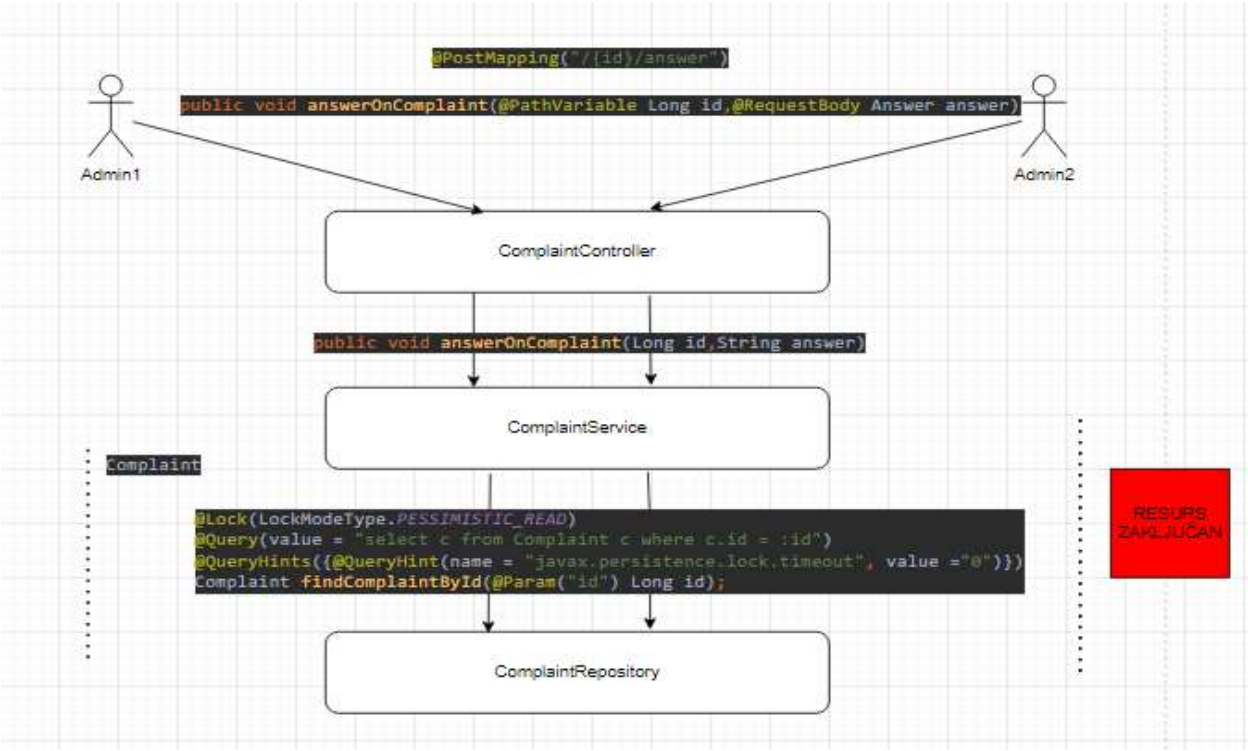
Konkurentni pristup bazi

1. Odgovaranje na žalbe

Situacija - Admin ima mogućnost pregleda svih žalbi koje još nemaju odgovor.Odgovor se unosi u slobodnoj formi i šalje se mejlom i klijentu koji je napisao žalbu i instruktoru pecanja/vlasniku broda/vlasniku vikendica kome je žalba upućena.Kada admin odgovori na žalbu ,ona nestaje iz liste žalbi na koje je moguće odgovoriti.Potencijalna konfliktna situacija nastaje u slučaju kada dva admina žele da odgovore na istu žalbu jer u trenutku kada su im žalbe izlistane na njih nije bilo odgovoreno.Ako bi oba admina odgovorila na žalbu klijent i instruktor/vlasnik bi dobili po dva mejla koja predstavljaju odgovore na istu žalbu ali od strane različitih admina.




