4.4 Konkurentni pristup resursima u bazi

KONFLIKT 1

Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta u isto (ili preklapajuće) vreme

Jedan od mogućih problema jeste situacija u kojoj više korisnika istovremeno pokuša da rezerviše isti entitet. Ukoliko bi oba korisnika za isto, ili barem preklapajuće vreme odradila proveru dostupnosti jednog entiteta, obe provere bi vratile validan rezultat, te bismo bili u riziku dualne zauzetosti jednog objekta, ili avanture. U idealnoj situaciji, ovaj problem bio bi regulisan time što bi samo jedan korisnik uspešno izvršio rezervaciju entiteta, dok bi drugi bili obavešteni o grešci. Kako se ovaj problem javlja na tri ekvivalentna mesta: prilikom rezervacije, vikendica, brodova i avantura, dijagram toka prikazaćemo za prvi slučaj.

Rjesenje:

Problem rešavamo postavljanjem pesimističkog zaključavanja na get metode repozitorijuma za vikendice, brodove i avanture. Dodatno, u klasama ReservationService, Home/ Boat/ Adventure Service, su logike zakazivanja uokvirene **try/catch** blokom, a metode označe ***@Transactional*** anotacijom.

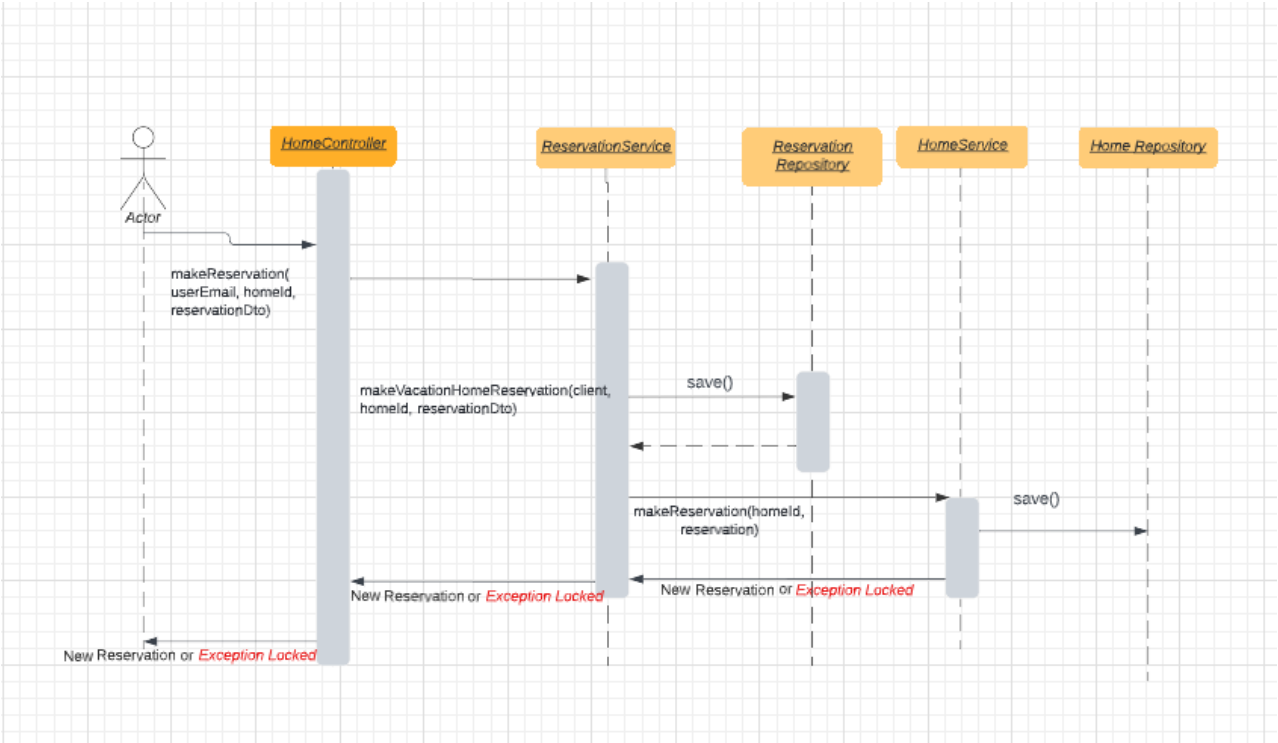
```
@Lock(LockModeType.PESSIMISTIC_READ)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
VacationHome findLockedById(Long id);
```

