Rešavanje konfliktnih situacija prilikom konkurentnog pristupa bazi podataka

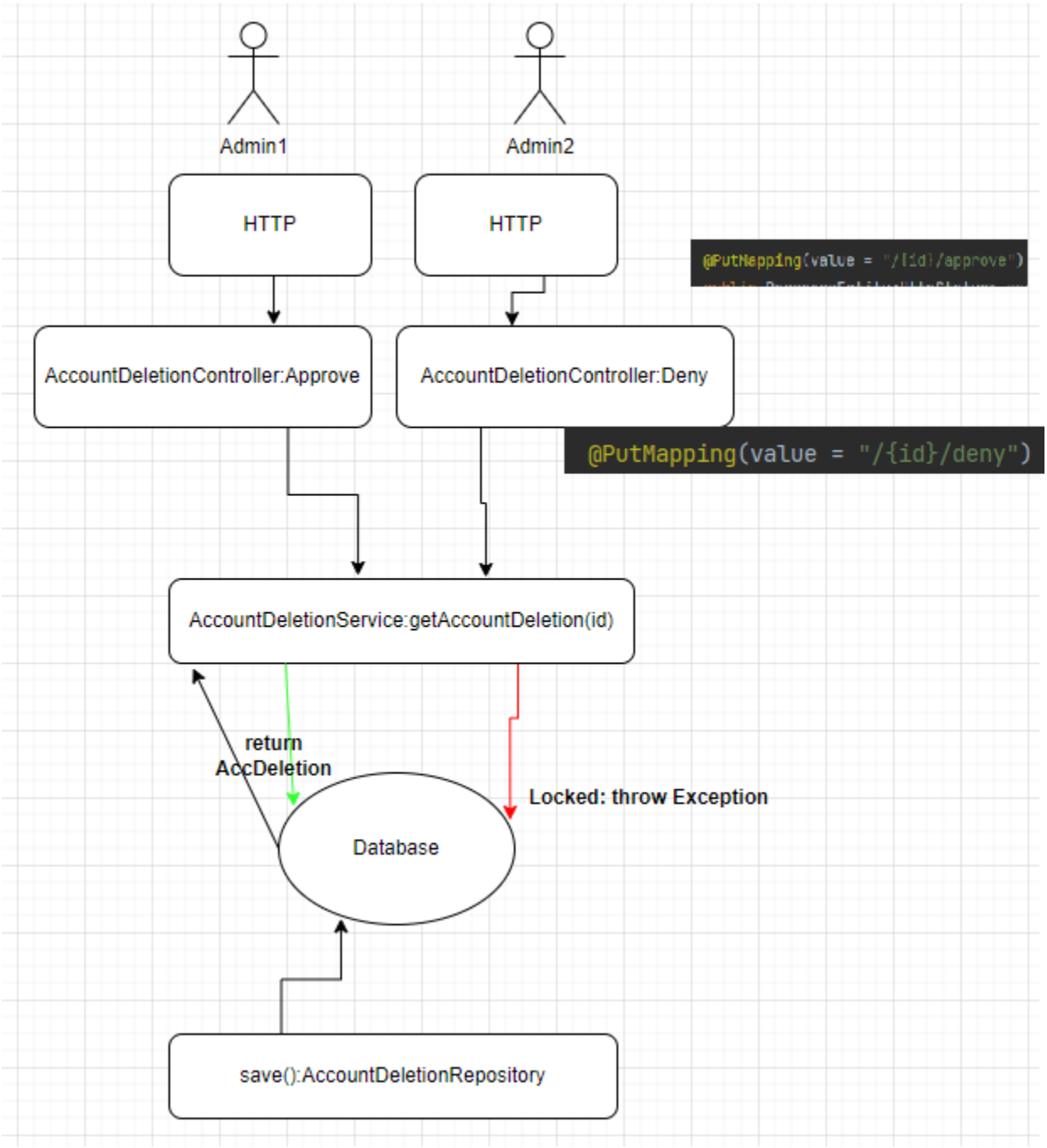
Student 3 – Vladimir Rokvić RA69-2018

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

Opis:

Problem nastaje kada imamo dva administratora koji žele da odgovore na isti zahtev brisanja naloga. Zamislimo da oba administratora istovremeno odgovaraju, prvi odbija zahtev za brisanje naloga, dok ga drugi potvrđuje. Kako su zahtevi poslati u isto vreme i sam proces obuhvata slanje email-a, sama verifikacija da postoji zahtev za brisanje će biti uspešno odrađena za oba zahteva, što dovodi do situacije da će nalog biti obrisan, a korisnik će dobiti dva mejla koja saopštavaju kontradiktorne informacije.

