KONFLIKTNE SITUACIJE – konkurentni pristup resursima u bazi

Rješavane su sledeće konfliktne situacije:

1. Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta
2. Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji
3. Jednu reviziju koju kreira klijent za vikendicu/brod ili čas pecanja, može da odobri ili odbije samo jedan administrator
4. Klijent ne može da ažurira svoje lične podatke koristeći više uređaja istovremeno

Rješenja konfliktnih situacija:

1. Konfliktna situacija

Ukoliko bi dva klijenta istovremeno odabrala dva ista ili preklapajuća termina za rezervaciju, pretraga dostupnosti termina bi obojici klijenata vratila entitete koji su dostupni u tom vremenskom periodu. Ako se klijenti odluče za isti entitet i potvrde rezervaciju, čuvanje te dvije rezervacije u bazi dovodi do nevalidnih podataka.

Ova konflikta situacije je riješena tako što je iznad metoda za rezervaciju stavljena anotacija @Transactional, a potom unutar metoda izvršeno rukovanje izuzecima, kako bi se spriječilo da baza podataka dođe u nevalidno stanje.

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public BoatReservation createReservation(ReservationDTO dto) throws Exception{
```

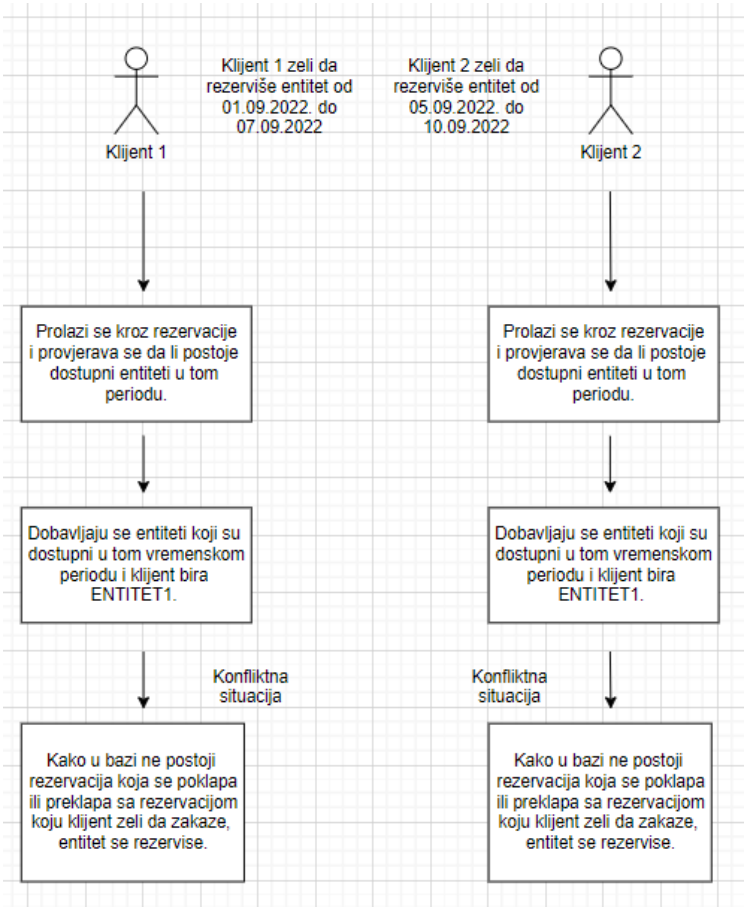
```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public CottageReservation createReservation(ReservationDTO dto) throws Exception{
```

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public FishingLessonReservation createReservation(ReservationDTO dto)  throws Exception{
```

