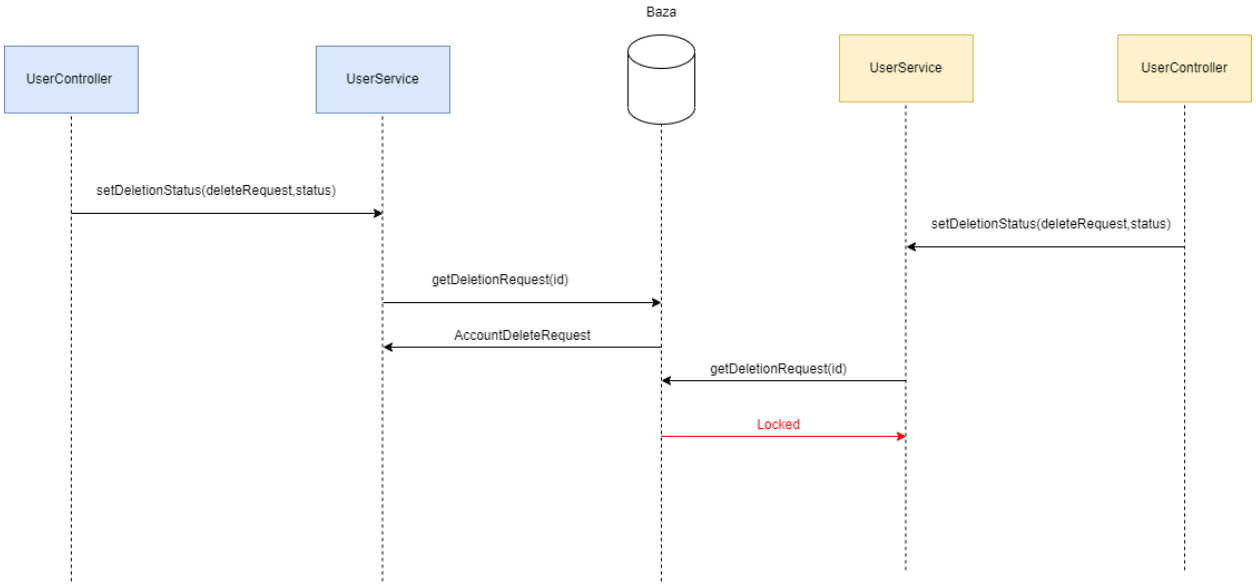
4.4 Konkurentni prisup resursima u bazi

1. *Na jedan zahtjev za brisanje naloga može da odgovori samo jedan administrator sistema*

Opis problema: Administrator sistema kao jednu od funkcionalnosti ima mogucnost uvida u zahtjeve za brisanje naloga na koje jos nije odgovoreno, kao i mogucnost da ih odbije / odobri. Problem nastaje kada dva administratora Mirko i Pera u istom ili preklapajućem periodu započnu odgovaranje na isti zahtjev. Ukoliko je Mirko odobrio zahtjev, a zahtjev se jos uvijek nije sacuvao u bazi i Pera u medjuvremenu proba da ga odbije to ce biti moguce. Nakon toga se cuva prvo Mirkov,pa Perin zahtjev I korisnik prvo dobija obavjestenje da mu je zahtjev odobren, a nakon toga odbijen sto nije konzinstentno. Rjesenje problema prikazano je na sljedecoj slici.



Slika 1. Rjesenje konfliktnog pristupa za brisanje naloga korisnika

Rjesenje problema:

Za rjesavanje ovog problema koristeno je pesimisticko zakljucavanje resursa. Kao sto je prikazano na slici prvi korisnik koji pristupi resursu u bazi moci ce da ga mijenja, dok ce se drugom vratiti PessimisticLockException. Implementacija je izvrsena tako sto je na nivou servisne klase dodata @Transactional anotacija.

