Konkurentni pristup – Lea Kalmar RA 130/2018

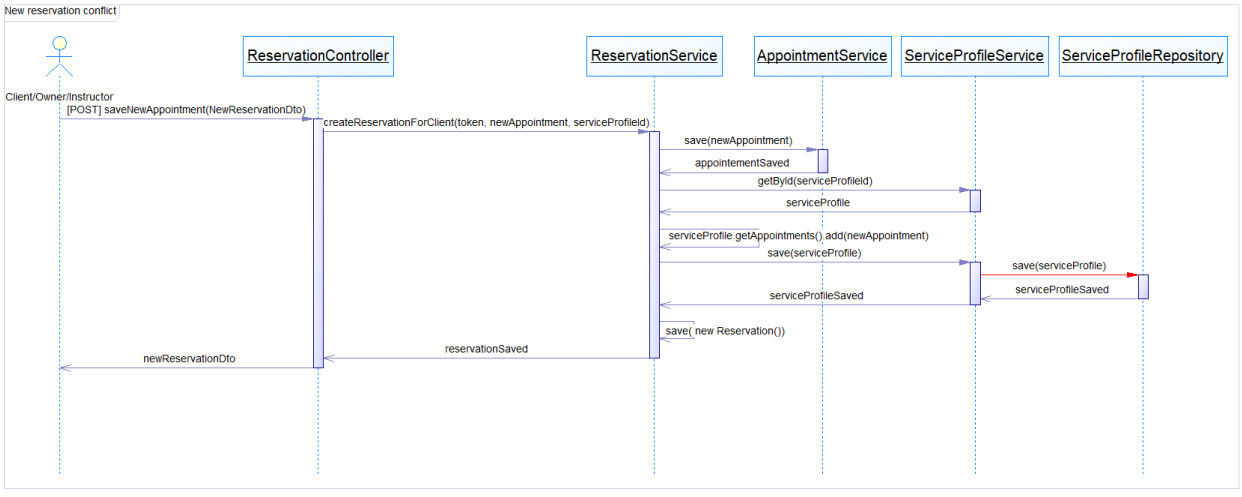
1. Vlasnik vikendice/broda ili instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent

Opis konfliktne situacije

Pretpostavka je da imamo vlasnika ili instruktora koji u trenutku dok traje neki termin želi da rezerviše novi termin za korisnika za kog trenutno traje termin. Vlasnik/instruktor vrši odabir termina i popunjava potrebne podatke gde je ustanovljeno da je za unešene podatke moguće izvršiti rezervaciju. U međuvremenu, neki drugi klijent pokušava da rezerviše isti entitet za period koji se u nekoj meri poklapa sa periodom koji je unet od strane vlasnika/instruktora. Takođe, i klijent dobija informaciju da je rezervaciju moguće izvršiti. Kako je u oba slučaja entitet slobodan za rezervaciju, oba korisnika će izvršiti rezervaciju i na taj način narušiti konzistentnost baze. Narušiće se ograničenje da se entitet ne može rezervisati ukoliko za njega već postoji rezervacija jer ćemo u ovom slučaju sačuvati obe rezervacije iako se njihovi datumi međusobno u nekoj meri preklapaju.

U konkretnoj implementaciji, do problema dolazi kada zahtevi obe pomenute strane dođu do poziva funkcije ***save*** iz ***ServiceProfilRepository***-a. U tom trenutku kao krajnji proizvod u samom entitetu imamo dve rezervacije koje se međusobno preklapaju i narušavaju pravilo da se jedan entitet ne može rezervisati ukoliko za taj period već postoji rezervacija.

Grafički prikaz konfliktne situacije



Rešenje konfliktne situacija

Konfliktnu situaciju je moguće rešiti ukoliko iskoristimo mehanizam Optimistic Lock-a. Unutar klase ServiceProfile dodaje se polje version koji omogućava praćenje izmene entiteta. Svaka izmena nad entitetom će prouzrokovati inkrement version polja. Kada dođe do situacije da jedna nit želi da izvrši izmenu entiteta, a kod sebe ima „zastarele“ podatke, tj. entitet koji ima staru verziju, transakcija se neće izvršiti.

