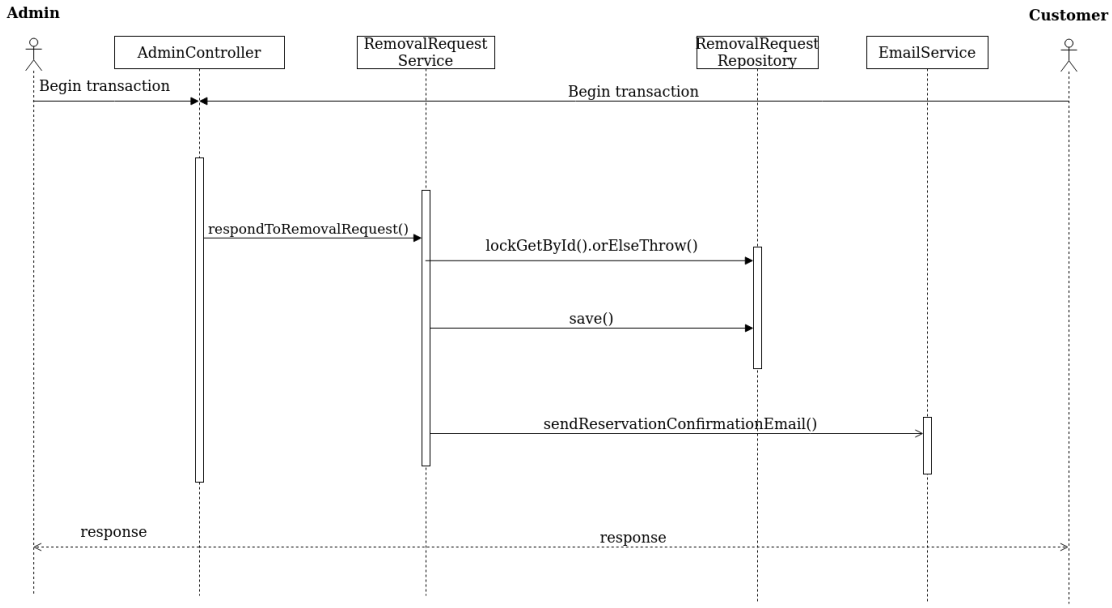
Konkurentni pristup podacima u bazi

Student 3 - Dušan Lazić SW 4/2019

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

***Problem*:** Klijent ima mogućnost da podnese zahtev za brisanje svog naloga. Na zahteve za brisanje odgovaraju adminsitratori, tako što odobre ili odbiju zahtev. Problem nastaje kada dva ili više adminsitratora u isto vreme na ekranu vide isti zahtev za brisanje naloga, i u isto vreme pokušaju da odgovore na njega.

***Tok zahteva*:** Na slici 1 je prikazan dijagram sekvence za odgovaranje administratora na zahtev za brisanje naloga.



Slika 1 - Dijagram sekvence za odgovaranje na zahtev za brisajne

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom odgovaranja na zahtev za brisanje od strane administratora je */admin/removal-requests/{id}/* kontrolera *AdminController* i ova metoda je prikazana na slici 2.

```
@PatchMapping(value = "/removal-requests/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasRole('ADMIN')")
public ResponseOK respondToRemovalRequest(@PathVariable Long id, @Valid @RequestBody RemovalRequestResponseDTO dto) {
    removalRequestService.respondToRequest(id, dto);
    return new ResponseOK("Request resolved.");
}
```