Za zaključavanje resursa korišćeno je pesimističko zaključavanje kako drugi admin ne bi mogao da mijenja resurs-Complaint.Iznad metode servisa ***void answer(Long id,String answer)*** stavljena je anotacija ***@Transactional*** kako bi se održala konzistencija svih metoda koje su potrebne za izvršavanje funkcionalnosti.

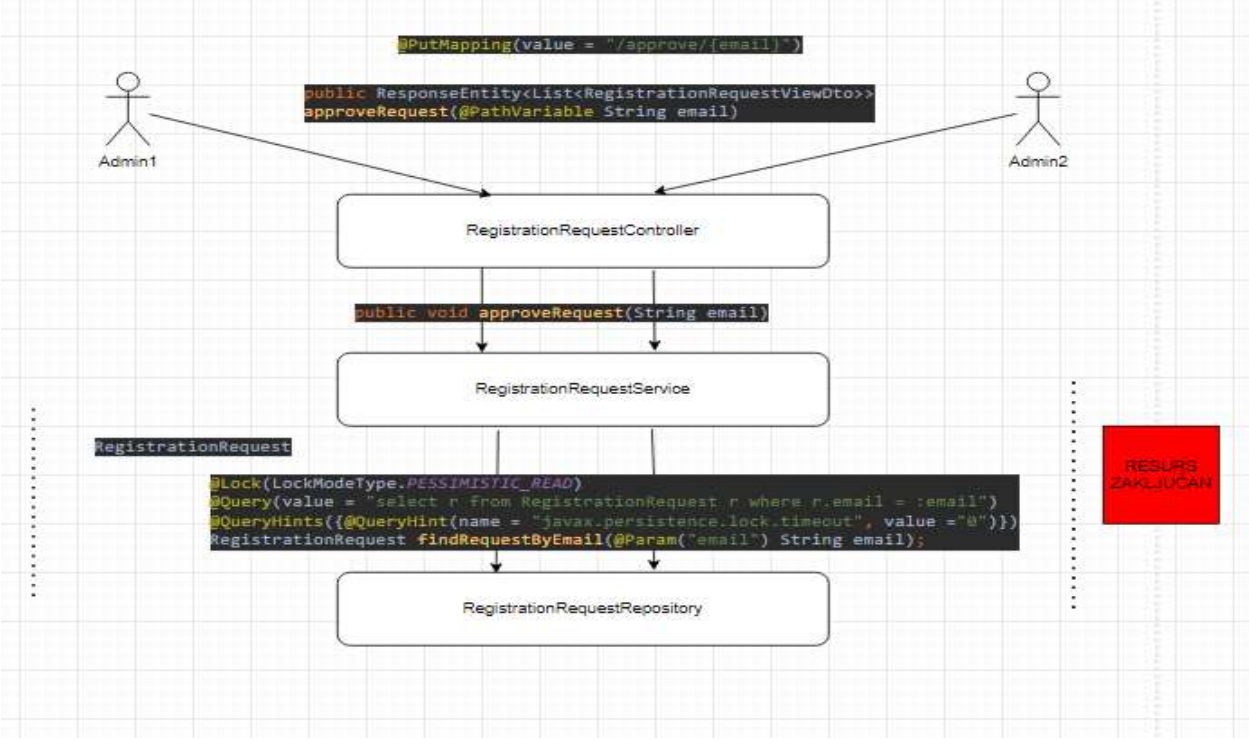
```
@Transactional
@Override
public void answerOnComplaint(Long id,String answer) {
    Complaint complaint = complaintRepository.findComplaintById(id);
    complaint.setAnswered(true);
    complaintRepository.save(complaint);
    notifyOwnerAndClient(complaint.getReservation(),answer,complaint.getComment());
}
```

