Konfliktne situacije – Stdent 4
Djordjije Kundacina

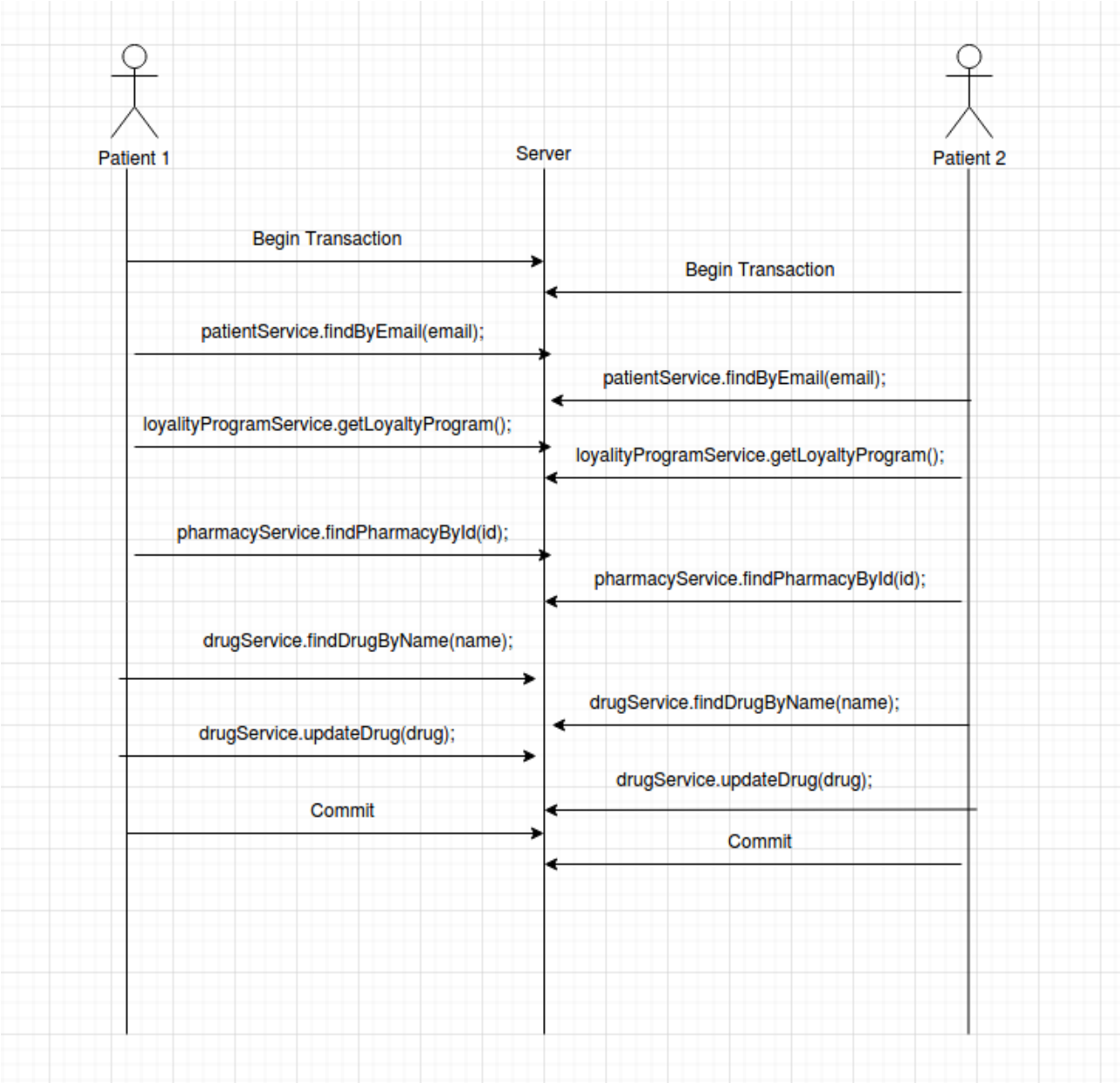
1. Prilikom izdavanja eRecepta se izdaju ili svi ili nijedan lijek I stanje lijeka u apoteci se azurira

Opis konfliktne situacije: Pacijent uz pomoc eRecepta moze da rezervise lijek, unosenjem qr koda, nakon cega bira jednu od ponudjenih apoteka koje mu sistem ponudi. Apoteke koje mu sistem nudi moraju imati na stanju sve lijekove. Pacijentu sistem nudi samo apoteke koje trenutno na stanju imaju lijek I dovoljnu kolicinu. Sto znaci da je obezbedjeno da pacijent izabere apoteku koja ima sve preduslove da izda eRecept. U trenutku kada dva pacijenta, pokusaju da rezervisu iste lekove iz istih apoteka, dolazi do konfliktne situacije. Pacijentu mogu da se prikazu apoteke koje vise nemaju te lijekove na stanju. Tj. Moze da se dogodi da dvije transakcije procitaju trenutno stanje lijekova u nekoj apoteci, ali da u medjuvremenu prva transakcije update to stanje, dok je druga transakcija ima stanje prije update.

