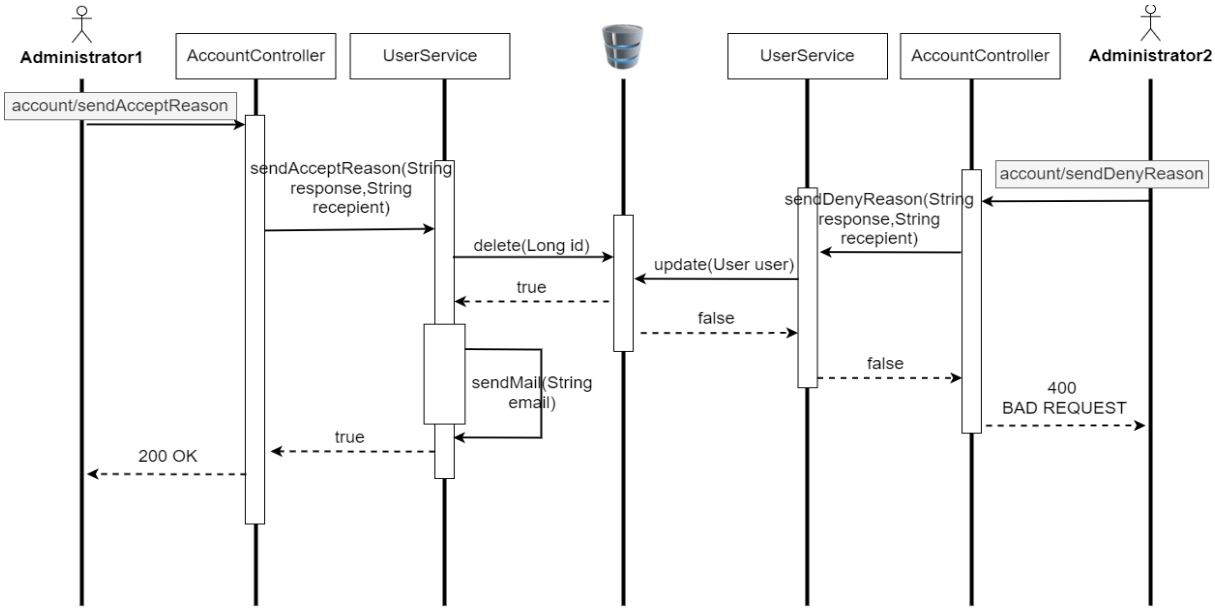
KONKURENTNI PRISTUP BAZI – Student 3

1. Odgovaranje na zahtev za brisanje naloga

Problem nastaje ukoliko dva različita administratora u isto vreme pokušaju da odgovore na isti zahtev za brisanje. Na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema, a odgovor administratora se šalje na mejl. Moguće situacije su da oba administratora žele da odobre, odbiju ili da jedan odobri a drugi odbije. Kako bismo izbegli da klijentu u isto vreme stignu dva kontradiktorna odgovora potrebno je rešiti ovu konfilktnu situaciju. Na slici 1 je prikazana situacija kada jedan administrator prihvati zahtev za brisanje naloga, a drugi ga nekoliko sekundi kasnije odbije.