Tok:



Rešenje: Koristi se pesimističko zaključavanje resursa. Prilikom pristupanja zahtevu za brisanje naloga vrši se zaključavanje baze podataka, ukoliko neko proba da pristupi dobija exception.

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

Odgovarajuće metode su označene sa transactional.

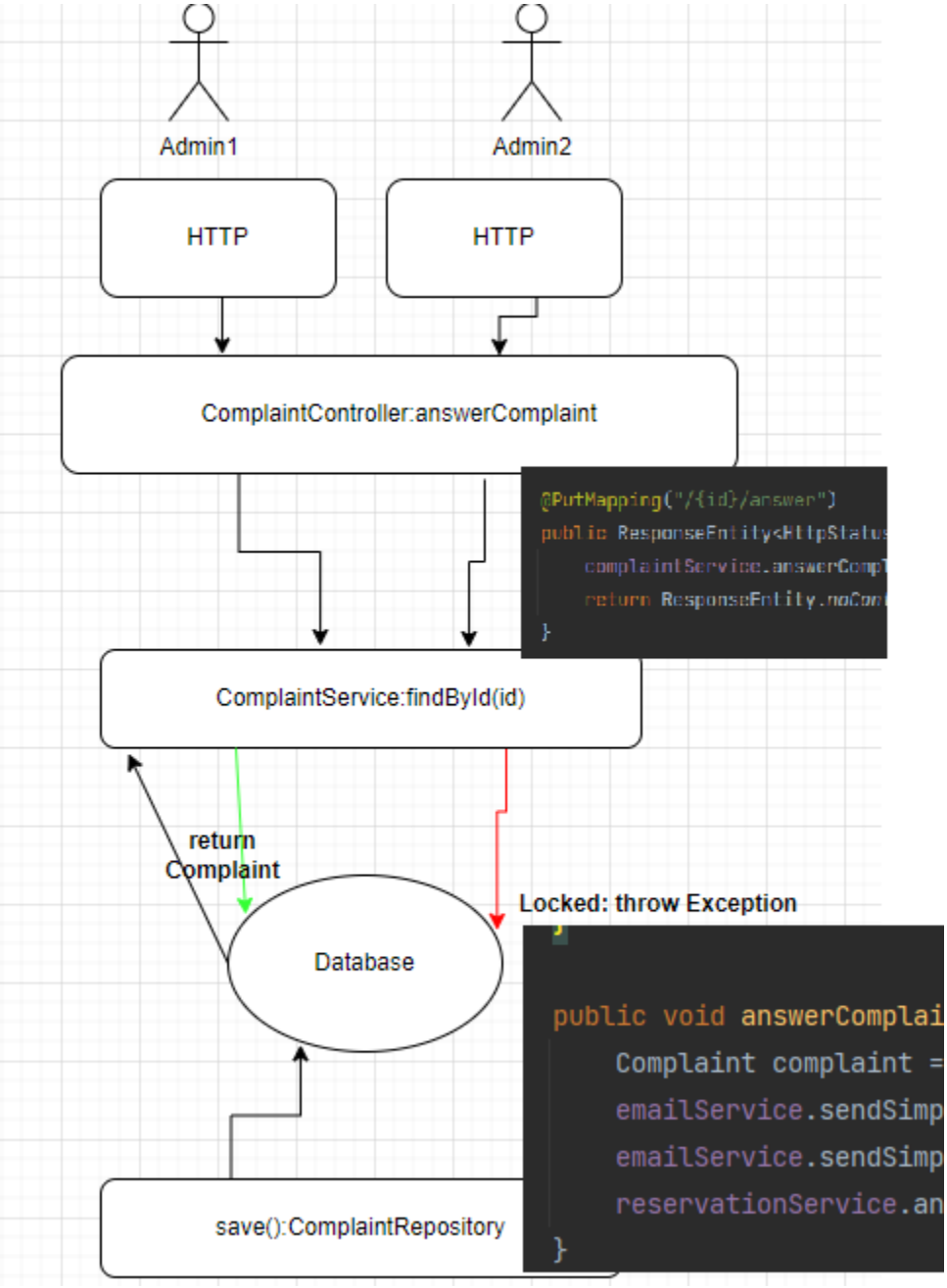
```java
@Transactional
public void approve(Long id, String reason) {
    AccountDeletion accountDeletion = repository.findOneById(id);
    emailService.sendSimpleMessage(accountDeletion.getEmail(), "Account deletion approved", reason);
    userService.deleteUser(accountDeletion.getEmail());
    repository.deleteById(id);
}

@Transactional
public void deny(Long id, String reason) {
    AccountDeletion accountDeletion = repository.findOneById(id);
    emailService.sendSimpleMessage(accountDeletion.getEmail(), "Account deletion denied", reason);
    repository.deleteById(id);
}
```