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query(value = "select c from Complaint c where c.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
Complaint findComplaintById(@Param("id") Long id);
```

Zaključavanje će onemogućiti drugog admina da pristupi resursu kako bi ga izmijenio-setovao polje answered = true i poslao mejl klijentu koji je napisao žalbu i vlasniku/instruktora na koga se žalba odnosi.

2.Prihvatanje/odbijanje zahtjeva za registraciju

Situacija - Da bi vlasnici,instruktori i klijenti mogli da se prijave na novonapravljeni nalog,neophodno je da admin odobri zahtjev za registraciju.Pored odobravanja ,admin može i da odbije zahtjev za registraciju.U slučaju odbijanja admin unosi razlog svoje odluke koji se šalje na mejl osobe koja je podnijela zahtjev za registraciju.Konfliktna situacija može nastati ukoliko dva admina pokušaju isti zahtjev da odobre /odbiju.Jedan admin bi mogao da prihvati zahtjev,dok drugi može da odbije taj isti zahtjev.U takvom slučaju potencijalni korisnik aplikacije će dobiti mejl da je njegov zahtjev odbijen,a može se desiti da je zapravo aktiviran ukoliko je admin koji je htio da aktivira nalog ,resurs izmijenio posljednji.




Za zaključavanje resursa korišćeno je pesimističko zaključavanje kako drugi admin ne bi mogao da mijenja resurs-RegistrationRequest.Iznad metoda servisa ***void approveRequest(String email)/ void rejectRequest(String email)*** stavljena je anotacija ***@Transactional*** kako bi se održala konzistencija svih metoda koje su potrebne za izvršavanje funkcionalnosti

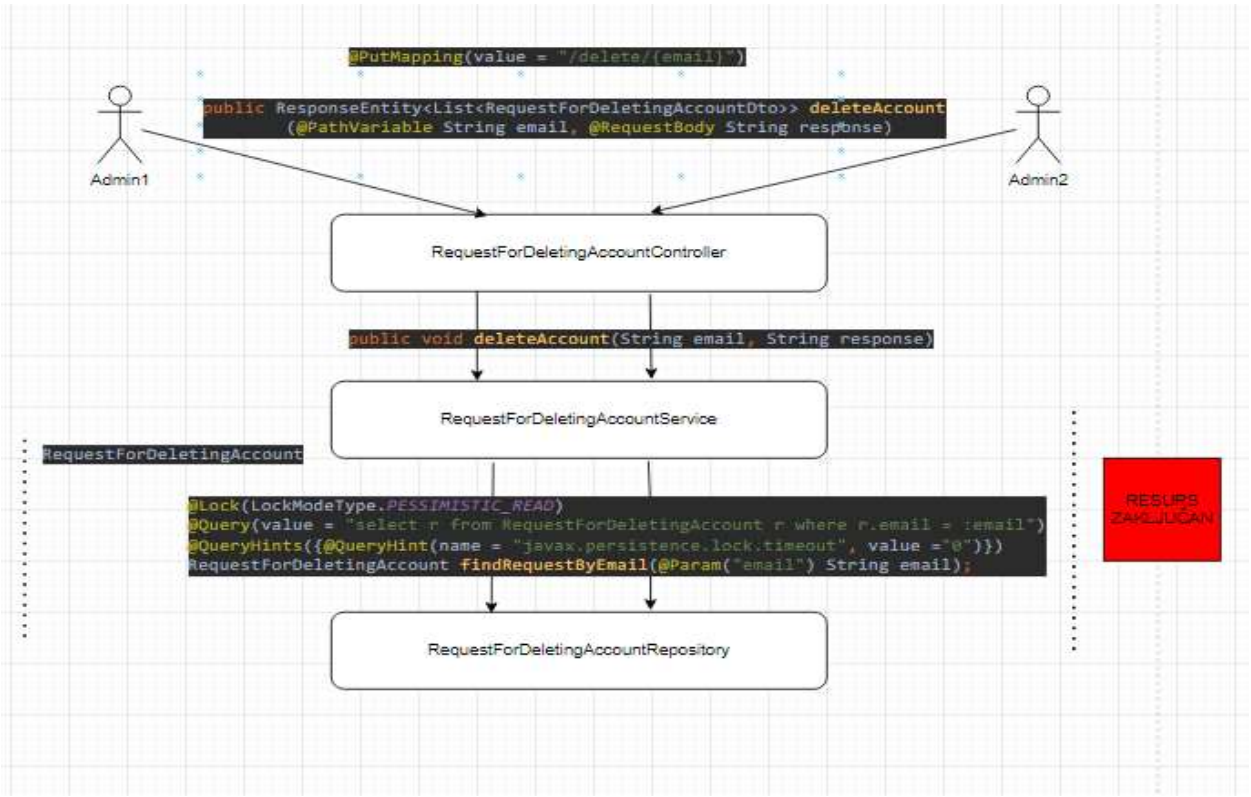
```java
@Transactional
@Override
public void approveRequest(String email) {
    RegistrationRequest request = registrationRequestRepository.findRequestByEmail(email);
    userService.activateAccount(email);
    registrationRequestRepository.delete(request);
    emailSenderService.sendEmail(email, subject: "Aktivacija naloga", body: "Vaš zahtjev za registraciju je prihvaćen."
            "Sada se možete prijaviti na naš sajt sa vašim kredencijalima.");
}
```

