

4.4 Konkurentni pristup resursima u bazi

1. instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent

Instruktor ima mogućnost da za klijenta za koga traje rezervacija, napravi još jednu rezervaciju avanture u budućnosti.

Scenario: Instruktor na web sajtu za trenutnog klijenta, vrši ponovnu rezervaciju. Ukoliko ona zadovoljava sve provere na bekendu ona će biti sačuvana u bazu. Postoji šansa da u istom trenutku neki klijent napravi rezervaciju kod ovog Instruktor i pošalje zahtev na bekeend. Problem konflikt i dolazak do nekonzistentnog stanja se može desiti u periodu između provere uslova i samog perzistiranja prve nove avanture.

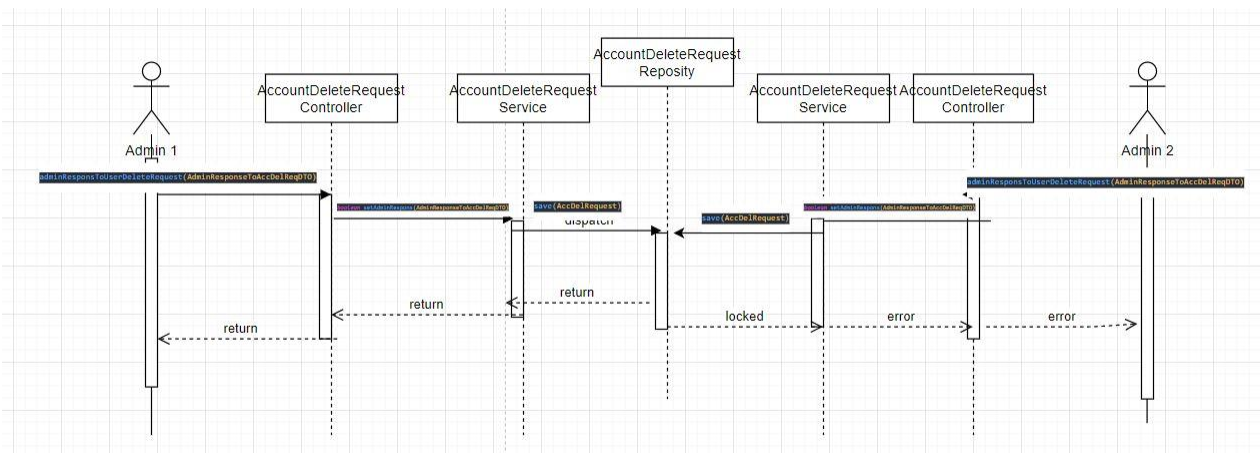
3. Na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema

Admin na sistemu ima mogućnost da gleda i odgovara na zahteve brisanja naloga.

Scenario: Imamo dva admina koja u isto vreme zele da odgovore na zahtev za brisanje naloga. Kad admin posalje odgovor na bekeend, vrši se validacija, proverava da li je zahtev u stanju PENDING, u suprotnom nije moguće odgovoriti na zahtev. Problem se javlja kada dva admina istovremeno pokušaju da odgovore na zahtev, tada će zahtev od onog sporijeg admina stići nešto kasnije od prvog ali ako dovoljno brzo stigne, tako da odgovor prvog admina još nije promenio stanje baze, javlja se problem. Zato zaključavamo resurs pesimistički.

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public boolean setAdminResponse(AdminResponseToAccDelReqDTO
adminResponseToAccDelReqDTO, String adminEmail)
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM AccountDeleteRequest r WHERE r.id=?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value =
"0")})
AccountDeleteRequest getDelReqById(int idRequest);
```

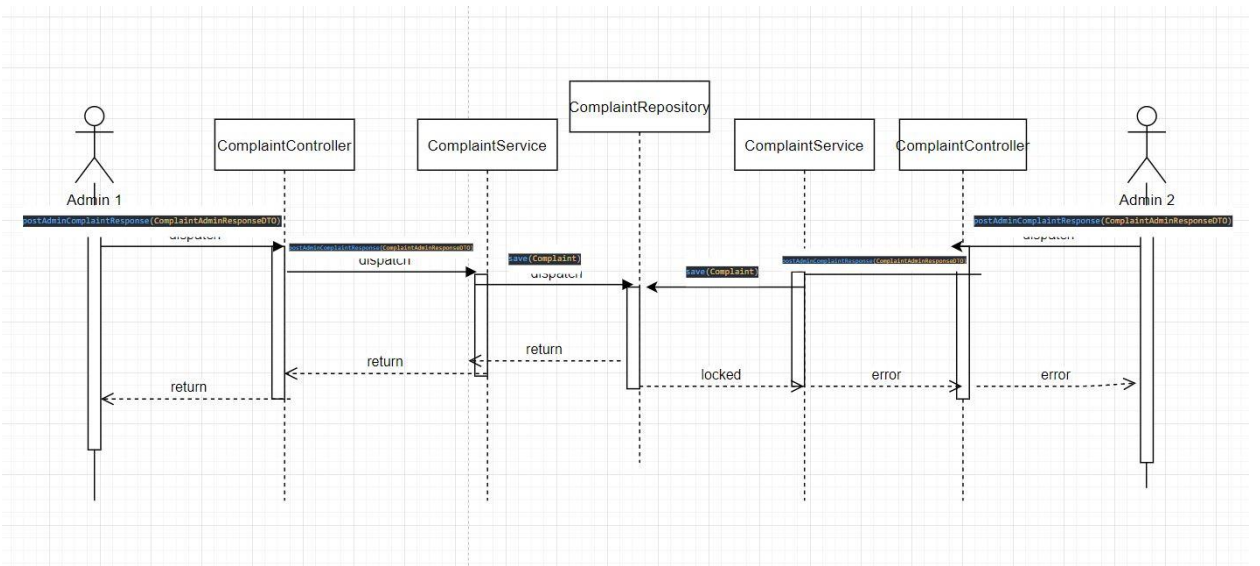


4. Na jednu žalbu može da odgovori samo jedan administrator sistema

Admin na sistemu ima mogućnost da gleda i odgovara na žalbe.

Scenario: Imamo dva Admina koja u isto vreme zele da odgovore na žalbu. Prilikom slanje adminovog odgovora na žalbu, Proveravamo da li je vec neki admin odgovorio na tu konkretnu žalbu. U Slučaju da žalba nije odgovorena, azurira se sa adminovim odgovorom i stanje prelazi iz stanja UNANSWERED u stanje ANSWERED, Problem se javlja kada se pojave ovakva dva zahteva koja su vremenski jako blizu, da druga žalba pristigne do provere u periodu dok je prva žalba izmedju validacije i promene stanja.

Problem koji se može dogoditi je da za jednu žalbu, klijentu i instruktoru dodju dva odgovora od dva admina.



Za resavanje ovog problema odlucio sam se za Pesimisticno zakljucavanja resursa.