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

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

```java
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
public void makeReservation(Long homeId, Reservation reservation) {
    try {
        //Ako pokusa ne daj da mijenja tj da doda reservation
        VacationHome home = vacationHomeRepository.findLockedById(homeId);
        home.getReservations().add(reservation);
        vacationHomeRepository.save(home);
        updateAvailability(reservation.getStartDate(), reservation.getEndDate(), homeId);
        homeOwnerService.updatePoints(home.getHomeOwner(), reservation.getPrice());
        earningsService.saveEarnings(reservation, home.getHomeOwner().getEmail(), home.getHomeOwner().getRank());

    } catch (PessimisticLockingFailureException exception) {
        throw exception;

    }
}
```

```java
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
public Reservation makeVacationHomeReservation(Client client, Long homeId, ReservationDto reservationDto) {
    Reservation reservation = makeNewReservation(client, reservationDto);

    try {
        Reservation newReservation = reservationRepository.save(reservation);
        homeService.makeReservation(homeId, newReservation);
        clientService.updatePoints(client, newReservation.getPrice());

        return newReservation;

    } catch (PessimisticLockingFailureException e) {
        System.out.println("Pessimistic lock: Vacation Home");
        throw  e;
    }
}
```



Slika 1

KONFLIKT 2

Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vreme

Akcijske ponude su dostupne svim prijavljenim klijentima za brzu rezervaciji pa se moze lako je desiti da dva klijenta u isto, ili približno isto vreme pokušaju da rezervišu istu ponudu. Takođe je moguće da isti klijent pokuša da napravi iz dva različita prozora rezervaciju istog entiteta, ili 2 različita čija su vremena trajanja preklapajuća ili ista. Oba slučaja dovela bi do nekonzistentosti baze.

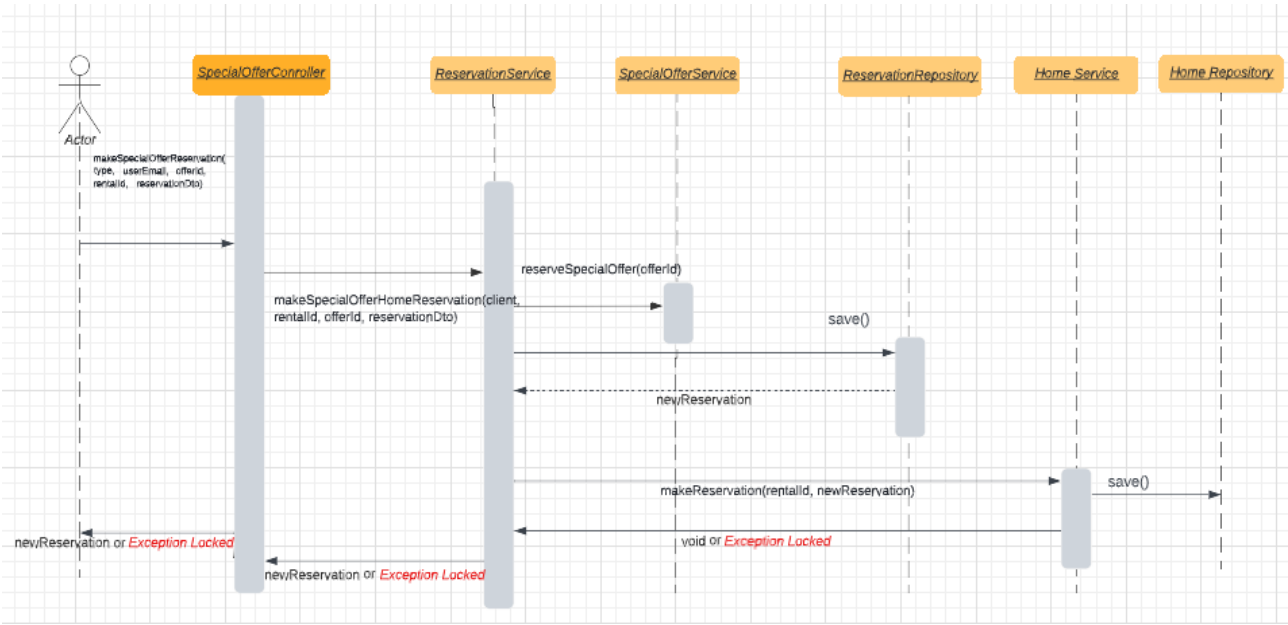
Rješenje:

