Konkurentni pristup

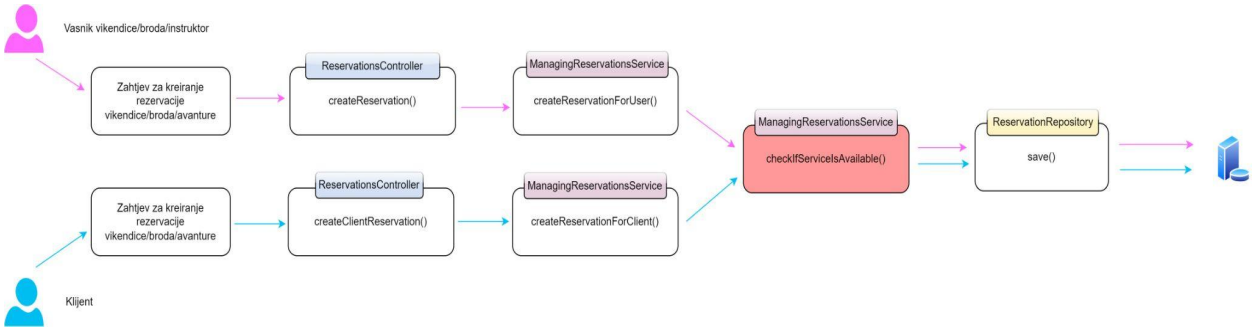
1. *Vlasnik vikendice/broda ili instruktor ne može da napravi rezervaciju u isto vrijeme kad i drugi klijenti*

Opis konfliktne situacije

Prvi korisnik ima ulogu vlasnika neke vikendice/broda ili ulogu instruktora i želi da napravi rezervaciju za nekog klijenta. Drugi korisnik je klijent koji želi da napravi rezervaciju za isti entitet kao i prvi korisnik u istom vremenskom periodu. Prilikom izvršavanja zahtjeva u oba slučaja se vrši provjera da li je entitet dostupan u zadatom periodu. Kako u trenutku provjere ni jedna od dvije navedene rezervacije nije kreirana, u oba slučaja biće dozvoljen upis u bazu. Ovo dovodi do nevalidnog stanja u bazi jer ćemo imati dvije rezervacije za termine koji se preklapaju.

Grafički prikaz konfliktne situacije

Do same konfliktne situacije dolazi prilikom provjere dostupnosti vikendice/broda/instruktora (označeno crvenom bojom).



Predloženo rješenje konfliktne situacije

Kako se u ovom slučaju ne radi o pokušaju izmjene podataka već dodavanja novih, rješenje se implementira kroz pessimistic lock. Koristimo metodu *findLockedByBookingServiceId* koja pronalazi sve rezervacije po ključu usluge za koju su kreirane i zaključava ih. Kao tip zaključavanja korišten je PESSIMISTIC_WRITE koji ne dozvoljava čitanje zaključanih rezervacija.
Metode *createReservationForUser* i *createReservationForClient* u *ManagingReservationsService* su označene kao transakcione. Kada se u metodama pozove *checkIfServiceIsAvailable* za dobavljanje rezervacija koristiće se metoda *findLockedByBookingServiceId* iz repozitorijuma.

```java
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="2000")})
List<Reservation> findLockedByBookingServiceId(int serviceId);
```

```java
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public Reservation createReservationForUser(ReservationDTO reservationDTO) throws Exception {
    User client = userRepository.getById(reservationDTO.getUserId());
    Reservation lastingReservation = getLastingReservationForUser(client, reservationDTO.getServiceId());
    Reservation newReservation = getReservation(reservationDTO);
    if (lastingReservation == null)
        return null;

    if (!checkIfServiceIsAvailable(reservationDTO.getReservationStart(), reservationDTO.getReservationEnd(),
            reservationDTO.getServiceId())) {
        return null;
    }
```

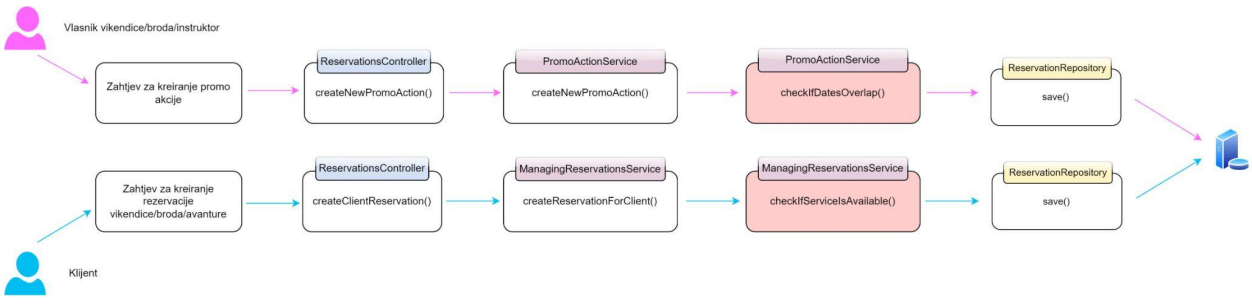
2. *Vlasnik vikendice/broda ili instruktor ne može da napravi novu promo akciju u isto vrijeme kad i drugi klijent vrši rezervaciju*