```

@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM Complaint r WHERE r.id=?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value =
  
```

```
"0"))})
Complaint getComplaintById(int id);
```

U metodi postAdminComplaintResponse klase ComplaintService se poziva metoda getComplaintById(int)

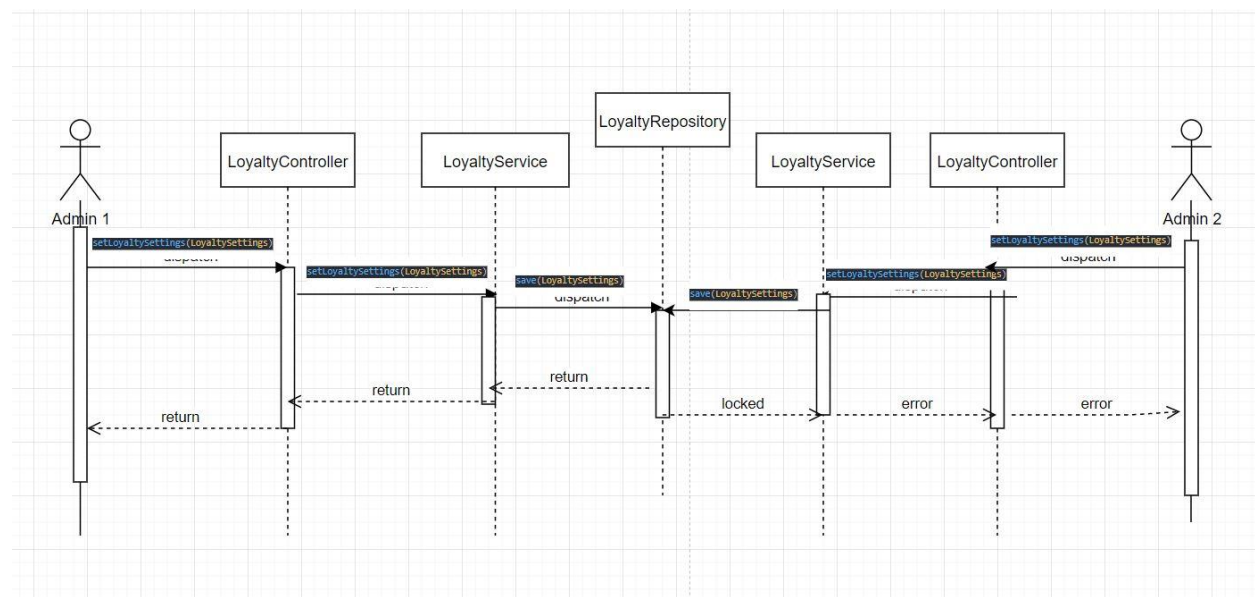
```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public Boolean postAdminComplaintResponse(ComplaintAdminResponseDTO
complaintAdminResponseDTO)
```

Na ovaj način sprecili smo štetno preplitanje, i izbegli smo nekonzistentnost do koje je sistem mogao doći.

5. Admin ne može da promeni Loyalty u isto vreme kad i drugi Admin

Admin na sistemu ima mogućnost da upravlja sistemskim Loyalty programom.

Scenario: Imamo dva Admina koja u isto vreme gledaju Loyalty settings i imaju želju da promene podatke. Prilikom slanja, Backend će proveriti validnost parametara loyaltySettings i krenuti u izmenu Eniteta u bazi.



U ovoj situaciji može da se pojavi štetno preplitanje prilikom primene novih parametara na korisnike sistema. Sledi deo koda koji se nalazi u funkciji setLoyaltySettings

```
public LoyaltySettings setLoyaltySettings(LoyaltySettings newLoyaltySettings)
```

```
if(updateUsers){
    instructorService.updateLoyaltyForAll(loyaltySettings);
    boatOwnerService.updateLoyaltyForAll(loyaltySettings);
    cottageOwnerService.updateLoyaltyForAll(loyaltySettings);
    clientService.updateLoyaltyForAll(loyaltySettings);
}
```

Za resavanje ovog problema odlucio sam se za Pesimisticno zakljucavanja resursa LoyaltySettings U LoyaltyRepository je odrađeno pesimističko zaključavanje getLoyaltySettings(int id) metode na nivou jednog reda u tabeli koji predstavlja entitet. Korišten je PESSIMISTIC_WRITE kao tip zaključavanja. Servis Loyalty Settings poziva metodu getLoyaltySettings.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM LoyaltySettings r WHERE r.id=?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value =
"0")})
LoyaltySettings getLoyaltySettings(int id);
```

Rastislav Kukucka RA129-2018