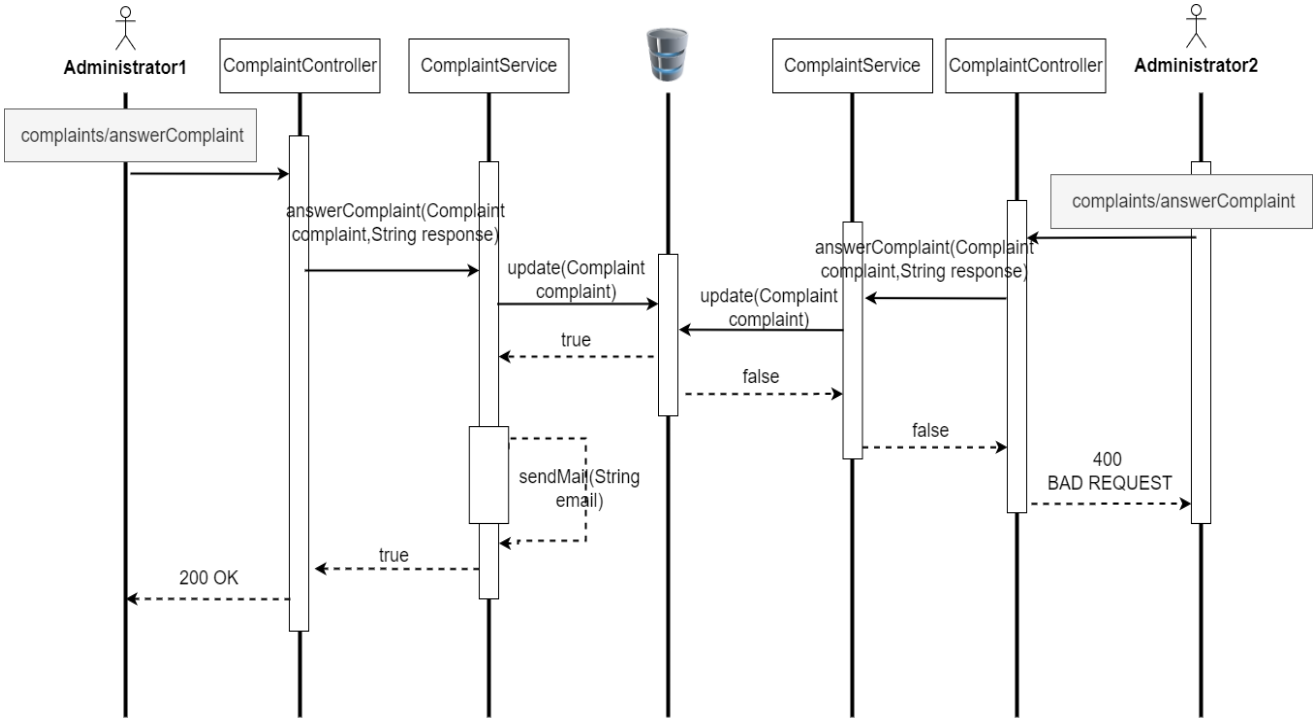


Slika 1 - Odgovaranje na zahtev za brisanje naloga.

Problem je rešen upotrebom optimističkog zaključavanja. U klasu User dodato je polje version sa anotacijom @Version. Ona nam omogućava da izvršimo verzionisanje torke u bazi. Za odobravanje zahteva za brisanje naloga korišćena je metoda sendAcceptReason, dok je za odbijanje korisćena metoda sendDenyReason. Obe metode nalaze se u servisu UserService i označene su kao transakcione anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE, atribut readonly postavljen na false, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje. Takođe izabranim nivoom izolacije rešena su četiri glavna problema konkurentnog pristupa bazi.

2. Odgovaranje na žalbu klijenta

Problem nastaje ukoliko dva različita administratora u isto vreme pokušaju da odgovore na istu žalbu klijenta na vlasnika ili entitet koji vlasnik poseduje. Na žalbu može da odgovori samo jedan administrator sistema, a odgovor administratora se šalje na mejl klijentu i vlasniku. Potrebno je obezbediti da dva administratora ne mogu u isto vreme da odgovore na istu žalbu jer će klijentu i vlasniku u toj situaciji stići dva potencijalno različita odgovora. Na slici 2 prikazana je konfliktna situacija.