Opis konfliktne situacije

Prvi korisnik ima ulogu vlasnika neke vikendice/broda ili ulogu instruktora i želi da napravi novu promo akciju. Drugi korisnik je klijent koji želi da napravi rezervaciju za isti entitet za koji prvi korisnik pravi akciju u istom vremenskom periodu. Kako bi se kreirala nova promo akcija provjerava se zauzetost vikendice u datom periodu. Takođe prilikom kreiranja rezervacije za klijenta provjerava se da li je vikendica/brod/instruktor dostupan. Kako se ove dvije situacije odvijaju paralelno ni jedna od njih ne zna za promjene sa druge strane, tako da će upisivanje entiteta biti moguće. Ovo dovodi do nevalidnog stanja u bazi jer će postojati promo akcija koju ostali klijenti mogu da rezervišu i već napravljena rezervacija sa periodom trajanja koji se preklapa.

Grafički prikaz konfliktne situacije

Do same konfliktne situacije dolazi prilikom provjere dostupnosti vikendice/broda/instruktora (označeno crvenom bojom).



Predloženo rješenje konfliktne situacije

Kao i u prethodnom slučaju, koristićemo metodu *findLockedByBookingServiceId*. Metoda *createNewPromoAction* u PromoActionService je označena kao transakciona i u njoj se poziva *checkIfDatesOverlap* gdje dobavljaju rezervacije pomoću *findLockedByBookingServiceId*.

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW, rollbackFor = Exception.class)
public PromoAction createNewPromoAction(PromoActionDTO actionDTO) {
    if (checkIfDatesOverlap(actionDTO.getStartDate(), actionDTO.getEndDate(), actionDTO.getBookingServiceId())) {
        return null;
    }
```

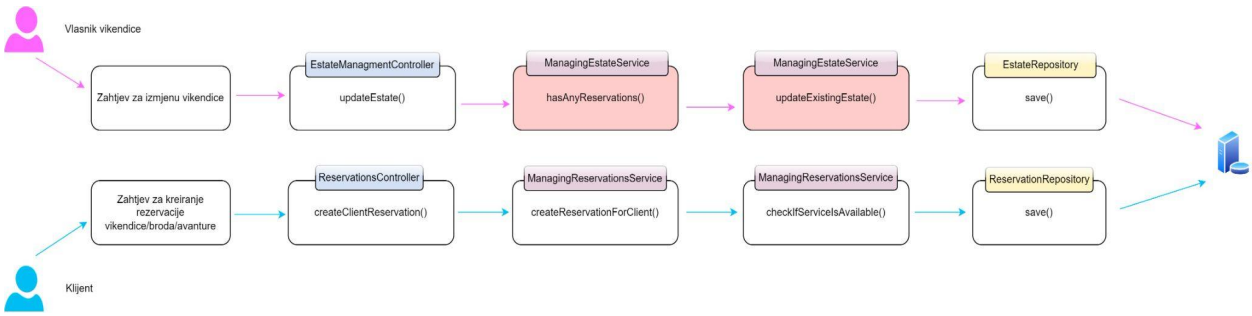
```
public boolean checkIfDatesOverlap(Date start, Date end, int serviceId) {
    List<Reservation> existingReservations = reservationRepository.findLockedByBookingServiceId(serviceId);
```

3. *Vlasnik vikendice ne može da napravi izmjenu vikendice u trenutku kada klijent pokušava da je rezerviše*

Opis konfliktne situacije

Prvi korisnik ima ulogu vlasnika neke vikendice i želi da napravi izmjene informacija o vikendici. Drugi korisnik je klijent koji želi da napravi rezervaciju za isti entitet za koji prvi korisnik pravi izmjenu u istom vremenskom periodu. Vlasnik može da napravi izmjene entiteta samo ukoliko on nije rezervisan. Klijent nesmetano može da izvrši rezervaciju vikendice ukoliko se period koji želi ne poklapa sa već postojećim rezervacijama. U isto vrijeme, na drugoj strani vlasnik nesmetano pravi izmjene vikendice i čuva ih u bazi. Klijent je u ovom slučaju izvršio rezervaciju za entitet sa starim podacima.

Grafički prikaz konfliktne situacije



Predloženo rješenje konfliktne situacije

I u ovom slučaju vršimo zaključavanje prilikom dobavljanja svih rezervacija pomoću *findLockedByBookingServiceId* prilikom provjere da li postoje zakazane rezervacije. Metoda *updateExistingEstate* u ManagingEstateService je označena kao transakciona i prilikom provjere *hasAnyReservations* koristiće se *findLockedByBookingServiceId*.

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW, rollbackFor = Exception.class)
public void updateExistingEstate(NewEstateDTO estateDTO, Estate existingEstate) {
    if (!hasAnyReservations(existingEstate))
        return;
```

Anđela Đurić RA28/2018