Na žalbu može da odgovori samo jedan administrator sistema

Opis:

Posmatrajmo dva admina koja imaju pristup svim žalbama u sistemu i odlučili su da odgovore na istu žalbu istovremeno. HTTP zahtev se šalje na isti endpoint na backend-u. Kako su zahtevi poslati u isto vreme i sam proces obuhvata slanje email-a, sama verifikacija da postoji žalba će biti uspešno odrađena za oba zahteva, što dovodi do situacije da će žalba biti razrešena, a korisnik i instruktor pecanja, tj. vlasnik će dobiti dva mejla koja saopštavaju različite informacije.

Tok:



Rešenje:

Koristi se pesimističko zaključavanje resursa. Prilikom pristupanja žalbi vrši se zaključavanje baze podataka, ukoliko neko proba da pristupi dobija exception.

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

Dodaje se transactional na metodu servisa.

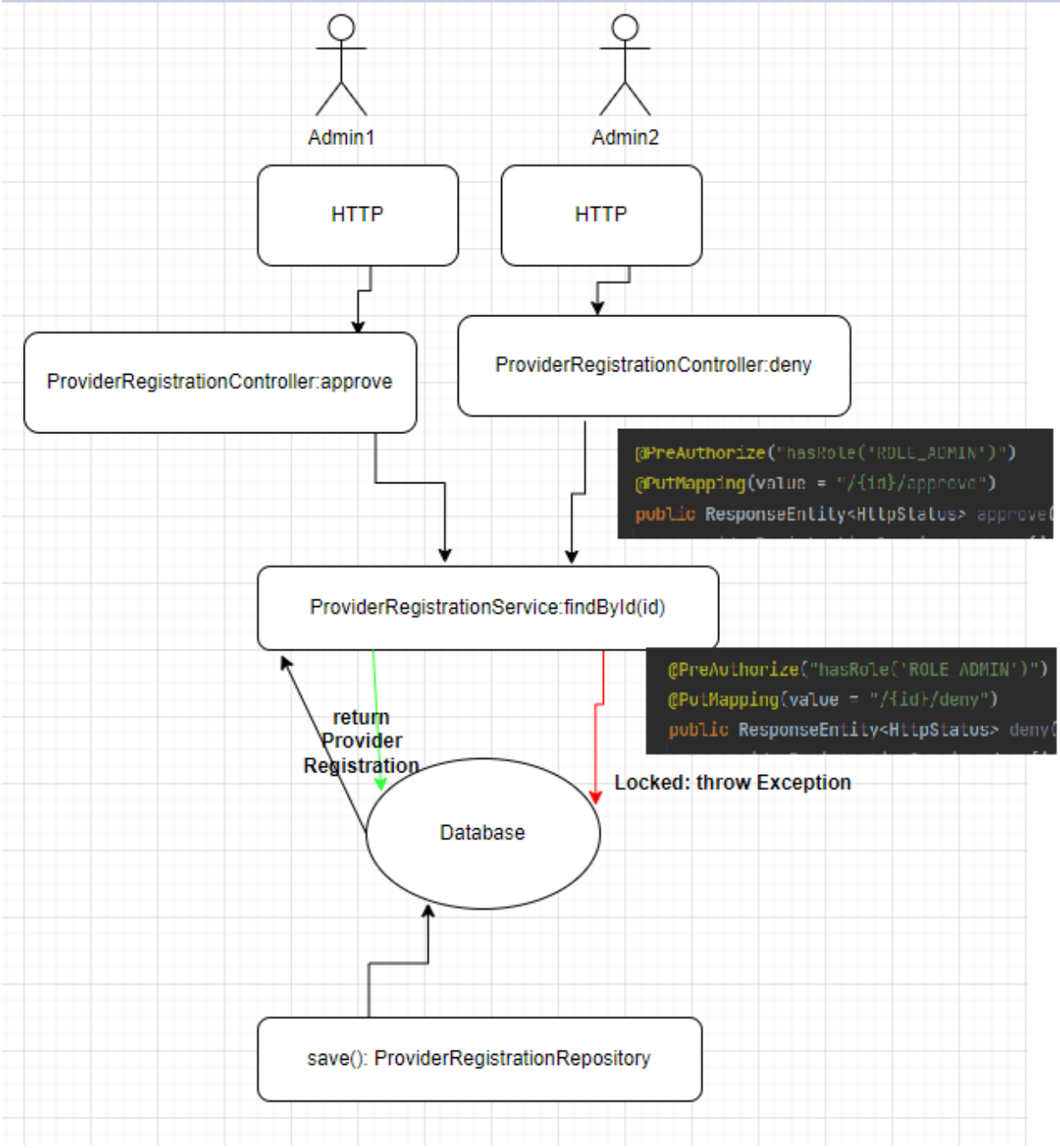
```
@Transactional
public void answerComplaint(Long id, String message) {
    Complaint complaint = complaintRepository.findOneById(id);
    emailService.sendSimpleMessage(complaint.getReservation().getOwnerEmail(), "Complaint Resolved", message);
    emailService.sendSimpleMessage(complaint.getReservation().getClient().getEmail(), "Complaint Resolved", message
    reservationService.answerComplaint(complaint.getReservation().getId());
}
```

Na zahtev za registraciju može da odgovori samo jedan administrator sistema

Opis:

