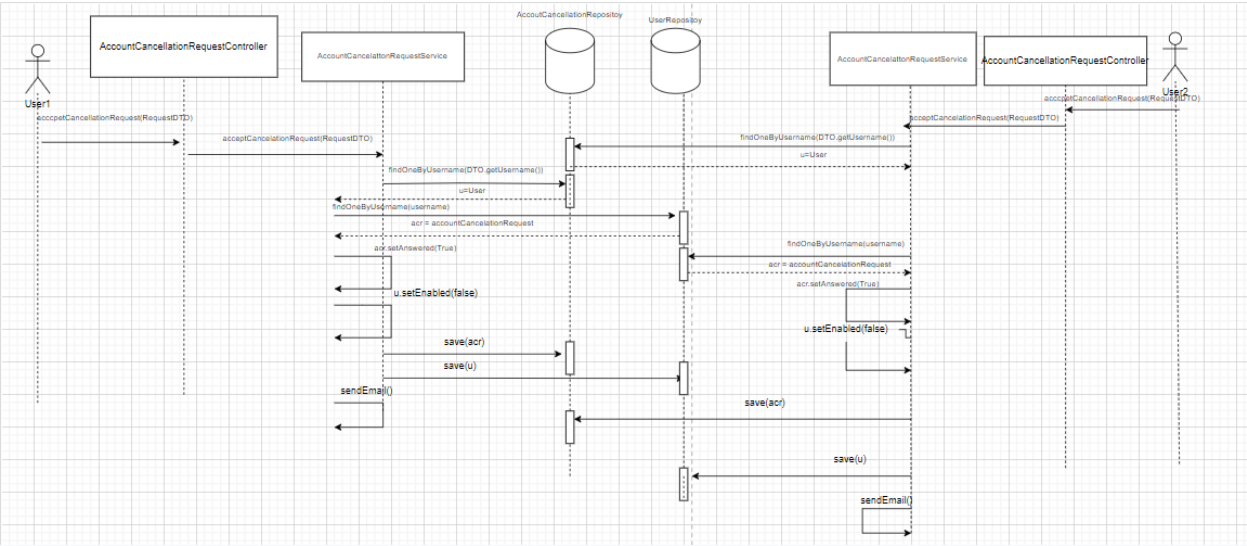
Konflikt 1 - na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema

Opis problema:

Mogući problem jeste situacija u kojoj više admina želi da odgovori na isti zahtev za brsanje profila. Ukoliko bi dva ili više admina u isto vreme pokušali da odgovore na odredjeni zahtev za brisanje profila, postojala bi mogucnost da bi na isti zahtev korisnika bilo odgovoreno na različit način i da klijentu stignu dva mejla s odgovorom. Dijagram opisane situacije može se videti na Slika 1.

Slika 1.



Rešenje:

Problem se može rešiti pomoću pesimističkog ili optimističkog zaključavanja baze. Za ovaj primer korišćeno je pesimističko zaključavanje baze na find metode AccountCancellationRequestRepository.U klasi AccountCancellationRequestService metode AcceptCancellationRequest i DeclineCancelation Request su označene @Transactional anotacijom gde amin koji je kasnije odgovorio mora da sačeka odgovor prvog.

Slika 2.

```
2 usages  Vanja Šerfeze
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT  a FROM  AccountCancellationRequest a where a.user.username=:username")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "1000")})
public AccountCancellationRequest getLockedCancellationRequest(String username);
```

Slika 3.