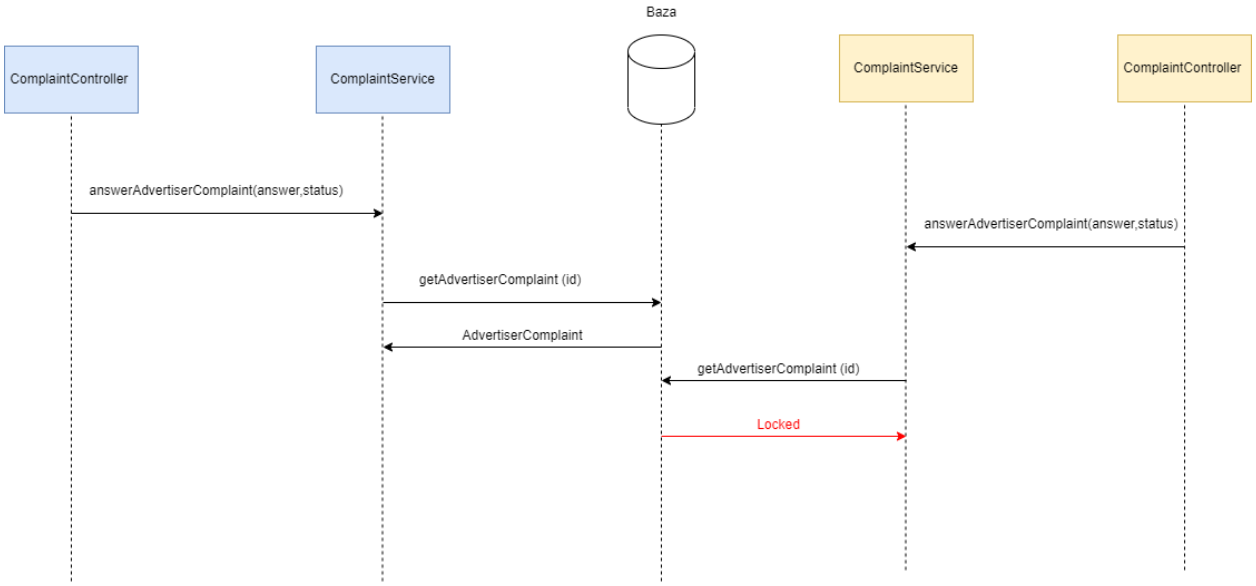
```
@Service
@Transactional
public class UserService  implements UserDetailsService
```

A iznad metode u repozitorijumu gdje se vrši dobavljanje zahtjeva za brisanje dodate su sljedece anotacije:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query(value = "SELECT a FROM AccountDeleteRequest a WHERE a.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
AccountDeleteRequest getDeletionRequest(@Param("id") Integer id);
```

2. *Na jednu žalbu može da odgovori najvise jedan administrator*

Opis problema: Na slicnom principu kao i prethodna stavka funkcionise odobravanje/odbijanje zalbi za oglasivaca i entitete. Administrator moze da vidi sve zalbe na koje nije odgovoreno. Problem nastaje kada dva administratora pokusaju da u isto vrijeme ili preklapajuce vrijeme odgovore na istu zalbu. Ako je prvi administrator odobrio zalbu, i ako se taj status jos uvijek nije sacuvao u bazu,a drugi administrator pokusa da odbije tu istu zalbu, to ce biti moguce. Rjesenje ovog problema prikazano je na slici:



Slika 2. Rjesenje konfliktnog pristupa za odgovaranje na zalbe korisnika

Rjesenje problema:

Za rjesavanje ovog problema koristeno je pesimisticko zaljucavanje, na taj nacin prvi put kad administrator pristupi odredjenom objektu on postaje nedostupan za sve ostale korisnike koji zele da ga dobave. Na slici je prikazana situacija vezana za odobravanje zalbi vezanih za oglasivace. Na istom principu se zasniva i odobravanje zalbi za entitete s tim da se kod njih u kontroleru poziva metoda answerEntityComplaint,a u servisu se dobavljanje objekta vrsi putem metode getEntityComplaint.

Implementacija je izvrsena tako sto je na nivou servisne klase stavljena anotacija @Transactional.