Problem nastaje kada imamo dva administratora koji žele da odgovore na isti zahtev za registraciju. Zamislimo da oba administratora istovremeno odgovaraju, prvi odbija zahtev za registraciju naloga, dok ga drugi potvrđuje. Kako su zahtevi poslati u isto vreme i sam proces obuhvata slanje email-a, sama verifikacija da postoji zahtev za brisanje će biti uspešno odrađena za oba zahteva, što dovodi do situacije da će nalog biti odbijen ili potvrđen, a korisnik dobija dva mejla koja daju kontradiktorne informacije.

Tok:




```java
public void approve(Long id) {
    ProviderRegistration providerRegistration = findById(id);
    User newUser = userService.createUser(providerRegistration);
    emailService.sendSimpleMessage(newUser.getEmail(), "Fishing Acc
    repository.deleteById(id);
}

public void deny(Long id, String reason) {
    ProviderRegistration providerRegistration = findById(id);
    emailService.sendSimpleMessage(providerRegistration.getEmail(),
    repository.deleteById(id);
}
```

Rešenje:

Koristi se pesimističko zaključavanje resursa baze. Prilikom pristupanja zahtevima za registraciju klijenata vrši se zaključavanje baze podataka, ukoliko neko proba da pristupi dobija exception.

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

Servisne metode koje koriste se anotiraju sa Transactional.

```
@Transactional
public void approve(Long id) {
    ProviderRegistration providerRegistration = repository.findOneById(id);
    User newUser = userService.createUser(providerRegistration);
    emailService.sendSimpleMessage(newUser.getEmail(), "Fishing Account Approved",
    repository.deleteById(id);
}

@Transactional
public void deny(Long id, String reason) {
    ProviderRegistration providerRegistration = repository.findOneById(id);
    emailService.sendSimpleMessage(providerRegistration.getEmail(), "Fishing Accou
    repository.deleteById(id);
}
```