Konkurentan pristup resursima u bazi

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

Realan problem koji se može dogoditi korišćenjem mogućnosti za rezervisanje entiteta je istovremeno zakazivanje jednog entiteta od strane više korisnika u isto vreme. Ovo može dovesti do raznih problema u daljem radu sistema. Da bi se ovakva situacija sprečila, neophodno je osigurati ovaj deo sistema na neki način.

Imajući u vidu da se rezervacija sva tri tipa entiteta vrši na isti način, servisna metoda čiji je to zadatak je anotirana sa *@Transactional* anotacijom i obmotana try-catch blokom.

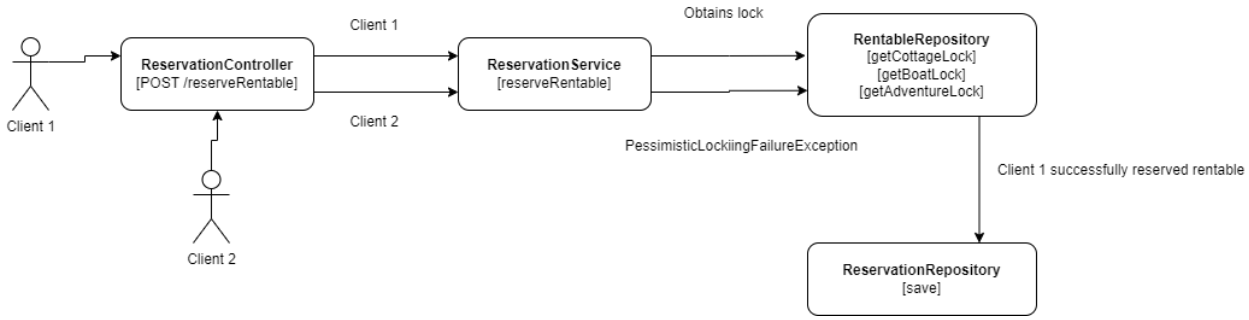
Na nivou repozitorijuma, metode za dobavljanje sva tri tipa entiteta su označene anotacijom *@Lock(LockModeType.PESSIMISTIC_WRITE)*.

```
    @Lock(LockModeType.PESSIMISTIC_WRITE)
    @Query("select cottage from Cottage cottage " +
            "where cottage.id = :id")
    @QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
    Cottage getCottageLock(Long id);

    @Lock(LockModeType.PESSIMISTIC_WRITE)
    @Query("select boat from Boat boat " +
            "where boat.id = :id")
    @QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
    Boat getBoatLock(Long id);

    @Lock(LockModeType.PESSIMISTIC_WRITE)
    @Query("select adventure from Adventure adventure " +
            "where adventure.id = :id")
    @QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
    Adventure getAdventureLock(Long id);
}
```

Proces istovremene rezervacije entiteta kroz slojeve aplikacije prikazan je sledećim dijagramom:



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

Za razliku od regularne rezervacije, akcije se prvobitno pri kreiranju tretiraju kao rezervacije koje nemaju pridruženog klijenta. Imajući ovo u vidu, koristiće se drugačiji tip zaključavanja.

U servisnom sloju, koristi se *@Transactional* anotacija za metodu čiji je kod umotan u try-catch blok.

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
@Override
public void reserveAction(Long id, String customerUsername){
    try {
        Reservation reservation = reservationRepository.getReservationById(id);
```

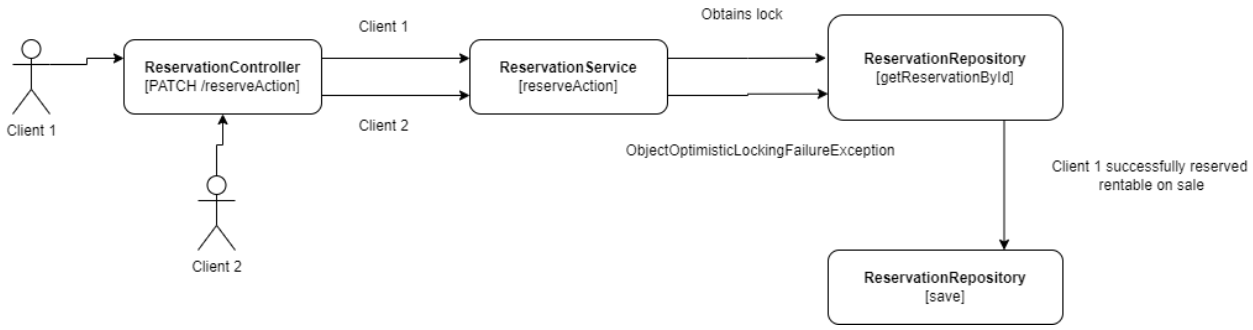
U sloju repozitorijuma, koristi se optimističko zaključavanje anotiranjem metode za dobavljanje rezervacije/akcije anotacijom *@Lock(LockModeType.OPTIMISTIC_FORCE_INCREMENT)*.

```java
@Lock(LockModeType.OPTIMISTIC_FORCE_INCREMENT)
Reservation getReservationById(Long id);
```

Takođe, da bi se pratilo stanje rezervacija u ovom kontekstu, u modelskom sloju domena se rezervaciji dodeljuje novo polje version sa anotacijom *@Version*.

```java
@Version
@Column(name = "optlock", columnDefinition = "integer DEFAULT 0", nullable = false)
private Long version = 0L;
```

Proces istovremene rezervacije entiteta na akciji kroz slojeve aplikacije prikazan je sledećim dijagramom:



3. Više istovremenih korisnika ne mogu da kreiraju nalog sa istim korisničkim imenom

Imajući u vidu da se korisničko ime koristi kao jedinstveni identifikator svakog korisnika, od ključnog značaja je obezbediti da se ne mogu kreirati dva različita korisnička naloga sa istim korisničkim imenima.

Ovo je realizovano korišćenjem *@Transactional* anotacije iznad servisnih metoda za dobavljanje korisnika po korisničkom imenu i čuvanje novog naloga. Ove dve metode se koriste u metodi za registraciju koja je takođe anotirana istom anotacijom i umotana u try-catch blok.

```
@Transactional
@Override
public RegisteredUser register(RegistrationDTO registrationDTO) {
    try {
        UserDetails existUser = loadUserByUsernameLocked(registrationDTO.getUsername());
```

```
@Transactional
@Override
public UserDetails loadUserByUsernameLocked(String s) throws UsernameNotFoundException {
    try {
        return userRepository.findByUsernameLocked(s);
```

```
@Override
@Transactional
public RegisteredUser save(RegisteredUser user) {
    try {
        return userRepository.save(user);
```

Na nivou repozitorijuma se koristi pesimističko zaključavanje pri dobavljanju korisnika po korisničkom imenu, na isti način kao i dobavljanje entiteta iz prve tačke.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select u from RegisteredUser u where u.username = :username")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
RegisteredUser findByUsernameLocked(String username);
```

Proces istovremene registracije korisnika sa istim korisničkim imenom prikazan je sledećim dijagramom: