Konkurentnost - Student 4 – Veljko Tošić - SW55-2018

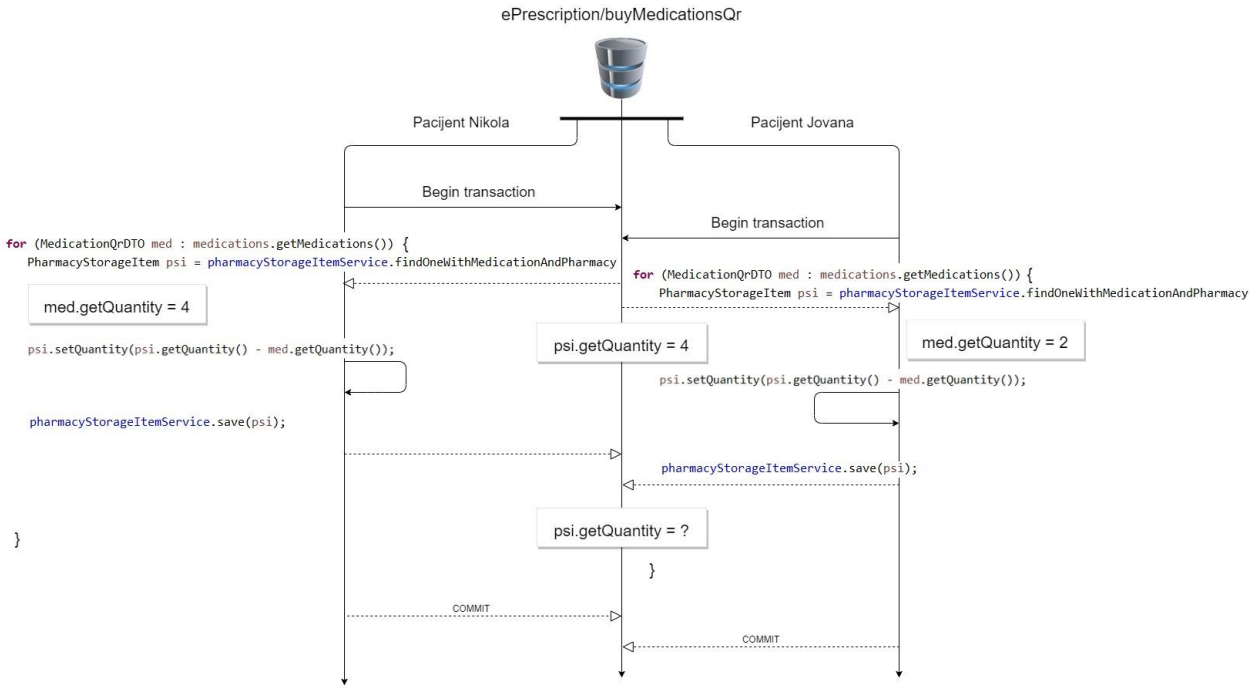
Rešavane konfliktne situacije:

1. Prilikom izdavanja eRecepta se izdaju ili svi ili ni jedan lek i stanje leka u apoteci se ažurira
2. Na jednu žalbu može da odgovori samo jedan administrator sistema
3. DODATNA SITUACIJA: Istovremena izmena loyalty programa od strane dva admina

Rešenja:

1. **Prilikom izdavanja eRecepta se izdaju ili svi ili ni jedan lek i stanje leka u apoteci se ažurira**

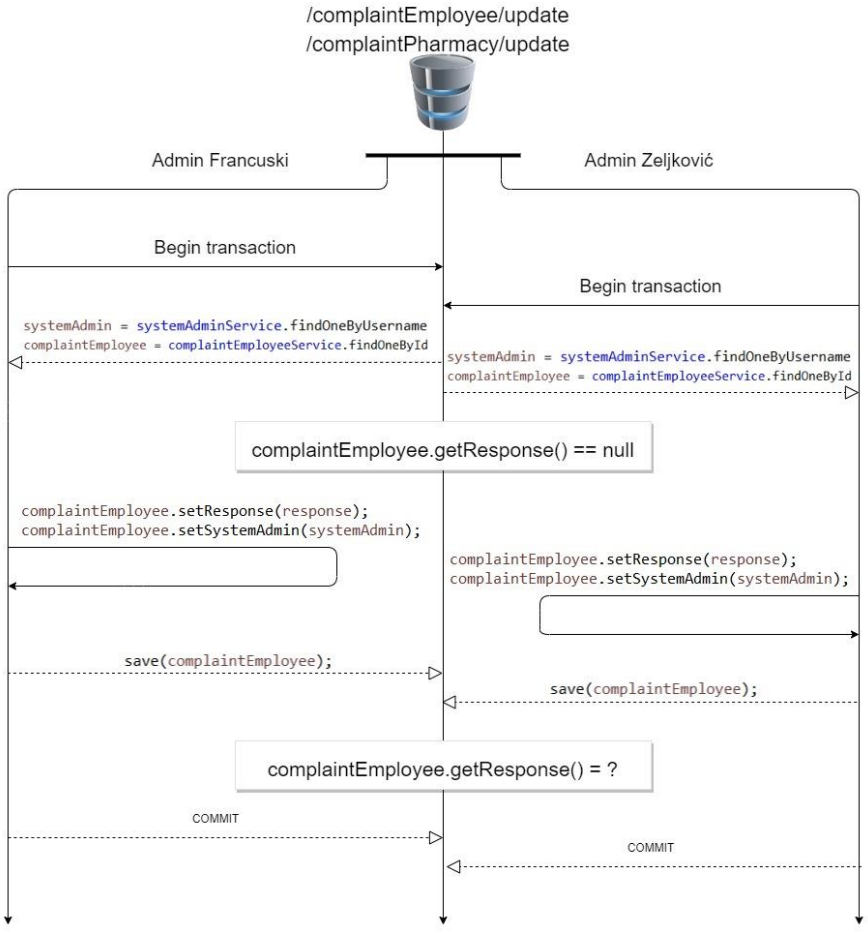
   Prilikom izdavanja leka preko eRecepta postoji šansa da dva korisnika u istom vremenskom trenutku žele da kupe isti lek sa određenom količinom, gde je zbir tih količina veći od ukupne količine leka u apoteci. Ovo dovodi do konfliktne situacije gde će se izdati oba eRecepta, a stanje u bazi biće nekonzistentno. Ukoliko imamo više lekova koje je potrebno izdati preko eRecepta može se desiti da opisan scenario desi za jedan od lekova, te je potrebno obezbediti vraćanje baze u prethodno stanje ukoliko je bilo izmena na nekom od ostalih lekova eRecepta. Primer: Dva pacijenta su ulogovana na sajt i oba u istom trenutku potvrđuju da žele da kupe lek "Probiotik". Količina leka kod prvog pacijenta je 2, kod drugog 4, a na stanju ima 4 leka. Ukoliko se trenuci kupovine dovoljno precizno poklope može se desiti da oba pacijenta kupe lek, iako to ne bi smeli, jer je ukupna količina leka na stanju 4.



Za rešavanje ovog problema korišćeno je pesimističko zaključavanje leka na nivou apoteke (PharmacyStorageItem). Pošto se izdavanje leka preko eRecepta vrši za više lekova, ovo zaključavanje će se vršiti za svaki lek koji se izdaje. Na taj način, ukoliko imamo više eRecepta koji potencijalno mogu da ugroze konzistentnost u bazi neće moći da pristupe istom leku u istom vremenskom periodu. Potrebno je napomenuti da je ovaj pristup moguće rešiti i optimističkim zaključavanjem, ali je koleginica (student 1) tokom rešavanja svog prvog zadatka, odabrala pesimističko zaključavanje jer je isti smatrala za adekvatan. Zaključavanje se vrši nad metodom repozitorijuma PharmacyStorageItemRepository dodavanjem anotacije **@Lock(LockModeType.PESSIMISTIC_WRITE)**. Dodatno, dodata je i anotacija **@Transactional** na metodu servisa. Ova anotacija obezbeđuje vraćanje u prethodno stanje svakog izmenjenog leka u procesu ukoliko dođe do greške.

