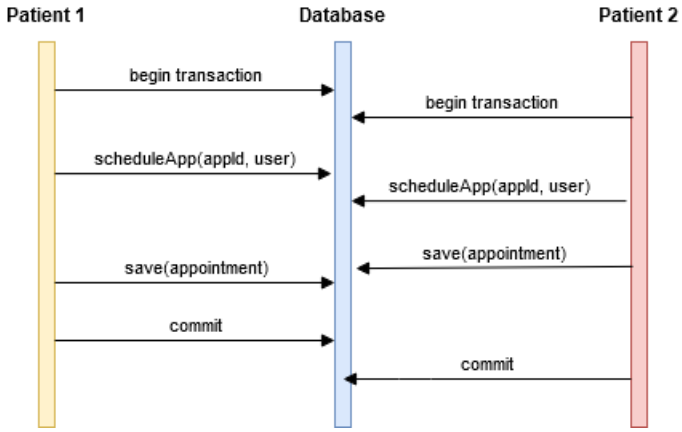
Konfliktne situacije – student 2 (Tamara Ranković)

1. *Pregledi koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika*

Opis konfliktne situacije: Predefinisani pregledi kreiraju se tako što administrator apoteke bira dermatologa koji je u njoj zaposlen, datum pregleda i njegovo trajanje, kao i cenu. Ukoliko su svi preduslovi zadovoljeni (da dermatolog nije na odmoru/bolovanju, da nema preklapajućih termina i da tada radi u toj apoteci), termin se kreira. Ulogovani pacijenti imaju prikaz liste svih apoteka i za svaku od njih mogu dobiti prikaz detaljnih informacija. U okviru te stranice nalazi se spisak svih slobodnih predefinisanih pregleda kod dermatologa koje pacijent može zakazati jednim klikom. Konfliktna situacija javlja se kada više pacijenata pokuša da zakaže isti predefinisani pregled.

Crtež: Prvi pacijent započinje transakciju pre drugog pacijenta, zatim oba pacijenta vrše zakazivanje pregleda i komituju izmene tako da drugi pacijent dobija pregled, što predstavlja problem jer je prvi pacijent pokušao da izvrši zakazivanje pre drugog pacijenta, ali neuspešno. Dolazi do Lost update-a i aplikacija se dovodi u nekonzistentno stanje.

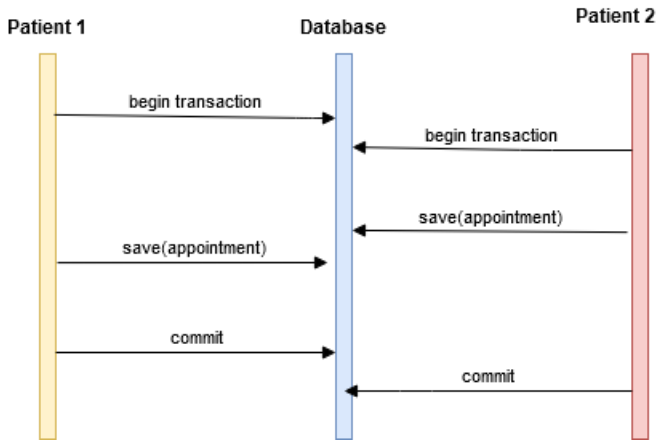


Rešenje: Kako bi se izbegao opisani problem, primenjuje se strategija optimističkog zaključavanja nad objektom DermatologistAppointment-a. U klasu se dodaje polje long version sa anotacijom @Version koje pamti izmene izvršene nad konkretnim pregledom. Kada pacijenti zatraže dostupne preglede, dobijaju i verziju tog pregleda u tom trenutku. Kada prvi pacijent započne transakciju i napravi izmenu pregleda tako što mu dodeli pacijenta, ukoliko drugi pacijent započne transakciju i pokuša da takođe izmeni isti pregled pre nego što je izvršen komit prve transakcije, dobiće odgovor da pregled nije više moguće zakazati jer ga je neko drugi zakazao. Do toga će doći zato što se izmenom pregleda od strane prvog pacijenta vrši inkrement polja version što će u slučaju drugog pacijenta biti indikator da je u međuvremenu došlo do neke izmene i da on ne može da zakaže taj pregled.

2. *Više istovremenih korisnika aplikacije ne može da zakaže savetovanje u istom terminu kod istog farmaceuta (termini se ne smeju ni preklapati)*

Opis konfliktne situacije: Ulogovani pacijent ima mogućnost zakazivanja savetovanja kod farmaceuta u nekoliko koraka. On prvo bira datum i vreme za koje želi da zakaže savetovanje, zatim iz liste dobijenih apoteka koje zadovoljavaju taj uslov bira jednu i dobija spisak svih farmaceuta koji su slobodni u tom terminu. Od njih, on bira jednog i vrši potvrdu zakazivanja. Do konflikta može doći ukoliko je dva ili više pacijenata istovremeno odabralo preklapajući termin u istoj apoteci kod istog farmaceuta.

