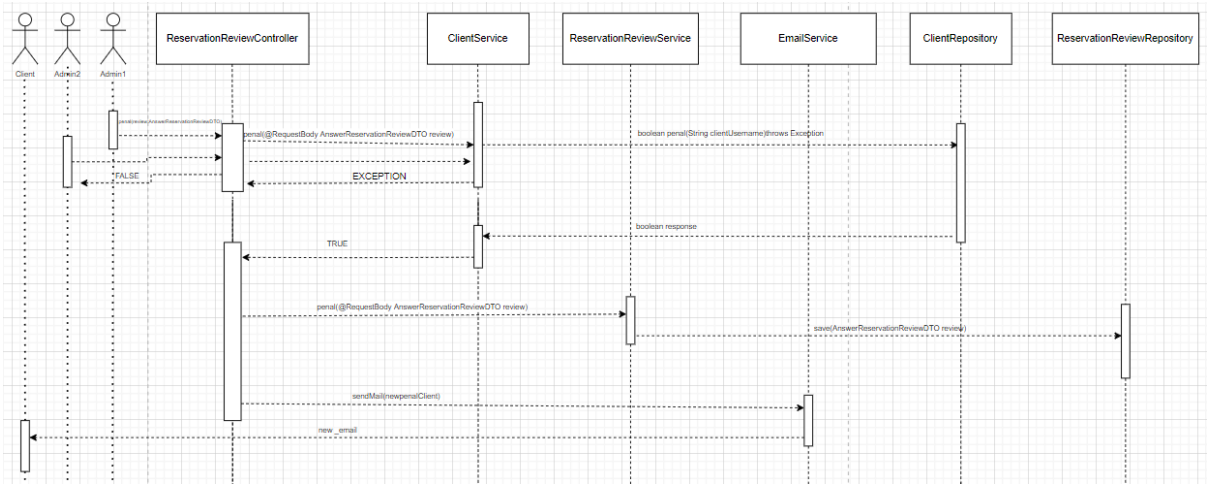
Извештај о имплементираним трансакционим методама

Студент3:Јован Гашпар IN60/2018

У овом раду је представљен пример решења трансакционих метода за следећа функционална ограничења:

1. на један захтев за брисање налога може да одговори само један администратор система;
2. на једну жалбу може да одговори само један администратор система;
3. на једну оцену може да одговори само један администратор система;
4. један клијент може да добије један пенал од једог захтева за санкционисање;
   (када је узастопно позвана метода за додељивање пенала,
   само први пут додељује пенал)

   У даљем наставку ћемо се бавити приказом и решавањем конфликтне ситуације 4. На слици испод је приказан ток интеракције.



Слика 1 – Приказ тока интеракције

Проблем настаје у ситуацији када би истовремено више админа покушало да казни клијента. У том случају би добио више од једног пенала за један извештај. Проблем је решен оптимистичким закључавањем клијента. У току прве трансакције верзија појаве типа ентитета клијент ће се променити. У тренутку када треба да се одвије трансакција коју је започео админ2 долази до повлачења трансакције због тога што се верзије не поклапају.

Имплементација оптимистичког закључавања функционалног ограничења 4.

```
@PostMapping("/penal")
public ResponseEntity<Boolean> penal(@RequestBody AnswerReservationReviewDTO review) {
    Boolean response = false;
    try {
        if(clientService.penal(review.getClientUsername()))
        {
            response = true;
            this.reservationReviewService.save(review);

            try {
                NewPenalEmailContext newpenalOwner = new NewPenalEmailContext();
                newpenalOwner.init(review);
                newpenalOwner.setTo(review.getOwnerEmail());
                emailService.sendMail(newpenalOwner);
            } catch (MessagingException e) {
                e.printStackTrace();
            }

            try {
                NewPenalEmailContext newpenalClient = new NewPenalEmailContext();
                newpenalClient.init(review);
                newpenalClient.setTo(review.getClientEmail());
                emailService.sendMail(newpenalClient);
            } catch (MessagingException e) {
                e.printStackTrace();
            }


        }
    } catch (Exception e) {
        System.out.println("клијент је већ кажњен");}
    return new ResponseEntity<>(response, HttpStatus.OK);
}
```

Слика 2 – позив метода из контролера(class:ReservationReviewController.java ¬line 55)

```
@Transactional(value = TxType.REQUIRES_NEW)
public boolean penal(String clientUsername)throws Exception {
    Boolean response = false;
    try {
    Client client = this.clientRepository.findByAccountUsername(clientUsername);
    client.setPenals(client.getPenals()+1);
    this.clientRepository.save(client);
    response = true;
        }
    catch(Exception e) {
        throw e;
    }
    finally {return response;}
}
```

Слика 3 – трансакциона метода за додавање пенала (class:ClientService.java ¬line 345)

```
@Version
private Long version;

private int penals;
```

Слика 4 – база чува верзију чуваног ентита која се пореди са верзијом нове трансакције (class : Client.java ¬line 53)