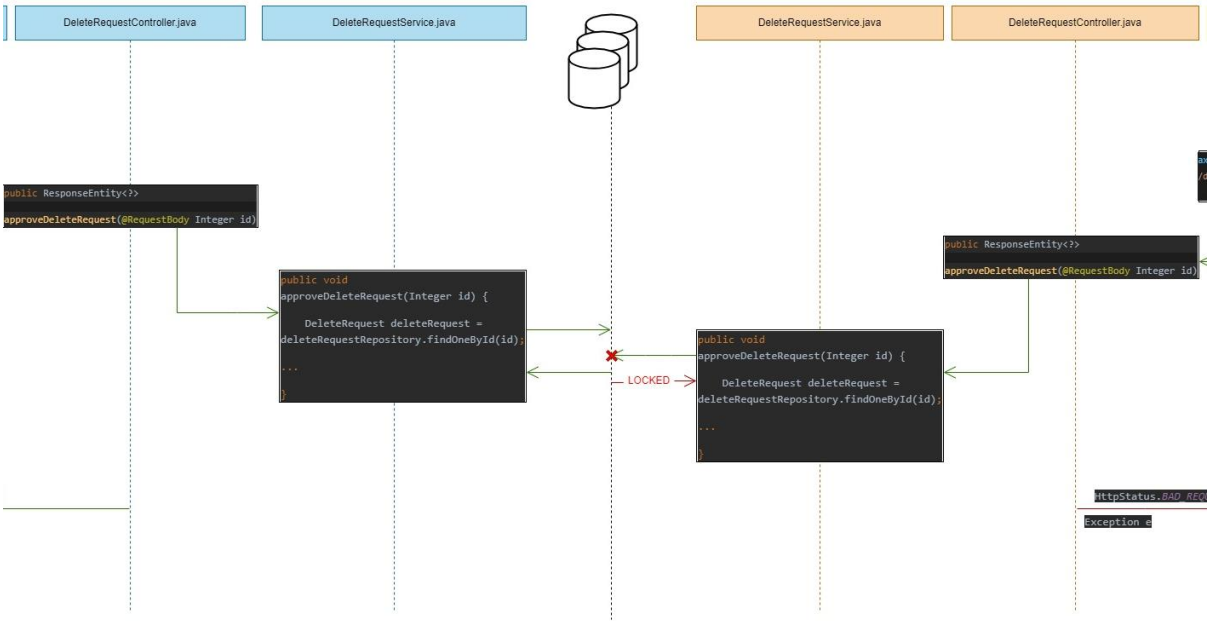
4.4 Konkurentan pristup resursima u bazi

Rešene konfliktne situacije:

Na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema

Opis situacije: Administratori odgovaraju na zahtev korisnika za brisanje naloga. Administrator može da potvrdi ili odbije zahtev. U oba slučaja, šalje se email korisniku o administratorovoj odluci. Potrebno je obezbediti da dva administratora ne mogu u isto vreme da odgovore na jedan isti zahtev kako ne bi došlo do različitog rukovanja tj. različitog odgovora na zahtev od strane njih.

Isečak crteža konfliktne situacije (cela slika u folderu ./Transakcije/Student3/ Student3-1.jpg):

