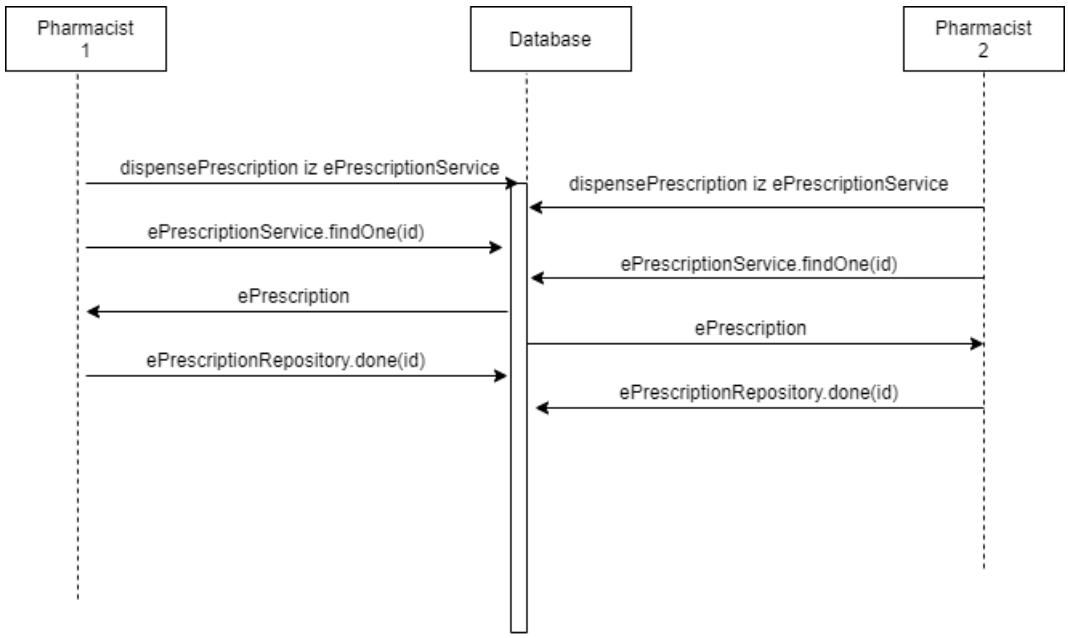
Opis konfliktne situacije

Farmaceuti koji rade u istoj apoteci pokušaju da izdaju istovremeno isti e-recept.

Tok zahteva klijenta i odgovor servera

Ukoliko dva farmaceuta sa dve različite mašine u isto vreme pokušaju da izdaju isti e-recept pacijentu, može doći do konfliktne situacije.

Endpoint koji se gađa GET zahtevom je /api/eprescriptions/{id_recepta}/dispense. Nakon uspešne autorizacije farmaceuta, metodi dispensePrescription iz ePrescriptionService se prosleđuje imejl autorizovanog farmaceuta i id recepta. U ovoj servisnoj metodi se prvo dobavlja ePrescription prosleđenog id-a i proverava se da li farmaceut radi u apoteci u kojoj treba da se izda recept, kao i to da li je prošao rok za izdavanje. Ukoliko ove provere prođu uspešno, pozivom ePrescriptionRepository.done(id) se postavlja flag koji označava da je e-recept izdat. Ovde nastaje problem jer teoretski može više farmaceuta doći do ovog dela u metodi, što znači da bi se recept izdao više puta. U nastavku je prikazan jedan primer ovakve konfliktne situacije.



Rešenje

Ovaj problem rešen je optimističkim zaključavanjem, odnosno dodavanjem version atributa/kolone u ePrescription klasu/tabelu, jer će se na taj način sprečiti dupli update zbog činjenice da će se u transakcijama koje pokušaju naknadno da menjaju ovaj red u tabeli verzija reda razlikovati od početne.