Problem se odnosi na Brod, Vikendicu i Avanturu. Vodeci racuna da je rješenje isto za sve pomenute entitete, prikazaće se rješenje za Vikendicu. Rješavanje počinjemo postavljanjem anotacije ***@Transactional*** na metodu klase ReservationService **makeSpecialOfferHomeReservation*()***. Takođe, metodu klase SpecialOffer – **reserveSpecialOffer** je potrebebno označiti kao ***@Transactional()*** . Pomenuta metoda poziva na pesimističko zaključavanje resursa – Special Offer. Pesimističko zaključavanje sa anotacijom ***PESSIMISTIC_WRITE*** sprečiava da 2 ili više korisnika istovremeno čitaju podatke o dostupnim akcijama u svrhu zakazivanja istih. U okviru ***try/catch*** bloka postavljen je deo koda koji barata sa postavljenim zaključavanjem.

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
public Reservation makeSpecialOfferHomeReservation(Client client, Long rentalId, Long offerId, ReservationDto reservationDto) {
    Reservation reservation = makeNewSpecialOfferReservation(client, reservationDto);

    try {
        specialOfferService.reserveSpecialOffer(offerId);
        Reservation newReservation = reservationRepository.save(reservation);
        homeService.makeReservation(rentalId, reservation);
        clientService.updatePoints(client, newReservation.getPrice());

        return newReservation;

    } catch (Exception e) {
        System.out.println("Pessimistic lock: SpecialOfferReservation");

    }
    return null;