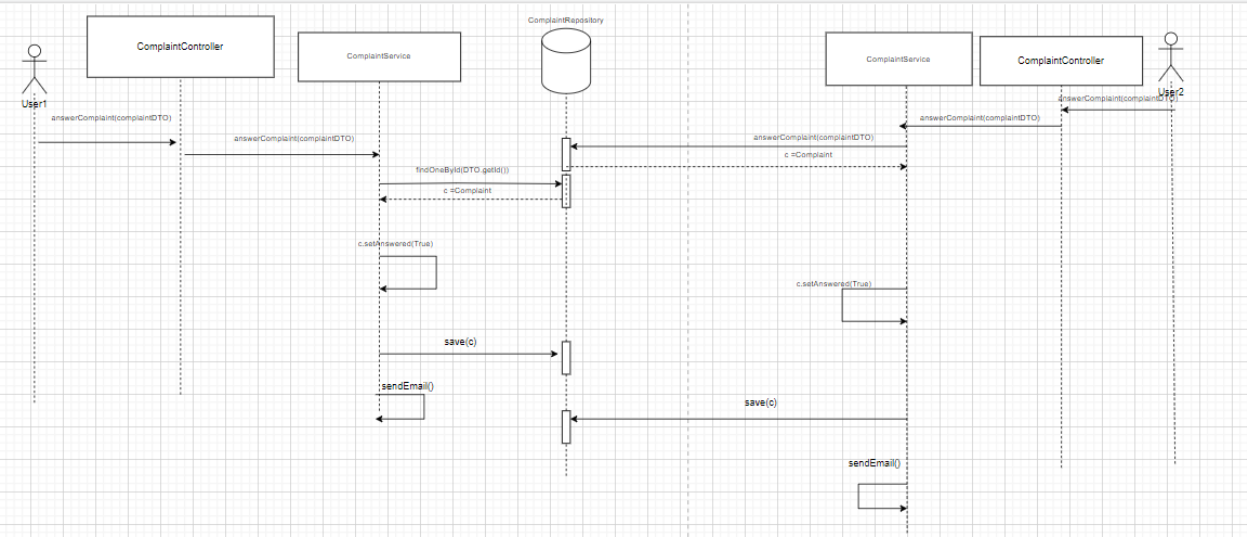
```
@Transactional
public boolean acceptCancellationReques(CancellationRequestDTO dto) {
        User u =  userRepository.findOneByUsername(dto.getClient());

        if(u == null){
            return false;
        }
        System.out.println( u.getUsername());
         AccountCancellationRequest acr = accountCancellationRequestRepository.getLockedCancellationRequest(u.getUsername());
            if(!acr.isAnswered()){
                acr.setAnswered(true);
                accountCancellationRequestRepository.save(acr);
                u.setDeleted(true);
                u.setEnabled(false);
                userRepository.save(u);
                emailService.deleteRequestApprovedEmail(dto.getClient());
            }


        return true;
```

Konflikt 2 - na jednu žalbu može da odgovori samo jedan administrator sistema

Opis problema

Mogući problem jeste situacija u kojoj više admina želi da odgovori na istu žalbu . Ukoliko bi dva ili više admina u isto vreme pokušali da odgovore na odredjenu žalbu, postojala bi mogudnost da bi na istu žalbu korisnika bilo odgovoreno na različit način i da klijentu stignu dva mejla s odgovorom. Dijagram opisane situacije može se videti na Slika 4.

Slika 4.



Rešenje

Problem se može rešiti pomoću pesimističkog ili optimističkog zaključavanja baze. Za ovaj primer korišćeno je pesimističko zaključavanje baze na find metode ComplaintRepositoya.U klasi ComplaintService metode i AnswerComplaint je označena @Transactional anotacijom gde amin koji je kasnije odgovorio mora da sačeka odgovor prvog.

Slika 5.