Slika 2 - Metoda kontrolera za prvi konfliktni slučaj

Metoda *respondToRemovalRequest* ovog kontrolera poziva metodu *respondToRequest* servisa *RemovalRequestService* koja je prikazana je na slici 3. Ova metoda je ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje zahteva za brisanje. Zahtev za brisanje ostaje zaključan do kraja metode. Ako u bilo kom trenutku administrator pokuša da odgovori na zahtev, odnosno pokuša da zaključa isti ovaj resurs, doći će do bacanja izuzetka pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen izuzetak *RegistrationRequestResponseConflictException,* zbog kojeg će *ControllerAdvisor* adminsitratoru vratiti odgovor sa porukom da neko drugi u ovom trenutku već pokušava da odgovori na ovaj zahtev.

```
@Transactional
public void respondToRequest(Long id, RemovalRequestResponseDTO dto) {
    RemovalRequest request;
    try {
        request = removalRequestRepository.lockGetById(id).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new RegistrationRequestResponseConflictException(
                "Another admin is currently attempting to respond to this removal request." +
                        "Try again in a few seconds.");
    }
    if (!request.getApprovalStatus().equals(ApprovalStatus.PENDING))
        throw new RegistrationRequestAlreadyResolvedException();

    if (dto.getApprove()) {
        request.setApprovalStatus(ApprovalStatus.APPROVED);
        disableUser(request.getUser());
        emailService.sendRemovalApprovalEmail(request);
    } else {
        request.setApprovalStatus(ApprovalStatus.REJECTED);
        request.setRejectionReason(dto.getRejectionReason());
        emailService.sendRemovalRejectionEmail(request);
    }
    removalRequestRepository.save(request);
}
```

Slika 3 - Metoda servisa za prvi konfliktni slučaj

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM RemovalRequest r WHERE r.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<RemovalRequest> lockGetById(Long id);
```

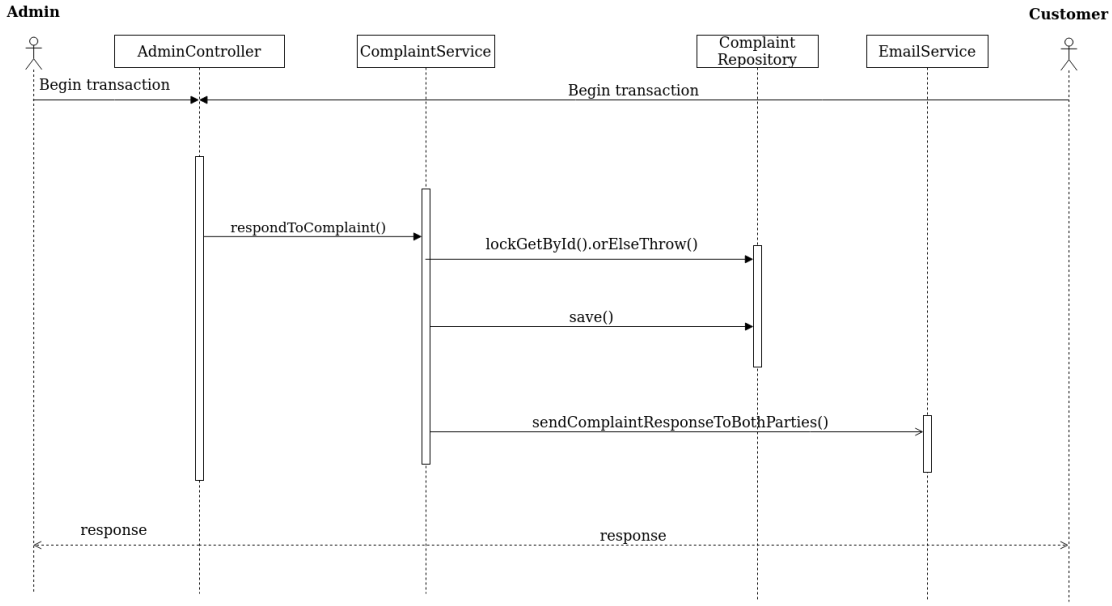
Slika 4 - Metoda *RemovalRequestRepository* repozitorijuma za zaključavanje

Metoda *lockGetById* repozitorijuma *RemovalRequestRepository* služi za dobavljanje i zaključavanje zahteva za brisanje.

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

***Problem*:** Klijent ima mogućnost da podnese žalbu na oglašavača i njegovu uslugu. Na žalbe odgovaraju adminsitratori, tako što u slobodnoj formi pišu odgovor i za klijenta i za oglašavača koji se potom šalju obe strane na mejl. Problem nastaje kada dva ili više adminsitratora u isto vreme na ekranu vide istu žalbu, i u isto vreme pokušaju da odgovore na nju.

Tok zahteva: Na slici 5 je prikazan dijagram sekvence za odgovaranje administratora na žalbu.



Slika 5 - Dijagram sekvence za odgovaranje na žalbu

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom odgovaranja na zahtev za brisanje od strane administratora je */admin/complaints/{id}/* kontrolera *AdminController* i ova metoda je prikazana na slici 6.