Korišćeno je pesimističko zaključavanje.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select b from Boat b where b.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="10000")})
public Boat findLockById(@Param("id")Long id);
```

Dijagram:



2. Konfliktna situacija:

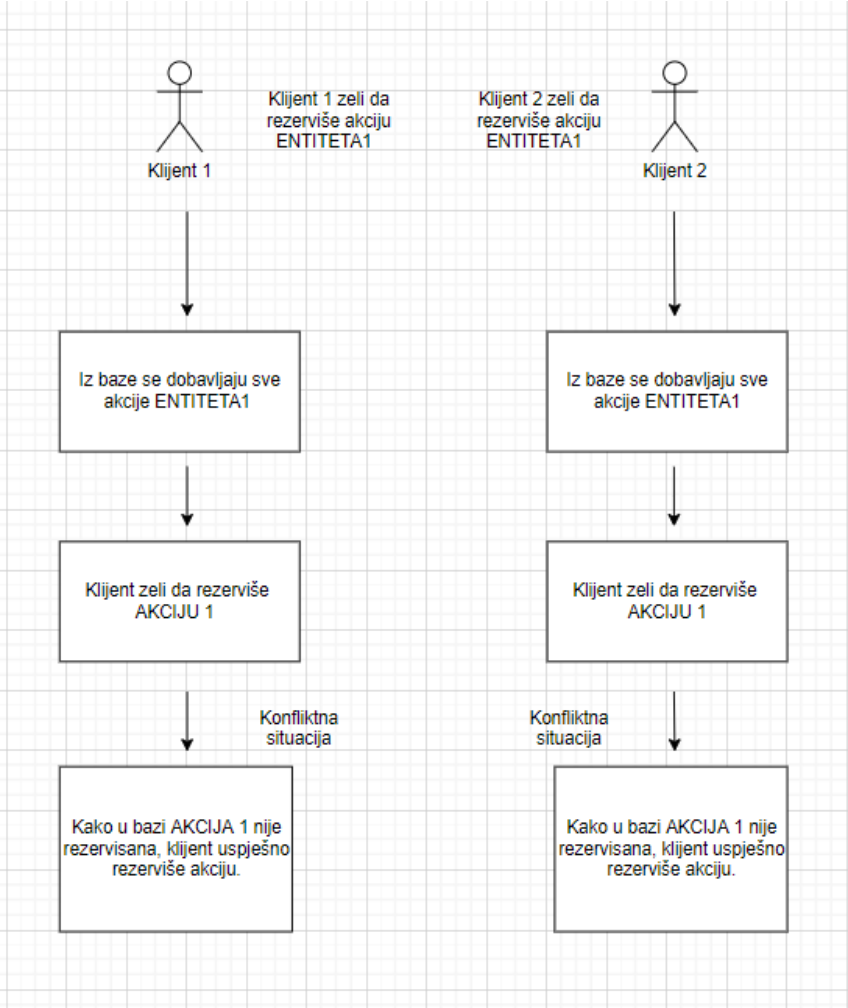
Ukoliko dva klijenta žele da zakažu brzu rezervaciju istog entiteta, prilikom odabira entiteta, prikazaće im se iste brze rezervacije. Ukoliko se odluče za istu rezervaciju i rezervišu entitet, doći će do situacije da klijent pokušava da rezerviše entitet koji je prethodni klijent već rezervisao.

Ova konfliktna situacija je riješena tako što je iznad metoda za brzu rezervaciju stavljena anotacija @Transactional, a potom unutar metoda izvršeno rukovanje izuzecima. Takođe je korišćeno pesimističko zaključavanje.

```java
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public Client clientReservation(QuickClientReservationDTO dto)  throws Exception{