```java
@Transactional
@Override
public void rejectRequest(String email,String reason) {
    RegistrationRequest request = registrationRequestRepository.findRequestByEmail(email);
    registrationRequestRepository.delete(request);
    emailSenderService.sendEmail(email, subject: "Aktivacija naloga", body: "Vaš zahtjev za registraciju je odbijen.\n " +
            "Razlog za odbijanje: " + reason);
}
```

```java
@Lock(LockModeType.PESSIMISTIC_READ)
@Query(value = "select r from RegistrationRequest r where r.email = :email")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
RegistrationRequest findRequestByEmail(@Param("email") String email);
```

3.Odobravanje/odbijanje zahtjeva za brisanje naloga

Situacija - Admin ima mogućnost pregleda svih zahtjeva za brisanje koji nisu ni prihvaćeni ni odbijeni.Razlog odluke se unosi u slobodnoj formi i šalje se mejlom korisniku koji je podnio zahtjev za brisanje naloga.Kada admin odbije/prihvati zahtjev,on nestaje iz liste zahtjeva na koje je moguće odgovoriti.Potencijalna konfliktna situacija nastaje u slučaju kada dva admina žele da prihvate/odbiju isti zahtjev.Ako bi oba admina uspjela da izvrše prihvatanje/odbijanje zahtjeva korisnik aplikacije bi dobio dva mejla koja predstavljaju odgovore na istu žalbu ali od strane različitih admina.




```java
@Transactional
@Override
public void deleteAccount(String email, String response) {
    RequestForDeletingAccount request = requestForDeletingAccountRepository.findRequestByEmail(email);
    User user = userService.findByEmail(email);
    userService.deleteUser(user);
    deleteRequest(request);
    emailSenderService.sendEmail(email, subject: "Brisanje naloga", body: "Vaš zahtjev za brisanjem naloga je prihvaćen.\n" +
            "Odgovor admina na zahtjev: " + response);
}
@Transactional
@Override
public void rejectDeleting(String email, String response) {
    RequestForDeletingAccount request = requestForDeletingAccountRepository.findRequestByEmail(email);
    deleteRequest(request);
    emailSenderService.sendEmail(email, subject: "Brisanje naloga", body: "Vaš zahtjev za brisanjem naloga je odbijen.\n" +
            "Odgovor admina na zahtjev: " + response);
}
```

```java
@Lock(LockModeType.PESSIMISTIC_READ)
@Query(value = "select r from RegistrationRequest r where r.email = :email")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
RegistrationRequest findRequestByEmail(@Param("email") String email);
```

Za zaključavanje resursa korišćeno je pesimističko zaključavanje kako drugi admin ne bi mogao da mijenja resurs-RequestForDeletingAccount.Iznad metoda servisa ***void deleteAccount(String email,String response)/ void rejectDeleting(String email,String response)*** stavljena je anotacija ***@Transactional*** kako bi se održala konzistencija svih metoda koje su potrebne za izvršavanje funkcionalnosti