Crtež: Prvi pacijent bira datum i vreme i zatim vrši upit kojim dobija spisak svih apoteka u kojima može zakazati savetovanje kao i sve famaceuta u njoj koji su slobodni. Istu akciju vrši i drugi pacijent i dobija potpuno iste rezultate za istog farmaceuta. Prvi pacijent bira farmaceuta i započinje transakciju. U međuvremenu i drugi pacijent bira istog tog farmaceuta i on započinje transakciju za zakazivanje savetovanja. Oba pacijenta uspešno zakazuju savetovanje kod istog farmaceuta u preklapajućem terminu što dovodi aplikaciju u nekonzistentno stanje jer farmaceut ne može imati više od jednog savetovanja u bilo kom trenutku.

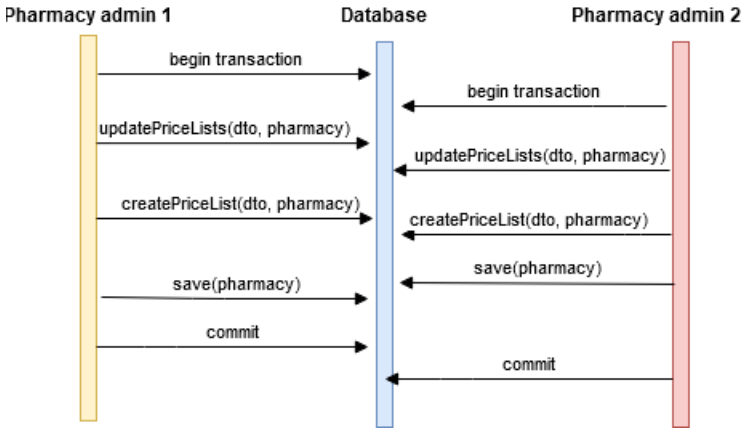


Rešenje: Strategija primenjena na rešavanje ove konfliktne situacije je pesimističko zaključavanje objekta klase Pharmacist koji se dobavlja u metodi za zakazivanje savetovanja. U PharmacistRepository dodata je metoda findOneById(long id) sa anotacijom @Lock(LockModeType.PESSIMISTIC_READ). Efekat koji ona ima je da se objekat dobavljen pomoću ove metode ne može čitati sve dok postoji druga transakcija koja je uzela taj objekat za sebe. Posledično, kada prvi pacijent započne transakciju za zakazivanje savetovanja u nekom terminu kod određenog farmaceuta, nijedan drugi pacijent to ne može uraditi za istog farmaceuta, u bilo kom terminu. Kako bi se izbegla određena nepotrebna odbijanja zakazivanja kod istog farmaceuta (kada se pokuša zakazati kod istog farmaceuta, ali u terminu koji se ne preklapa) na metodu findOneById(long id) PharmacistRepository-ja dodata je i anotacija @QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="2000")}) koja omogućava da druga transakcija ne bude odmah odbijena, već da sačeka 2 sekunde kada želi da dobavi farmaceuta i da ga uspešno dobavi ukoliko je on u međuvremenu postao slobodan, tako da onda može da se izvrši provera da li je taj farmaceut i dalje slobodan u željenom terminu.

3. *Više administratora iste apoteke ne može u isto vreme definisati cenovnik za isti datum*

Opis konfliktne situacije: Svaki administrator apoteke ima mogućnost da definiše cenovnik koji važi od određenog datuma tako što će selektovati listu lekova ili pregled/savetovanje i za odabrane stavke postaviti cenu koja će važiti od tog datuma. Ukoliko za odabrani datum već postoji cenovnik, on se izmenjuje na dva načina. Ukoliko u postojećem cenovniku ne postoji stavka koja je odabrana u novom cenovniku, ona će se dodati u cenovnik, dok ukoliko ta stavka već postoji, njena cena će biti izmenjena i postavljena na novu cenu. Konflikt nastaje ukoliko dva ili više administratora iste apoteke kreiraju cenovnik cenovnik za isti datum i ukoliko za taj datum ne postoji već kreiran cenovnik.

Crtež: Prvi administrator šalje cenovnik i započinje transakciju. Vrši se provera da li cenovnik za taj datum već postoji, i s obzirom da ne, poziva se metoda createPriceList(priceList, pharmacy). Pre komita prve transakcije, drugi administrator takođe šalje cenovnik za isti datum i započinje transakciju. Pošto izmene prve transakcije još uvek nisu komitovane, dobija se odgovor da cenovnik za taj datum ne postoji i poziva se metoda createPriceList(priceList, pharmacy). Ovakva sekvenca događaja dovodi do toga da postoje dva cenovnika za isti datum bez ikakve informacije o tome koji je validan i da zajedničke stavke u oba cenovnika imaju dve različite cene.



Rešenje: Kako bi se izbegla konfliktna situacija oko kreiranja cenovnika, primenjena je strategija optimističnog zaključavanja tako što je u klasu Pharmacy dodato polje long version sa anotacijom @Version koje će čuvati broj izmena određenog objekta apoteke. Kada administrator pokuša da izmeni cenovnik u problematičnoj opisanog situaciji, proveriće se da li je njegova verzija objekta apoteka ista kao i njena trenutna verzija u sistemu i ukoliko jeste, izvršiće se kreiranje cenovnika. Ukoliko se ipak verzija razlikuje, administrator će biti obavešten da je došlo do izmena i moći će ponovo da pokuša da izmeni cenovnik. Pošto je u međuvremenu cenovnik za odabrani datum već kreiran, kada administrator drugi put pošalje cenovnik, neće biti pozvana metoda createPriceList(priceList, pharmacy), već metoda updatePriceList(priceList, pharmacy), čime je konflikt razrešen.