KONKURENTNOST

KONFLIKT 1:

Više klijenata ne mogu da naprave rezervaciju istog entiteta u isto (ili vreme koje se preklapa) vreme Jedan od mogućih problema jeste situacija u kojoj više korisnika istovremeno pokušava da rezerviše isti entitet. Ukoliko bi dva ili više korisnika u isto, ili barem preklapajuće vreme odradila proveru dostupnosti jednog entiteta, obe provere bi vratile pozitivan rezultat, te bismo bili u riziku dualne zauzetosti jednog objekta.

Rešenje: Problem rešavamo postavljanjem pesimističkog zaključavanja na get metode u repozitorijumu za vikendice, brodove i avanture. Dodatno, u klasi ReservationService su logike zakazivanja uokvirene try/catch blokom, a metode označene @Transactional anotacijom.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Boat findLockedById(Long id);
```

```
@Transactional
public BoatReservation createBoatReservation(BoatReservation boatReservation) {
    try {
```

KONFLIKT 2 Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vreme. Kako su akcijske ponude dostupne svim prijavljenim klijentima za brzu rezervaciju, lako je moguće da dva klijenta u isto, ili približno isto vreme pokušaju da rezervišu istu ponudu.

Rešenje: Kao I kod konflikta broj 1, koristili smo pesimističko zaključavanje dodavanjem upita sa anotacijom PESSIMISTIC_WRITE locka, koji će sprečiti da 2 ili više korisnika istovremeno čitaju podatke o dostupnim akcijama u svrhu zakazivanja istih. Postavljene su i anotacije @Transactional na metodu klasa QuickReservations create.

```
@Query(value="select q from boat_quick_reservation q where q.id=:id  ", nativeQuery=true)
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
BoatQuickReservation getLock(Long id);
```