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT  c FROM  Complaint c where c.id=:id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "1000")})
public Complaint getLockedComplaint(int id);
```

Slika 6.

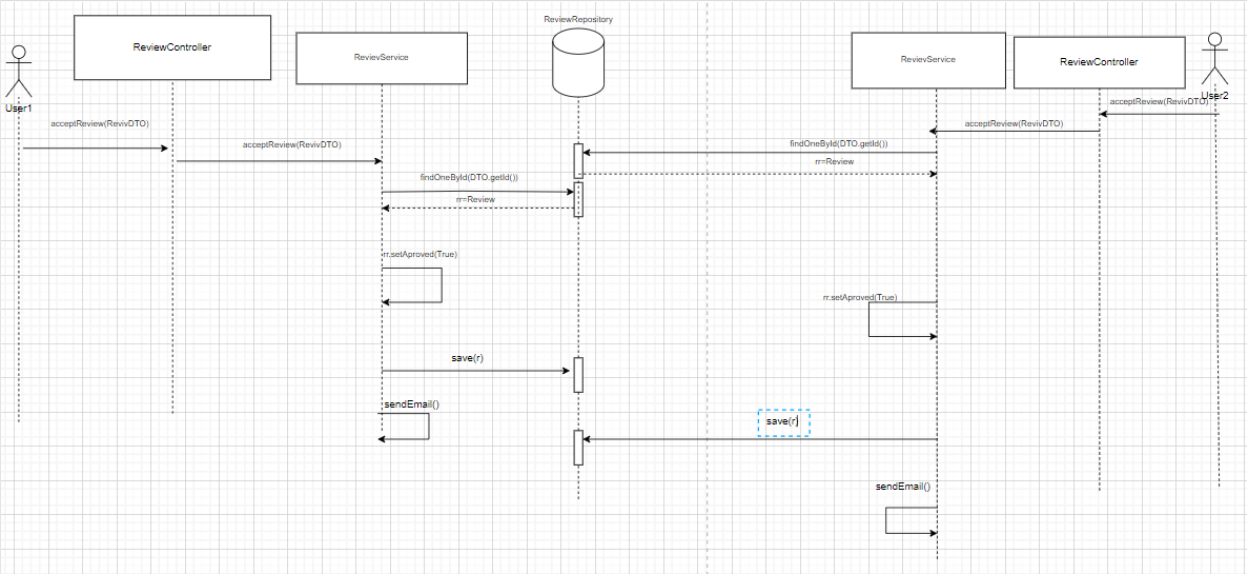
```
@Transactional
public boolean answerComplaint(ReservationReportDTO dto) {
    Complaint rr = complaintRepository.getLockedComplaint(dto.getId());
    if(!rr.isAnswered()) {
        rr.setAnswered(true);
        complaintRepository.save(rr);
        emailService.sendComplaintEmail(rr.getSender().getUsername(), rr.getSystemEntity().getOwner().getUsername(), rr.getText()
    }
    return true;

}
```

Konflikt 3 - na jednu recenziju može da odgovori samo jedan administrator sistema

Opis problema

Mogući problem jeste situacija u kojoj više admina želi da odgovori na istu recenziju . Ukoliko bi dva ili više admina u isto vreme pokušali da odgovore na odredjenu recenziju , postojala bi mogudnost da bi na istu recenziju korisnika bilo odgovoreno na različit način i da klijentu stignu dva mejla s odgovorom. Dijagram opisane situacije može se videti na Slika 7.

Slika 7.



Rešenje

Problem se može rešiti pomoću pesimističkog ili optimističkog zaključavanja baze. Za ovaj primer korišćeno je pesimističko zaključavanje baze na find metode ReveiwRepositoya.U klasi ReviewService metode i acceptReveiw je označena @Transactional anotacijom gde amin koji je kasnije odgovorio mora da sačeka odgovor prvog.

Slika 8.

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT  c FROM  Review c where c.id=:id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "1000")})
public Review getLockedReview(int id);
 Vanja Šerfeze
public Review findReviewById(int id);
```

Slika 9.

```
@Transactional
public boolean acceptReviw(ReviewDTO dto) {
    Review rr = reviewRepository.getLockedReview(dto.getId());
    rr.setApproved(true);
    reviewRepository.save(rr);
    emailService.sendReviewEmail(rr.getSystemEntity().getOwner().getUsername(),rr.getText(),rr.getClient().getUsername(),rr.getS
    return true;
}

1 usage  Vanja Šerfeze
@Transactional
public boolean declineReviw(ReviewDTO dto) {
    Review rr = reviewRepository.getLockedReview(dto.getId());
    reviewRepository.delete(rr);
    emailService.sendReservationReportDeclined(dto.getClient(),dto.getOwner(),rr.getText(),dto.getText());
    return true;
```