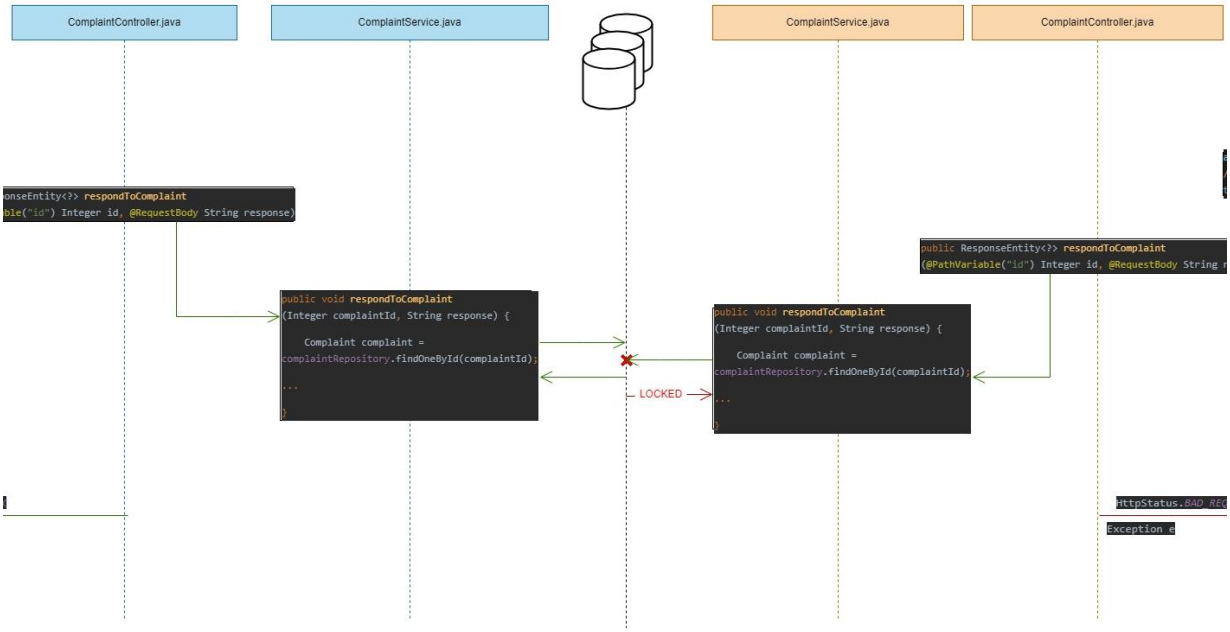


Rešenje: Postupak se nalazi u klasama *DeleteRequestController.java* (rukovanje izuzetkom), *DeleteRequestService.java* (poziva metodu findOneById(Integer id) repozitorijuma) i *DeleteRequestRepository.java* (gde je odrađeno pesimističko zaključavanje na nivou metode findOneById(Integer id)). Kako bi se izbegla konfliktna situacija, korišćeno je pesimističko zaključavanje jer se nakon odobravanja/odbijanja zahteva za brisanje entitet *DeleteRequest* odmah briše iz baze, pa je jedini način obezbeđivanje tog entiteta dok postoji, njegovo zaključavanje i zabrana da mu bilo ko drugi pristupi ako je zaključan od strane nekog drugog klijenta. U tom slučaju, dogodiće se izuzetak i administrator koji je izazvao konfliktnu situaciju biva obavešten o tome.

Na jednu žalbu može da odgovori samo jedan administrator sistema

Opis situacije: Administratori odgovaraju na žalbe korisnika na entitet ili vlasnika/instruktora. Nakon odgovora na žalbu, administratorov odgovor se šalje i klijentu i oglašivaču na email i žalba se briše iz sistema jer nije više relevantna. Potrebno je obezbediti da dva administratora ne mogu u isto vreme da odgovore na jednu žalbu kako ne bi došlo do različitog odgovora na žalbu od strane njih.

Isečak crteža konfliktne situacije (cela slika u folderu ./Transakcije/Student3/ Student3-2.jpg):