```
@Transactional
@Service
public class ComplaintService {
```

U okviru repozitorijuma iznad metode za dobavljanje objekta AdvertiserComplaint dodate su anotacije:

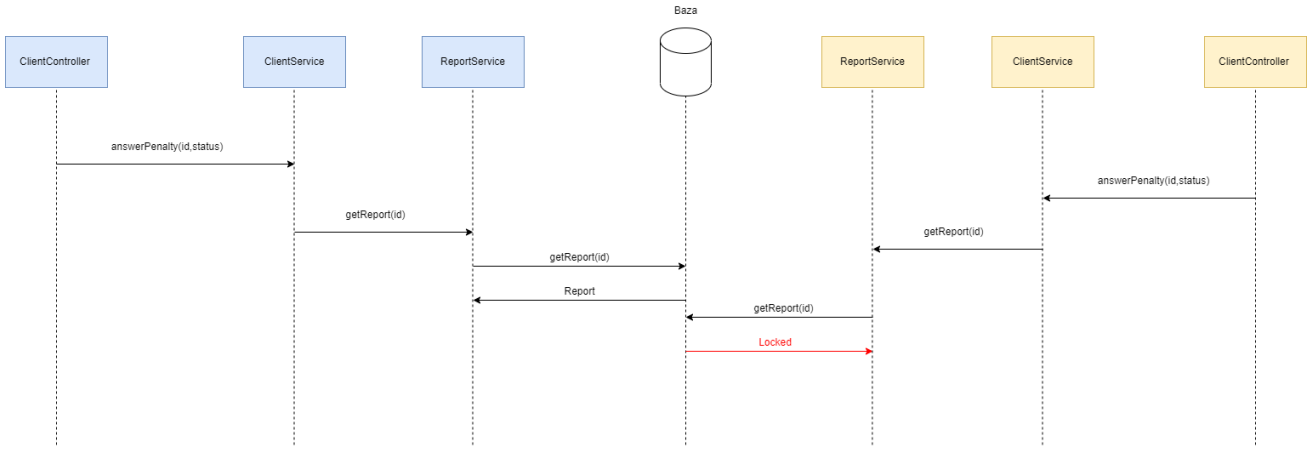
```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query(value = "SELECT a FROM AdvertiserComplaint a WHERE a.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
AdvertiserComplaint getAdvertiserComplaint(@Param("id") Integer id);
```

Slicna stvar je uradjena i za dobavljanje objekta EntityComplaint:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query(value = "SELECT e FROM EntityComplaint e WHERE e.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
EntityComplaint getEntityComplaint(@Param("id") Integer id);
```

3. *Na jedan zahtjev za izvjestaj i odobravanje penala za klijenta moze da odgovori najvise jedan administrator*

Opis problema: Administrator ima opciju da pregleda sve zahtjeve za penale na koje do tada nije odgovoreno i moze da ih odobri/odbije. Problem nastaje kada dva administratora istovremeno pokusaju da odgovore na zahtjev. Kao I u prethodnim slucajevima, ukoliko se status zahtjeva nije azurirao,a neki drugi administrator sistema pokusa da odgovori na isti zahtjev to ce biti moguce. Na taj nacin ce se dva puta azurirati zahtjev,sto predstavlja problem. Njegovo rjesenje opisano je u nastavku.



Slika 3. Rjesavanje konfliktne situacije za prihvatanje/odbijanje penala

Rjesenje problema:

Za rjesavanje opisanog problema koristen je pesimisticki pristup. Situacija je slicna kao u prethodno opisanim primjerima. Kada jedan administrator pristupi odobravanju penala, drugi administrator pristupi odobravanju istog penala dobice PessimisticLockingFailureException. Kao I u prethodnim slucajevima na servisnu klasu je stavljena anotacija:

```
@Service
@Transactional
public class ReportService {
```

A u okviru repozitorijuma iznad koristene metode dodate su sljedece anotacije:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query(value = "SELECT e FROM Report e WHERE e.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Report getReport(@Param("id") Integer id);
```