}
```

```java
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
public void reserveSpecialOffer(Long offerId) {
    try {
        SpecialOffer specialOffer = specialOfferRepository.findLockedById(offerId);
        specialOffer.setUsed(true);
        specialOfferRepository.save(specialOffer);
    } catch (Exception e) {
        System.out.println("Pessimistic lock: SpecialOffer");
        throw e;
    }
}
```

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



KONFLIKT 3

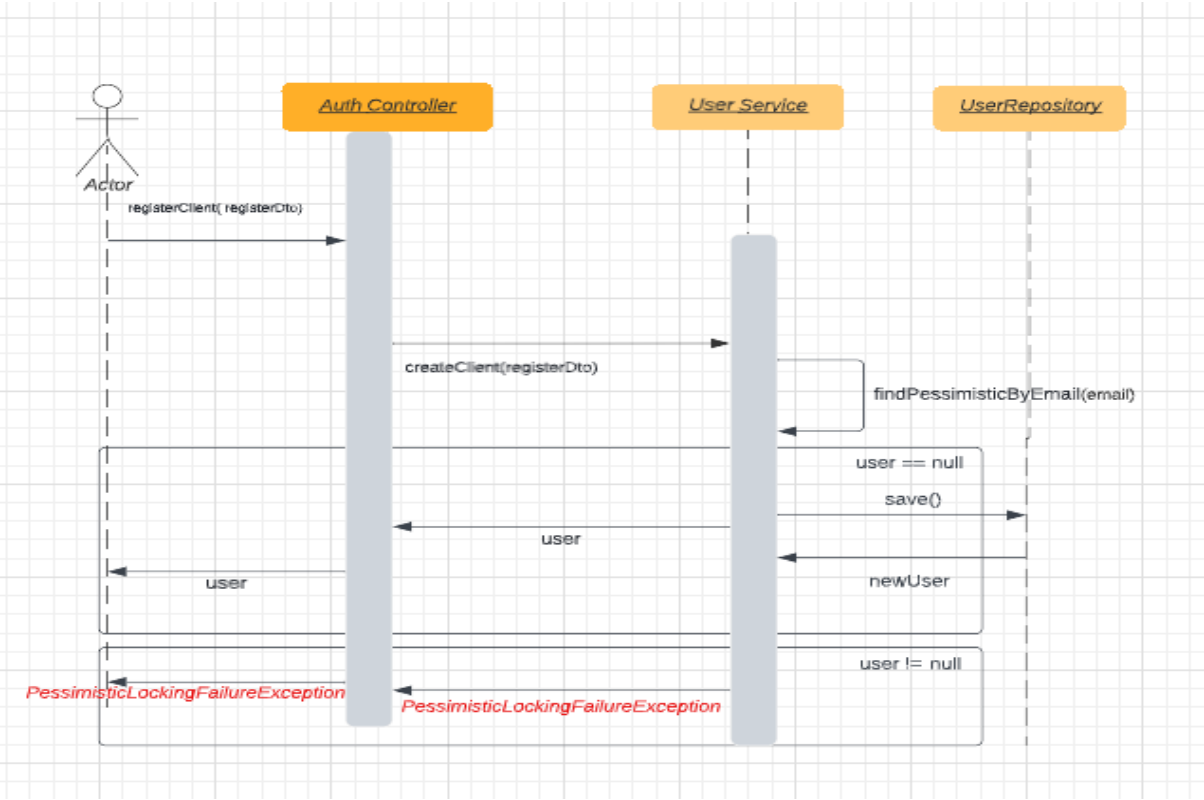
Više klijenata ne može da pošalje zahtev za registraciju u isto vreme sa istom mejl adresom

Kada korisnik pokuša da kreira nalog, najpre se izvršava provera unikatnosti unešene mejl adrese. U slučaju da je ista već zauzeta, korisnku se na stranici forme ispisuje poruka o

grešci, i neophodno je promeniti mejl adresu. U situaciji da više korisnika istovremeno pokrene ovaj zahtev, postoji mogućnost da oba zahteva prođu, što bi dovelo do brojnih greški u sistemu.
Rješenje:
Ova konfliktna situacija je riješena pesimističkim zaključavanjem. Zaključana je metoda za dobavljanje svih korisnika **findAllPesimistic()**. Provjerava se jedinstvenost mejla u metodi **findPessimisticByEmail()**. Pored toga metode koje su pozvane su anotirane anotacijom ***@Transactional*** i implementacija je uokvirena **try/catch** blokovima koji će vratiti **PessimisticLockingFailureException** u slučaju kada se isti mejl pronađe.

```
@Transactional
public User findPessimisticByEmail(String email) {
    for (User u : userRepository.findAllPessimistic()) {
        if (u.getEmail().equals(email)) {
            return u;
        }
    }
    return null;
}

@Override
@Transactional
public User createClient(RegisterDto registerDto) throws ResourceConflictException, MessagingException {
    User found = findPessimisticByEmail(registerDto.getEmail());
    if (found != null) {
        throw new PessimisticLockingFailureException("User with email exists");
    }
    Client user = RegistrationMapper.mapToClient(registerDto);
    clientService.registerClient(user);
    return user;
}
```

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("select u from User u")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
List<User> findAllPessimistic();
```



Maja Blagić RA110/2018