Konfliktna situacija 2:

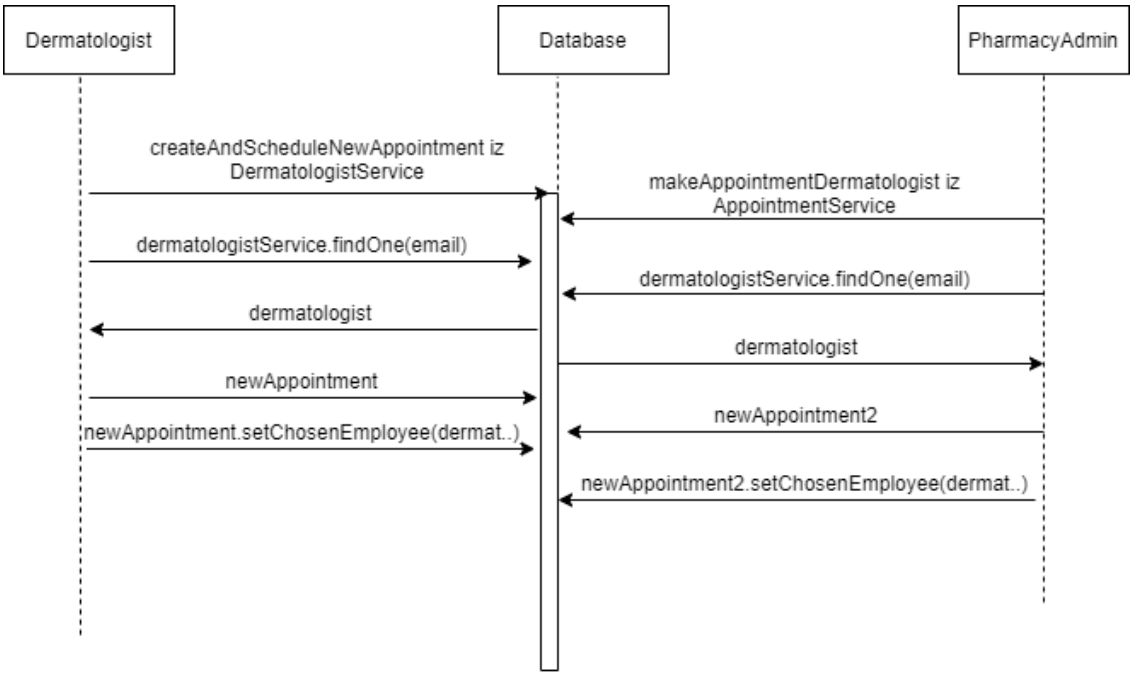
Opis konfliktne situacije

Dermatolog pokušava da kreira i zakaže termin za pacijenta koji je u međuvremenu postao nevalidan (npr. zakazan drugi termin dermatologa ili farmaceuta koji se preklapa sa ovim terminom).

Tok zahteva klijenta i odgovor servera

Ukoliko dermatolog pokuša da kreira i zakaže termin za pacijenta koji je u međuvremenu postao nevalidan, može doći do konfliktne situacije.

Endpoint koji se gađa POST zahtevom za zakazivanje termina za pacijenta je /api/dermatologist/appointments/schedule-new/{medical-report-id}. Tok koji je ugrubo opisan na narednoj slici predstavlja primer jedne konfliktne situacije za ovu funkcionalnost, gde kada dermatolog pokuša da kreira i zakaže novi termin za pacijenta, admin apoteke kreira novi slobodan termin za tog dermatolog, što dovodi do konfliktne situacije jer se termini bilo kog tipa ne smeju preklapati.



Rešenje

Ovaj problem rešen je pesimističkim zaključavanjem, gde je u dermatologistService dodata metoda findOneWithLock, koja poziva repozitorijumsku metodu u dermatologistRepository:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select d from Dermatologist d where d.email = :email")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
Dermatologist findOneWithLock(@Param("email") String email);
```

Ova metoda se poziva na početku servisne metode, pre bilo kakvih provera validnosti novog termina. Ovo rešenje je izabrano zato što će samo jedna transakcija koja bi pokušala da doda termin u jednom trenutku posedovati ključ nad tim redom u tabeli, te neće moći da dođe do konfliktne situacije.

Napomena: s obzirom na to da metoda za kreiranje i zakazivanje novog termina sadrži mnogo provera koje zavise i od ostatka sistema, posmatraće se ovaj problem u izolaciji i smatraće se da je dovoljno da se izvrši pesimističko zaključavanje nad dermatologom kako bi se izbegla konfliktna situacija.