@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select r from QuickCottageReservation r where r.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="10000")})
public QuickCottageReservation findLockById(@Param("id")Long id);
```

Dijagram:



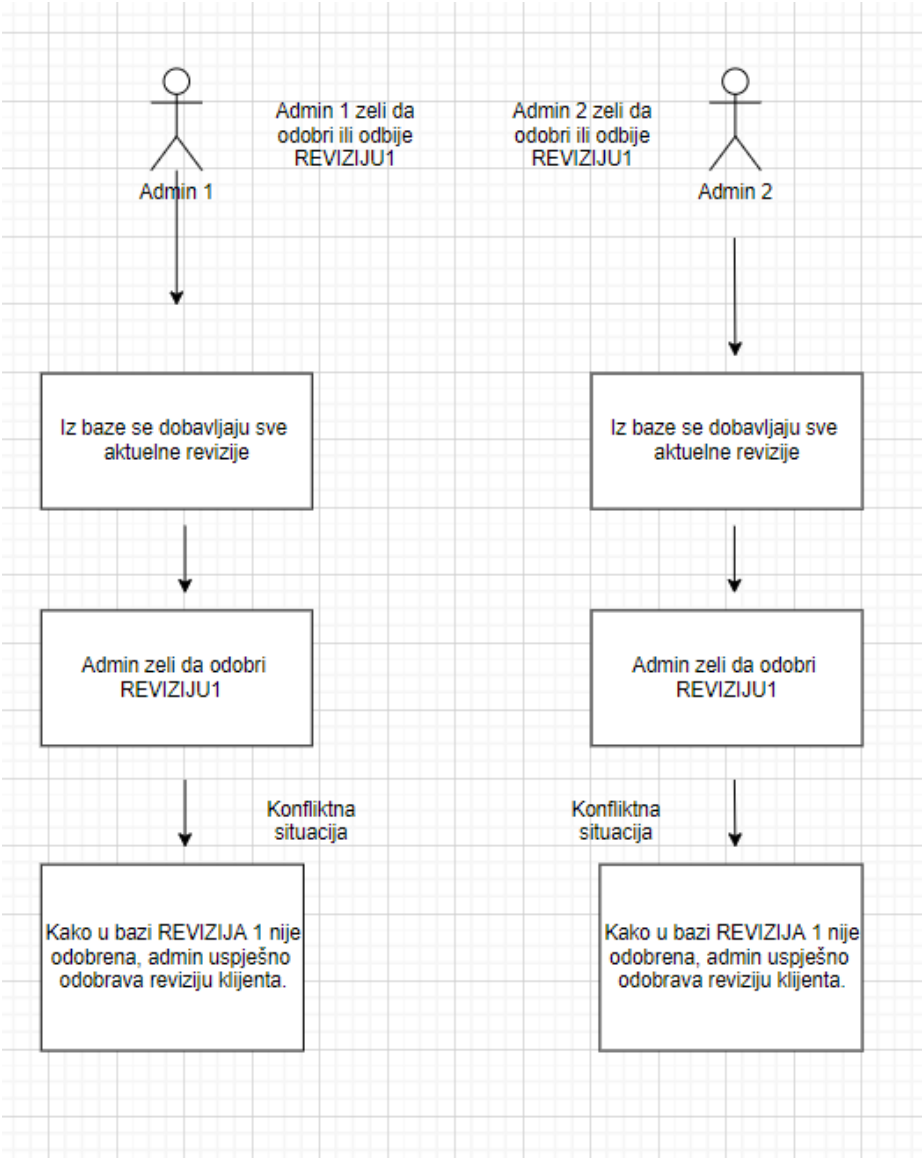
3. Konfliktna situacija:

Ukoliko dva administratora pregledaju istu reviziju koju je napravio klijent, može doći do situacije da oba istovremeno žele da odbiju ili odobre reviziju. U tom slučaju

će doći do konfliktne situacije jer će jedan od administratora pokušati da mijenja status revizije istovremeno kad o drugi.

Problem je riješen upotrebom anotacije @Transactional i pesimističkog zaključavanja kao i u prethodnom primjeru. Unutar metode je izvršena provjera da li je status revizije već promijenjen u međuvremenu.

Dijagram:

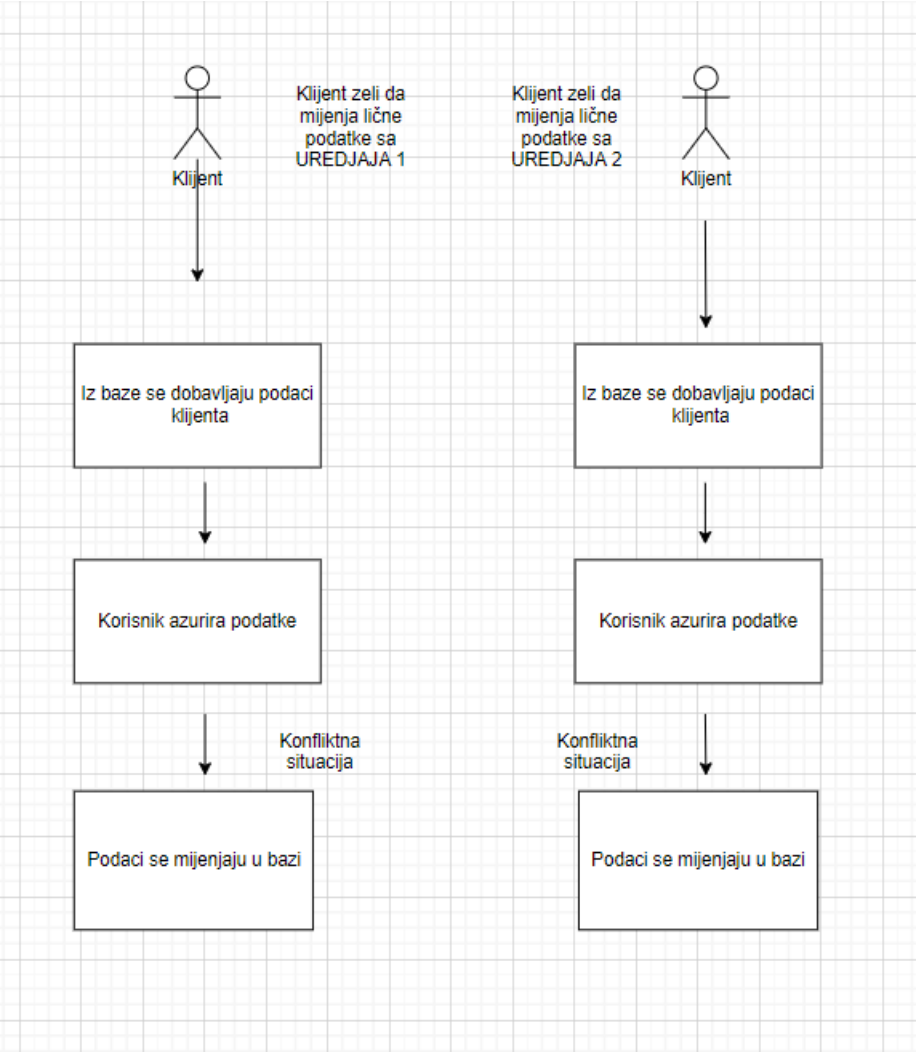


4. Konfliktna situacija:

Klijent može sa dva uređaja istovremeno da mijenja lične podatke na profilu. To bi dovelo do konfliktne situacije i pojave nevalidnih podataka u bazi.

Konfliktna situacije je takođe riješena upotrebom anotacije @Transactional i pesimističkog zaključavanja.

Dijagram:



Slađana Čolaković RA32/2018