Resenje: Ovaj problem se rijesava uz pomoc dodavanje anotacije iznad metode u ERecipeServiceImpl.

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public ERecipe addERecipe(ERecipeFromQrCodeDTO eRecipeFromQrCodeDTO)
```

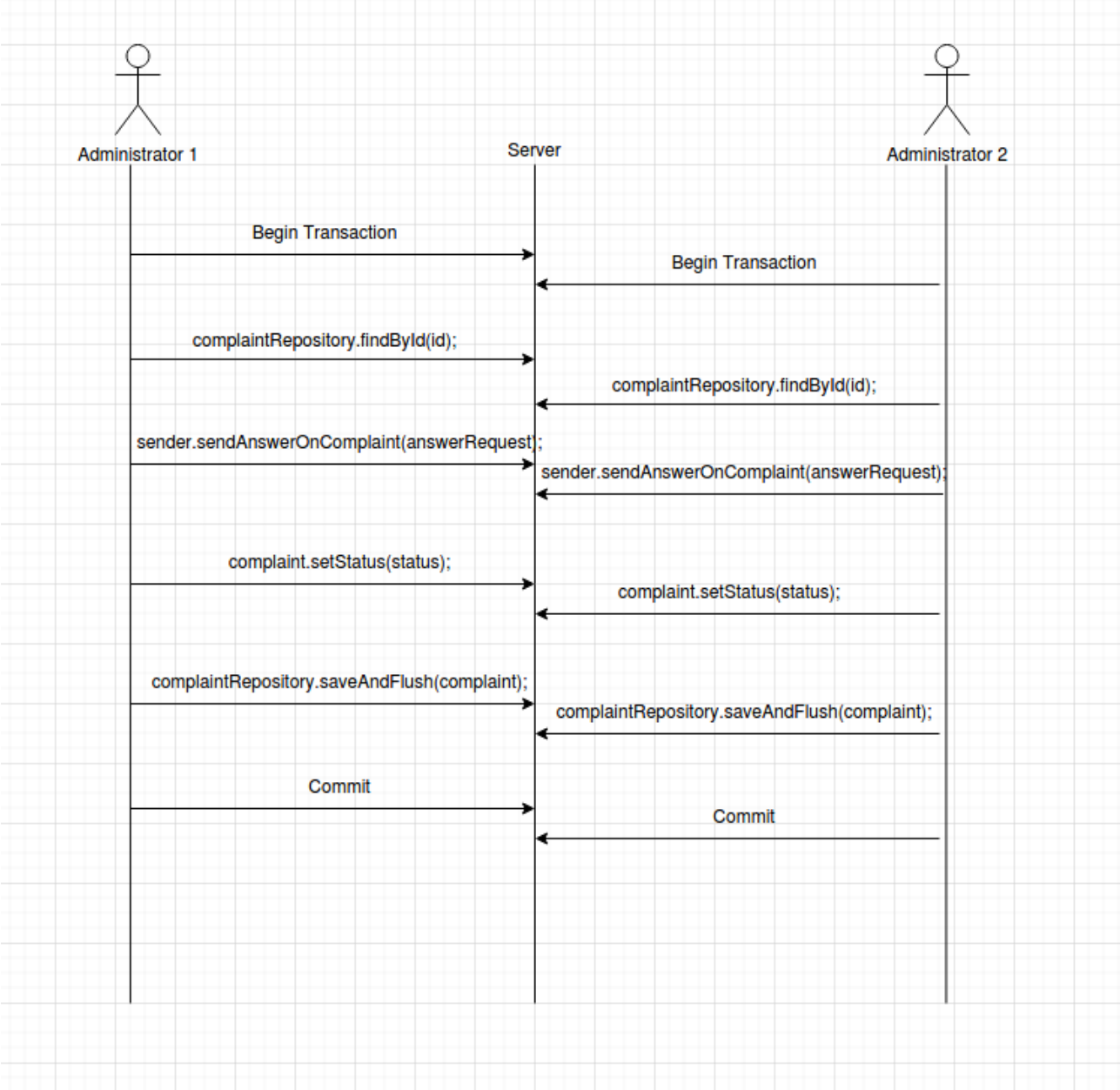
Anotacija omogucava kreiranje nove transakcije, sa readOnly omogucava izmjene podtaka, a sa Propagation.REQUIRES_NEW se omogucava kreiranje novih transakcija iako neke vec postoje.
Takodje anotaciju @Transactional sam dodao iznad svih metoda drugih servisa koji se pozivaju unutar funcije addERpecipe. Na ovaj nacin sve akcije budu izvrsene ili ne bude nijedna.



2.Na jednu zalbu moze da odgovori samo jedan administrator sistema

*<u>Opis konfliktne situacije:</u> Pacijenti mogu da unesu zalbu za dermatologa, farmaceuta ili apoteku. Administrator sistema moze da pregleda sve zalbe koje su napisane I u slobodnoj formi moze da unese odgovor na zalbu.Odgovor se salje na server, gdje server salje meil pacijentu, a status zalbe se stablja na **answered**. Problem nastaje kada dva administratora sistema posalju zahtjev za slanje odgovora u isto vrijeme, tada nastaje konfliktna situacija.*

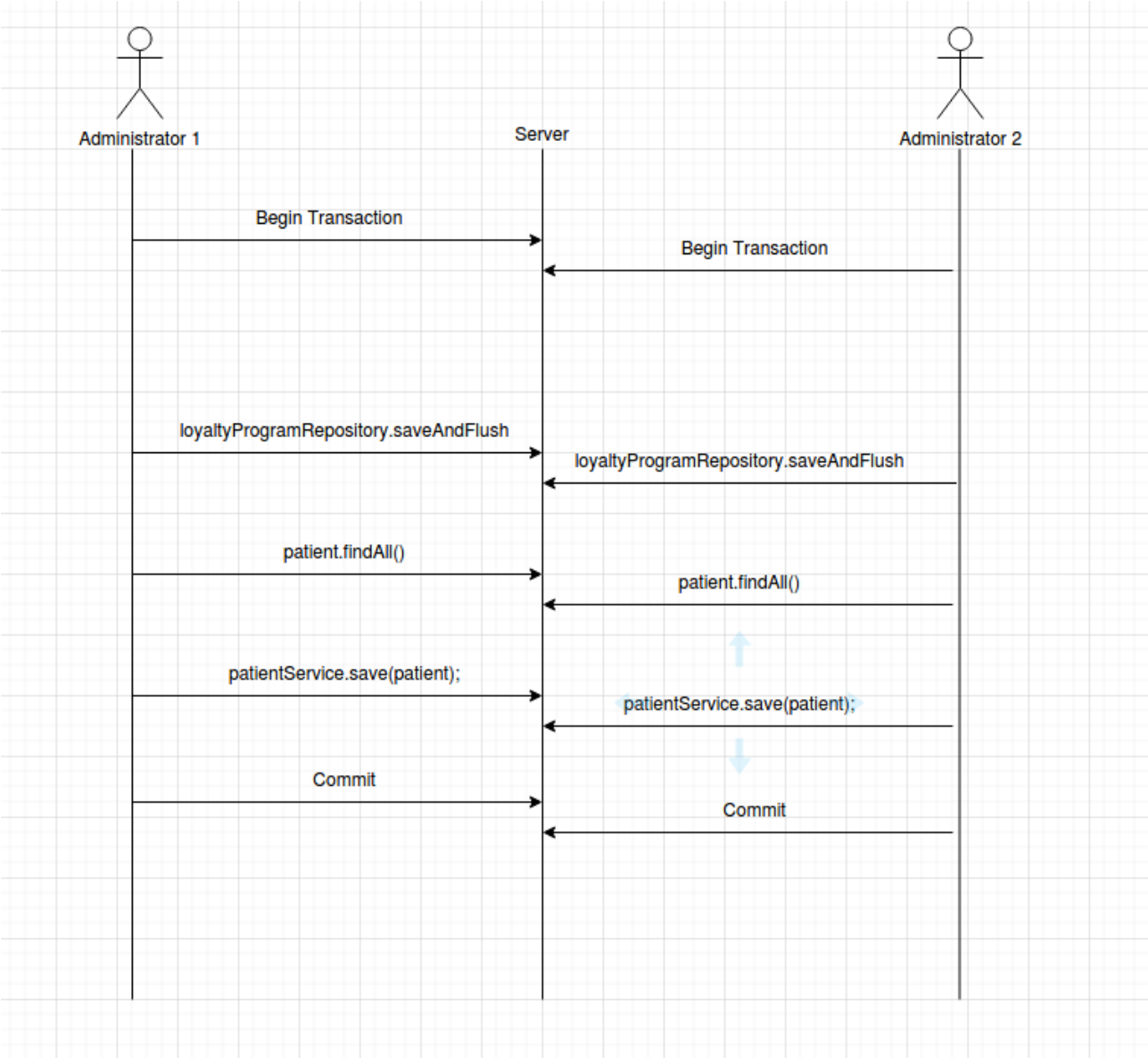
<u>Resenje:</u> Kako bi se zabranio pristup drugom administratoru, problem je rijesen pomocu optimistic locking-a. U klasu Complaint je ubaceno novo polje version sa anotacijom @Version. Pomocu @Version daje se verzija zalbi koju dobavljamo iz baze. Kada prvi administrator pristupi zalbi na koju zeli da da odgovor, vrijednost polja <u>version</u> se inkrementira. Tako da za svaki sledeci pokusaj pristupa zalbi, izvrsice se poredjenje verzija I one nece biti iste, sto znaci da je prvi administrator "zakljucao" zalbu I prilikom pokusaja cuvanja update-ovane zalbe prijavice se ObjectOptimisticLockingFailureException.



3. Loyalty program moze da mijenja samo jedan administratora

Opis konfliktne situacije: Loyality program daje odredjene pogodnosti korisnicima koji pripadaju odredjenoj kategoriji. Promjene loyaliti programa vrse administratori cijelog sistema. Mogu da vide trenutne granice za kategorije, kao i popust i broj bodova za svaki pregled i savjetovanje. Kada se update loyalty program potrebno je update-ovati kategoriju kod svih pacijenata.
Problem nastaje kada vise administratora mijenja podatke vezane za loyalty program i tada se javlja konflikt situacija.

Resenje: Kako bi se zabranio pristup drugom administratoru, problem je rijesen pomocu optimistic locking-a. U klasu LoyaltyProgram je ubaceno novo polje version sa anotacijom @Version. Pomocu @Version daje se verzija loyalty programu koju dobavljamo iz baze. Kada prvi administrator prisupi loyalty programu i krene da ga update, vrijednost polja version se inkrementira. Tako da za svaki sledeci pokusaj pristupa loyalty programu, izvrsice se poredjenje verzija I one nece biti iste, sto znaci da je prvi administrator "zakljucao" loyalty program I prilikom pokusaja cuvanja update-ovanog loyalty programa prijavice se ObjectOptimisticLockingFailureException.


Administrator 1
Server
Administrator 2
Begin Transaction
Begin Transaction
loyaltyProgramRepository.saveAndFlush
loyaltyProgramRepository.saveAndFlush
patient.findAll()
patient.findAll()
patientService.save(patient);
patientService.save(patient);
Commit
Commit