```
@Transactional
public BoatQuickReservation createBoatQuickReservation(BoatQuickReservation boatQuickReservation) {
```

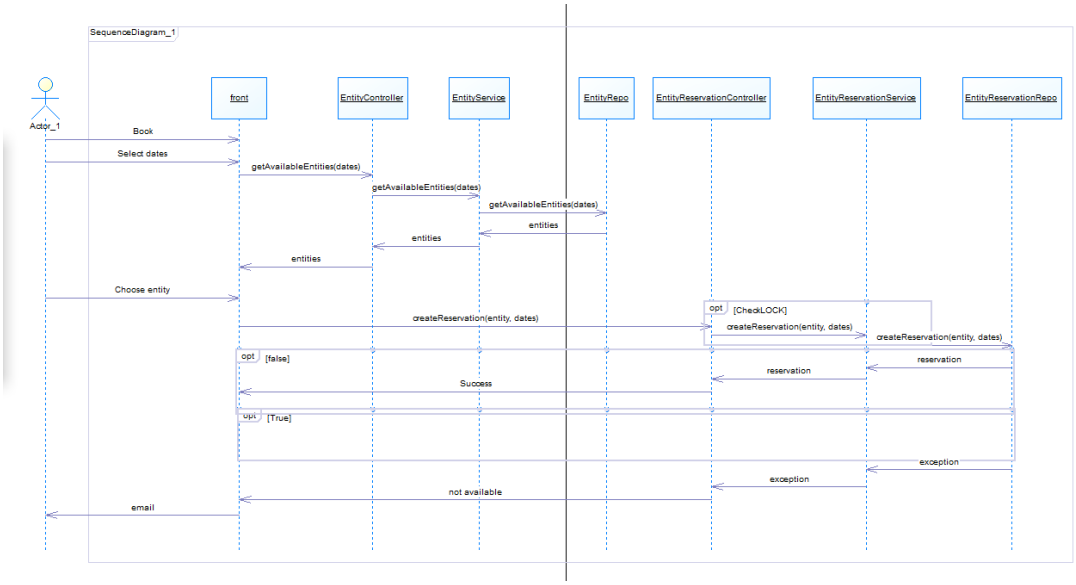
PRONALAZAK KONFLIKTA

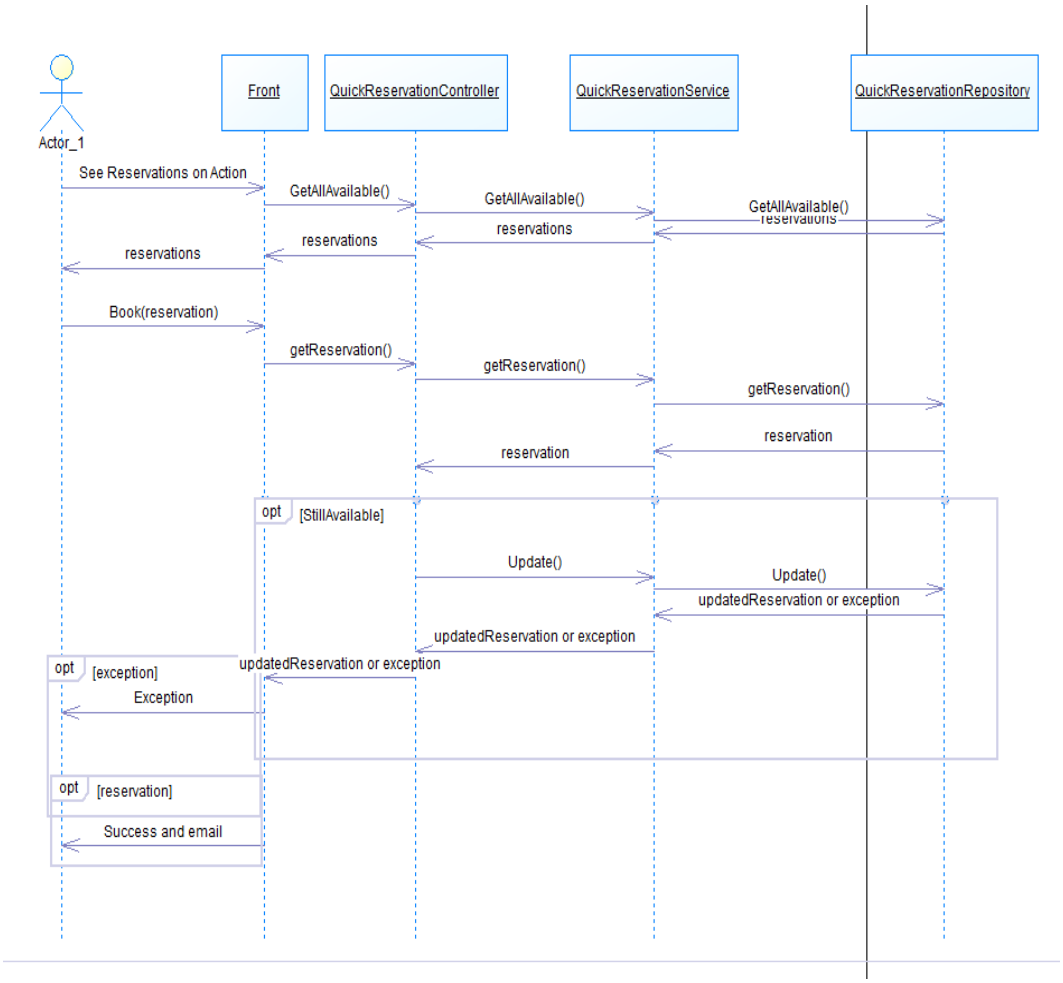
KONFLIKT 3: Više korisnika pokuša istovremeno da se registruje istim mejlom. Naš sistem dozvoljava kreiranje jednog usera na osnovu jednog mejla, drugim rečima ne mogu dva usera da imaju isti mejl. Do konflikta dolazi kada dva korisnika istovremeno pokušaju da naprave nalog istim mejlom.

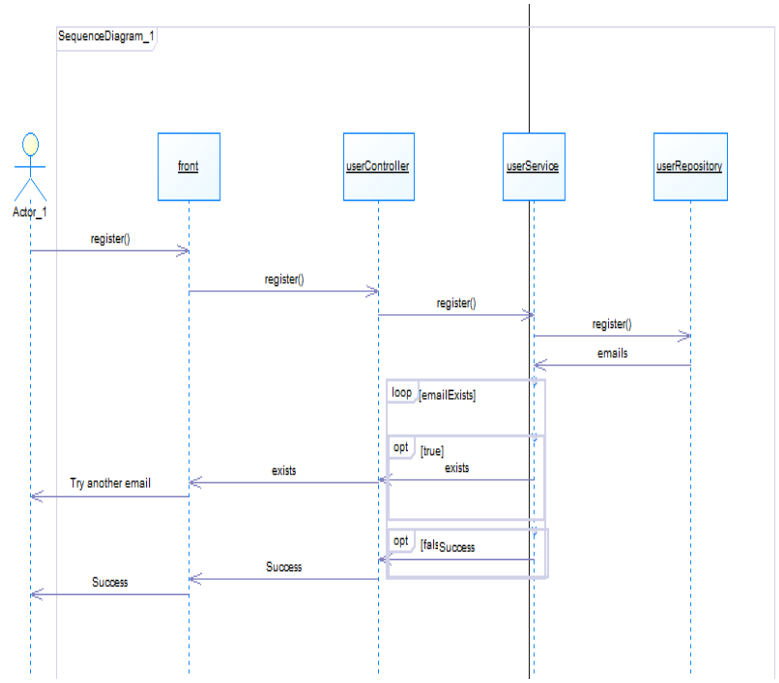
Rešenje: Opet slično, problem rešavamo postavljanjem anotacije @Transactional na metode klase UserService create() i findByEmail(). Pesimističko zaključavanje smo obezbedili u metodi za dobavljanje svih korisnika findAllLock()

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
@Query(value="select * from client where email=:email and password=:password",nativeQuery=true)
Client getClientByEmailAndPassword(String email, String password);

@Transactional
public Client createClient(Client client) throws Exception {

    try {
```

Dijagrami:




Front
QuickReservationController
QuickReservationService
QuickReservationRepository
Actor_1
See Reservations on Action
GetAllAvailable()
GetAllAvailable()
GetAllAvailable()
reservations
reservations
reservations
reservations
Book(reservation)
getReservation()
getReservation()
getReservation()
reservation
reservation
opt
[StillAvailable]
Update()
Update()
updatedReservation or exception
updatedReservation or exception
updatedReservation or exception
opt
[exception]
Exception
opt
[reservation]
Success and email


SequenceDiagram_1
front
userController
userService
userRepository
Actor_1
register()
register()
register()
register()
emails
loop
[emailExists]
opt
[true]
exists
exists
Try another email
opt
[fals
Success
Success
Success