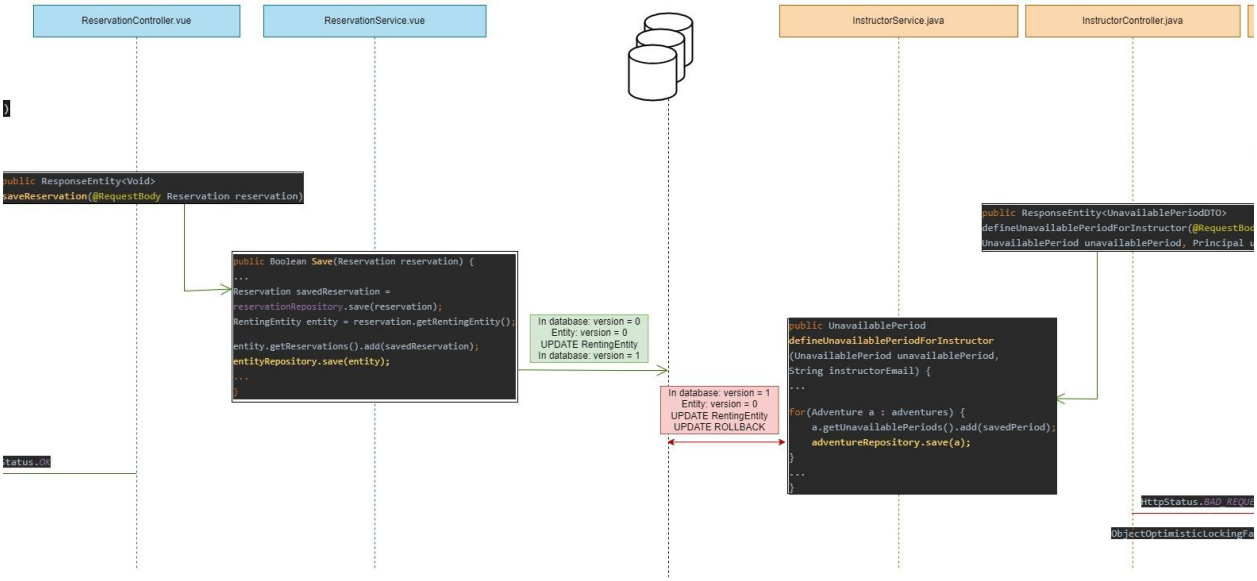


Rešenje: Postupak se nalazi u klasama *ComplaintController.java* (rukovanje izuzetkom), *ComplaintService.java* (poziva metodu findOneById(Integer id) repozitorijuma) i *ComplaintRepository.java* (gde je odrađeno pesimističko zaključavanje na nivou metode findOneById(Integer id)). Kako bi se izbegla konfliktna situacija, kao i u prethodnom primeru, korišćeno je pesimističko zaključavanje jer se nakon odgovora na žalbu entitet *Complaint* odmah briše iz baze, pa je jedini način obezbeđivanje tog entiteta dok postoji, njegovo zaključavanje i zabrana da mu bilo ko drugi pristupi ako je zaključan od strane nekog drugog klijenta. U tom slučaju, dogodiće se izuzetak i administrator koji je izazvao konfliktnu situaciju biva obavešten o tome.

Instruktor definiše period nedostupnosti u isto vreme kada klijent rezerviše njegov entitet

Opis situacije: Može se desiti da instruktor pecanja definiše period svoje nedostupnosti u istom trenutku kada klijent napravi rezervaciju njegove avanture. Ako se period nedostupnosti i vreme rezervacije poklope, dolazi do konfliktne situacije, koja ako se ne razreši, rezultira rezervisanom avanturom u vreme kada je instruktor nedostupan.

Isečak crteža konfliktne situacije (cela slika u folderu ./Transakcije/Student3/ Student3-3.jpg):



Rešenje: Postupak se nalazi u klasama *InstructorController.java* (rukovanje izuzetkom), *InstructorService.java* i *AdventureRepository.java* (metoda *defineUnavailablePeriodForInstructor(UnavailablePeriod period, String instructorEmail)* poziva metodu *save(adventure)* iz *AdventureRepository.java*). Kako bi se izbegla konfliktna situacija korišćeno je optimističko zaključavanje objekta *Adventure* (verzija je dodata u njegovu nadklasu *RentingEntity*). Pri pravljenju nove rezervacije, klijent čuva rezervaciju u objekat tipa *RentingEntity*, čime se update-uje verzija, tako da ukoliko instruktor pokuša da doda svoj period nedostupnosti (što podrazumeva i period nedostupnosti svakog njegovog entiteta tj. avanture) i sačuva entitet, ako se verzije ne poklapaju doći će do *ObjectOptimisticLockingFailureException*-a i biće mu onemogućeno da sačuva novi period nedostupnosti, o čemu će biti obavešten.