2. **Na jednu žalbu može da odgovori samo jedan administrator sistema**

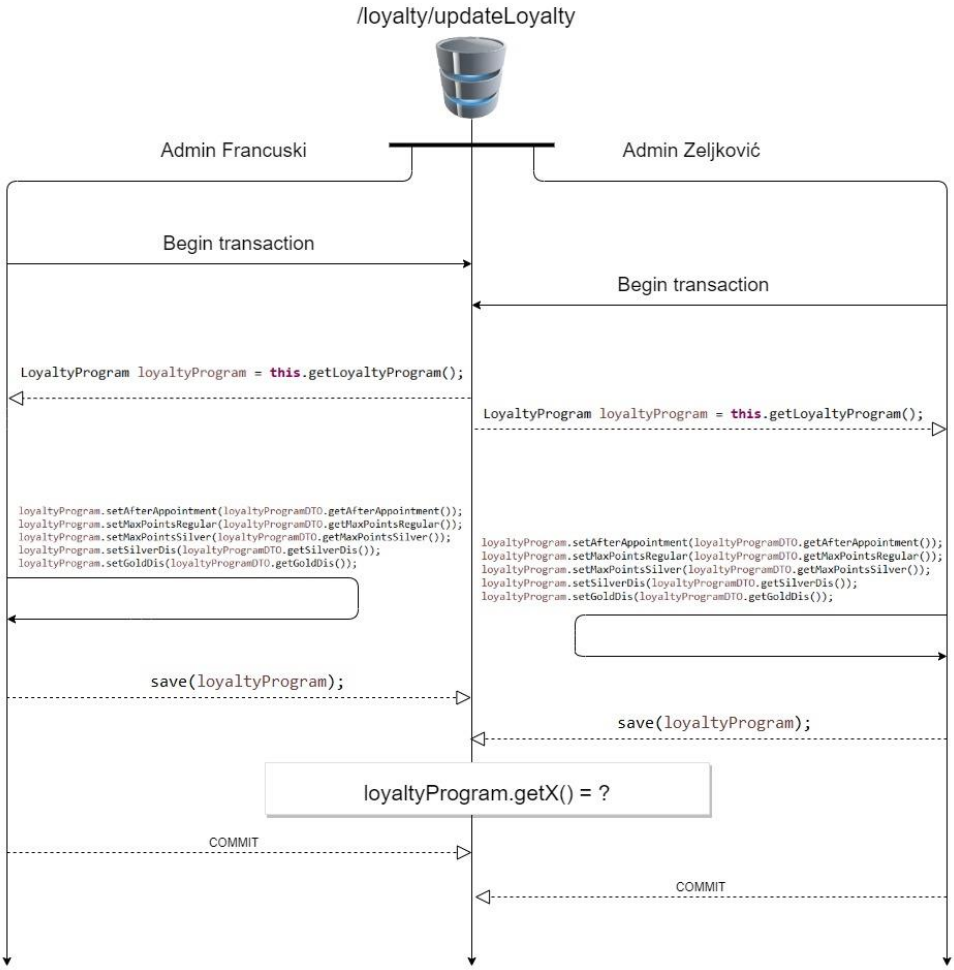
Kada neki pacijent napiše žalbu na nju može da odgovori svaki administrator sistema. Problem nastaje kada barem dva administratora odgovore na žalbu u istom trenutku. Ukoliko dva administratora pošalju odgovor u istom trenutku potrebno je ažurirati žalbu odgovorom samo administratora čiji je odgovor prvi došao do baze.



Za rešavanje ovog problema korišćeno je optimistička varijanta zaključavanja sa verzionisanjem žalbe. U apstraktnoj klasi koja predstavlja žalbu (**Complaint**) dodat je atribut **Long version**, anotiran anotacijom **@Version**. Ova anotacija obezbeđuje da se pri svakom čuvanju entiteta žalbe u bazi proveri verzija i ukoliko je ta verzija jednaka verziji objekta koji se čuva, čuvanje će proći, u suprotnom biće bačen **ObjectOptimisticLockingFailureException** i vrši se rollback. Rollback obezbeđuje **@Transactional** anotacija na metodi servisa koja služi za odgovaranje na žalbu.

3. DODATNA SITUACIJA: Istovremena izmena loyalty programa od strane dva admina

Loyalty program je entitet koji postoji u bazi koji može da menja svaki administrator sistema. Do problema dolazi kada dva admina istovremeno žele da redefinišu loyalty program. Svaki admin unosi nove vrednosti za sva polja i šalje zahtev. Da ne bi doslo do nekonzistentnog stanja u bazi potrebno je obezbediti da loyalty program ažurira samo administrator sistema čiji je zahtev prvi došao do baze.



Za rešavanje ove konfliktne situacije je korišćeno optimističko zaključavanje verzionisanjem loyalty programa. U klasi **LoyaltyProgram** dodat je atribut **Long version**, anotiran anotacijom @**Version**. Na ovaj način je obezbeđeno da samo admin koji je prvi poslao zahtev ažurira vrednosti loyalty programa, jer on ima odgovarajuću verziju resursa, dok će ostali biti prekinuti i dobiće **ObjectOptimisticLockingFailureException**. Metoda servisa koja menja loyalty program anotirana anotacijom @**Transactional** koja obezbeđuje povratak u prethodno stanje ukoliko dođe do greške.