```
@PatchMapping(value = "/complaints/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasRole('ADMIN')")
public ResponseOK respondToComplaint(@PathVariable Long id, @Valid @RequestBody ComplaintResponseDTO dto) {
    complaintService.respondToComplaint(id, dto);
    return new ResponseOK("Complaint resolved.");
}
```

Slika 6 - Metoda kontrolera za odgovaranje na žalbu

Metoda *respondToComplaint* ovog kontrolera poziva istoimenu metodu servisa *ComplaintService* i prikazana je na slici 7. Ova metoda ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje žalbe. Ako je pre toga drugi administrator već pristupio žalbi i zaključao je, doći će do bacanja izuzetka pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen izuzetak *AdminConflictException*, zbog kojeg će *ControllerAdvisor* adminsitratoru vratiti odgovor sa porukom da neko drugi u ovom trenutku već pokušava da odgovori na ovu žalbu.

```java
@Transactional
public void respondToComplaint(Long id, ComplaintResponseDTO dto) {
    Complaint complaint;
    try {
        complaint = complaintRepository.lockGetById(id).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new AdminConflictException(
                "Another admin is currently attempting to respond to this complaint. " +
                        "Try again in a few seconds.");
    }

    if (!complaint.getResponseStatus().equals(ResponseStatus.PENDING))
        throw new ComplaintAlreadyResolvedException();

    complaint.setResponseStatus(ResponseStatus.RESOLVED);

    emailService.sendComplaintResponseToBothParties(complaint, dto, complaint.getAdvertisement(),
            complaint.getAdvertisement().getAdvertiser(), complaint.getCustomer(),
            complaint.getCustomer().getAvatar().getStoredFilename());

    complaintRepository.save(complaint);
}
```

Slika 7 - Metoda servisa za drugi konfliktni slučaj