Nedostatak ovakve implementacije je potpuno odbacivanje svih transakcija koje su vezane za dati entitet (red u tabeli) ukoliko je u međuvremenu došlo do bilo kakve njegove izmene, iako ta izmena ne mora uopšte biti vezana za konflikt koji je u ovom slučaju obrađen.

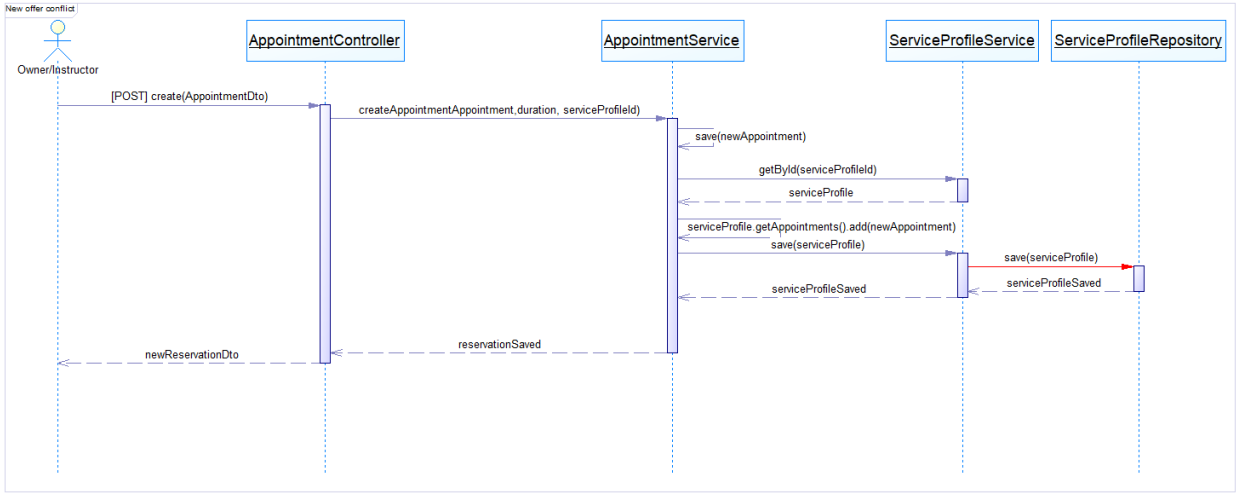
```java
@Entity
@Inheritance(strategy = TABLE_PER_CLASS)
public abstract class ServiceProfile {
    @Id
    @SequenceGenerator(name = "mySeqGenV1", sequenceName = "mySeqV1", initialValue = 1, allocationSize = 1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "mySeqGenV1")
    private Integer id;

    @Version
    @Column(nullable = false)
    private Integer version;
```

2. Vlasnik vikendice/broda ili instruktor ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta

Opis konfliktne situacije

Kao i u prethodnom konfliktu, imamo slučaj gde i vlasnik/instruktor unosi određene datume kako bi kreirao specijalnu ponudu i u trenutku provere ovih podataka, entitet je neizmenjen i slobodan za rezervaciju. Isto važi i za klijenta koji isti entitet želi da rezerviše za datum koji se u nekoj meri poklapa sa datumom koji je vlasnik/instruktor uneo kod sebe u formi. Oba zahteva su validna u trenutku slanja te će oba izvršiti izmenu nad istim entitetom i time dovesti do nekonzistentnosti u bazi podataka, gde ćemo za isti entitet imati rezervaciju i specijalnu ponudu čiji datumi se u nekoj meri poklapaju.

Grafički prikaz konfliktne situacije



Rešenje konfliktne situacija

Kao i u prethodnom konfliktu poziva se ista funkcija ***save*** iz **ServiceProfileRepository**-a. Kako su upitanju iste metode, rešavanjem prve konfliktne situacije rešena je i ova konfliktna situacija.