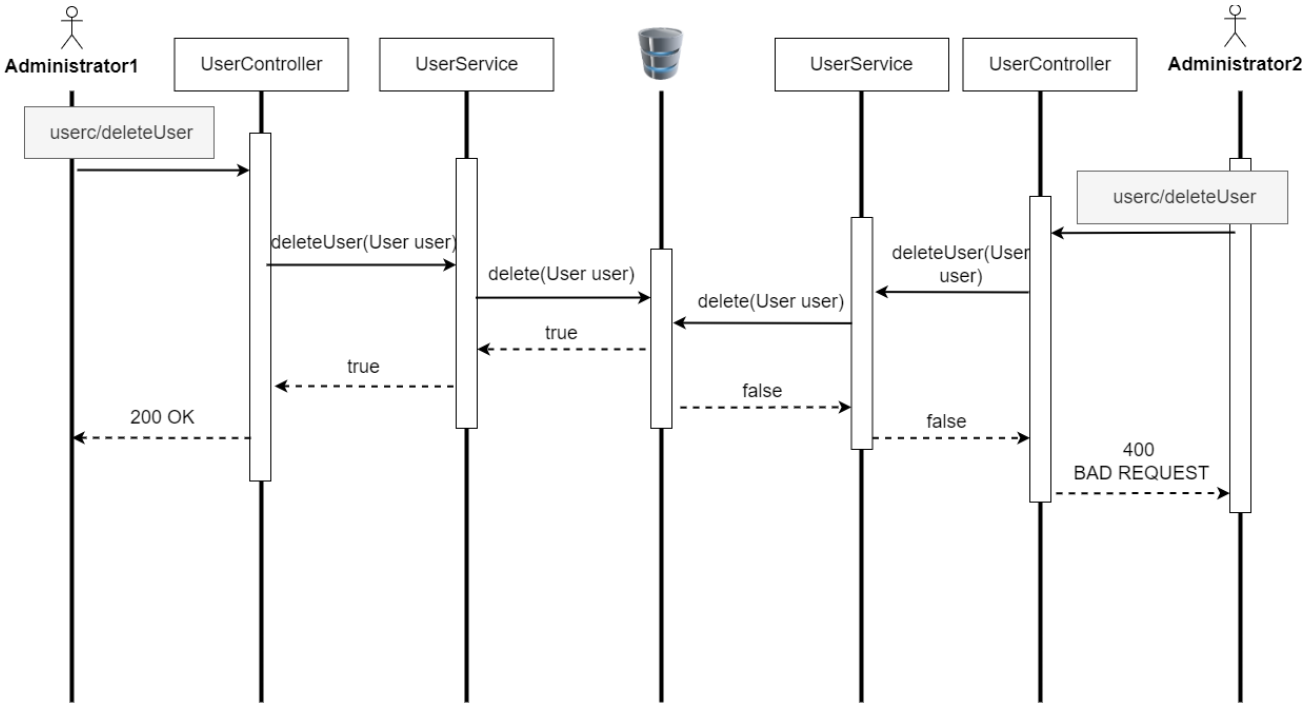


Slika 2- Odgovaranje na žalbu klijenta

Problem je kao i u prvom slučaju rešen upotrebom optimističkog zaključavanja. U klasu Complaint dodato je polje version sa anotacijom @Version. Ona nam omogućava da izvršimo verzionisanje torke u bazi. Za odgovaranje na žalbu klijenta korišćena je metoda answerComplaint koja je označena kao transakciona anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE, atribut readonly postavljen na false, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje. Takođe izabranim nivoom izolacije rešena su četiri glavna problema konkurentnog pristupa bazi.

3. Brisanje naloga korisnika

Svaki administrator sistema ima pravo da nazavisno od žalbi, ocena korisnika i zahteva za brisanje naloga obrise bilo kog korisnika. Problem nastaje kada dva administratora pokusaju da obrišu istog korisnika u isto vreme. Potrebno je onemoguciti da oba administratora izvrše brisanje istog korisnika. Primer ove konfliktne situacije nalazi se na slici 3.



Slika 3- Brisanje naloga korisnika

Problem je kao i u prva dva slučaja rešen upotrebom optimističkog zaključavanja. U klasu User dodato je polje version sa anotacijom @Version. Ona nam omogućava da izvršimo verzionisanje torke u bazi. Za brisanje naloga korišćena je metoda deleteUser User servisa koja je označena kao transakciona anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE, atribut readonly postavljen na false jer vršimo izmenu u bazi, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje. Takođe izabranim nivoom izolacije rešena su četiri glavna problema konkurentnog pristupa bazi.