```java
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Complaint c WHERE c.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<Complaint> lockGetById(Long id);
```

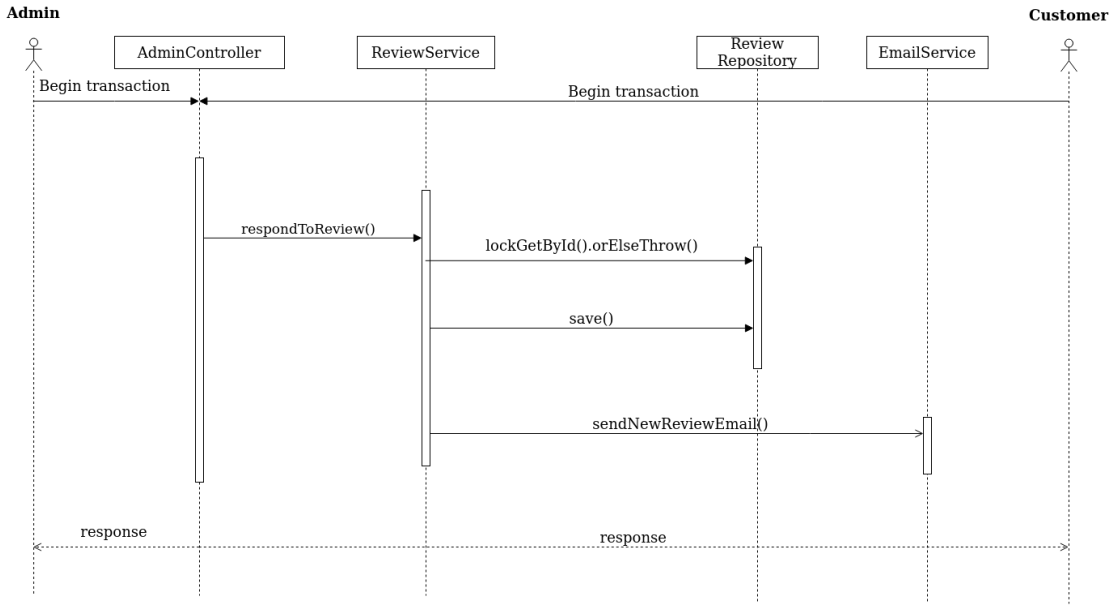
Slika 8 - Metoda ComplaintRepository repozitorijuma

Metoda *lockGetById* repozitorijuma *ComplaintRepository* služi za dobavljanje i zaključavanje žalbe.

3. Na jednu reviziju može da odgovori samo jedan administrator sistema

***Problem*:** Klijent ima mogućnost da ostavi reviziju za uslugu nakon završene rezervacije. Da li će revizija biti objavljena zavisi od administratora koji može da odobri ili odbije reviziju. Problem nastaje kada dva ili više adminsitratora u isto vreme na ekranu vide istu reviziju, i u isto vreme pokušaju da je odobre/odbiju.

Tok zahteva: Na slici 9 je prikazan dijagram sekvence za odgovaranje administratora na reviziju.



Slika 9 - Dijagram sekvence za odgovaranje na reviziju

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom odgovaranja na zahtev za brisanje od strane administratora je */admin/reviews/{id}/* kontrolera *AdminController* i ova metoda je prikazana na slici 10.

```
@PatchMapping(value = "/reviews/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasRole('ADMIN')")
public ResponseOK respondToReview(@PathVariable Long id, @Valid @RequestBody ReviewResponseDTO dto) {
    reviewService.respondToReview(id, dto);
    return new ResponseOK("Review resolved.");
}
```

Slika 10 - Metoda kontrolera za odgovaranje na žalbu

Metoda *respondToReview* ovog kontrolera poziva istoimenu metodu servisa *ReviewService* i prikazana je na slici 11. Ova metoda ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje revizije. Ako je pre toga drugi administrator već pristupio reviziji i zaključao je, doći će do bacanja izuzetka pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen izuzetak *AdminConflictException*, zbog kojeg će *ControllerAdvisor* adminsitratoru vratiti odgovor sa porukom da neko drugi u ovom trenutku već pokušava da odgovori na ovu reviziju.

```java
@Transactional
public void respondToReview(Long id, ReviewResponseDTO dto) {
    Review review;
    try {
        review = reviewRepository.lockGetById(id).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new AdminConflictException(
                "Another admin is currently attempting to respond to this review. " +
                        "Try again in a few seconds.");
    }

    if (!review.getApprovalStatus().equals(ApprovalStatus.PENDING))
        throw new ReviewAlreadyResolvedException();

    if (dto.getApprove()) {
        review.setApprovalStatus(ApprovalStatus.APPROVED);
        emailService.sendNewReviewEmail(review, review.getAdvertisement(),
                review.getAdvertisement().getAdvertiser(),
                review.getCustomer(), review.getRating());
    } else {
        review.setApprovalStatus(ApprovalStatus.REJECTED);
    }

    reviewRepository.save(review);
}
```

Slika 11 - Metoda servisa za treći konfliktni slučaj

```java
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM Review r WHERE r.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<Review> lockGetById(Long id);
```

Slika 12 - Metoda ReviewRepository repozitorijuma

Metoda *lockGetById* repozitorijuma *ReviewRepository* služi za dobavljanje i zaključavanje revizije, na identičan način i u skoro identičnoj ulozi kao i sa prethodne dve situacije.