Projekat je izmodelovan na način da *ServiceProfile* koji predstavlja sve entitete koji se mogu rezervisati(*Boat*, *VacationHome*, *FishingAdventure*), u sebi sadrži listu *Appointment*-a. Klasa *Appointment* jeste model nekog termina koji nije još uvek rezervisan te ga u osnovi koristimo za modelovanje specijalnih ponuda. Međutim kako se unutar jedne rezervacije nalaze isti podaci kao

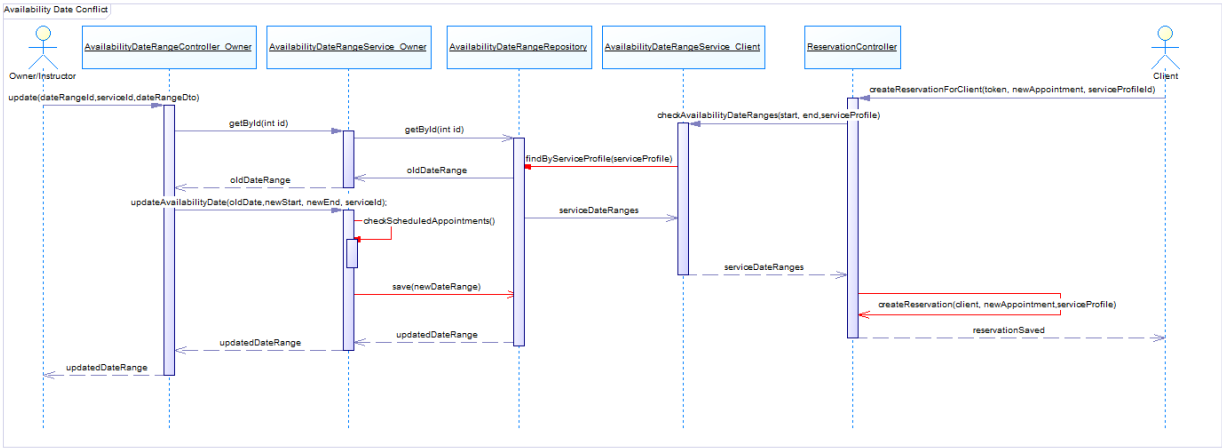
unutar specijalne ponude i još po neki, klasa *Reservation* opisuje rezervaciju ali u sebi sadrži klasu *Appointment*-a gde su sadržane sve osnovne informacije o terminu koji se rezerviše. Iz ovog razloga prilikom vršenja rezervacije ili kreiranja specijalne ponude uvek se ažurira lista klase *Appointment* unutar *ServiceProfile*-a o kom je reč.

3. vlasnik/instruktor ne može da izvrši izmenu perioda dostupnosti u isto vreme kada klijent želi da zakaže rezervaciju za taj period

Opis konfliktne situacije

Vlasnik/instruktor želi da izmeni dostupnost vikendice tako što će određeni datumi koji su do tada bili dostupni nakon izmene ipak biti nedostupni. Ukoliko klijent u datom trenutku pokuša da uradi rezervaciju za neki od datuma koje vlasnik/instruktor želi da izbaci, klijent će uspešno napraviti rezervaciju, dok će vlasnik/instruktor uspešno izmeniti period dostupnosti vikendice. U ovom slučaju kreiraće se rezervacija za period kada entitet nije dostupan, što narušava konzistentnost baze.

Grafički prikaz konfliktne situacije



Rešenje konfliktne situacija

Prilikom rešavanja datog konflikta odabran je pristup pesimističkog zaključavanja. Na ovaj način omogućeno je da pratimo kada se dati entitet čita i u tom trenutku, taj resurs zaključamo. Konflikt je proizilazio iz samog čitanja liste koja sadrži dostupnosti nekog entiteta, i kasnije sprovođenje određene logike od strane klijenata. Pomoć pesimističnog zaključavanja možemo da zabranimo da se dati entitet čita sve dok se operacija ne izvrši, kada će poziv koji je do tada čekao moći da završi svoj posao sa validnim podacima.

Mana ovakvog pristupa jeste samo zaključavanja redova koji se odnose određeni entitet što sprečava sprovođenje i drugih akcija koje ne moraju prouzrokovati problem zbog kog je pesimistično zaključavanje uvedeno.

```
@Repository
public interface AvailabilityDateRangeRepository extends JpaRepository<AvailabilityDateRange, Integer> {

    @Lock(LockModeType.PESSIMISTIC_READ)
    @QueryHints({ @QueryHint(name = "javax.persistence.lock.timeout", value = "0") })
    public List<AvailabilityDateRange> getAllByServiceProfileId(Integer id);
}
```