4.4 Konkuretni pristup resursima u bazi

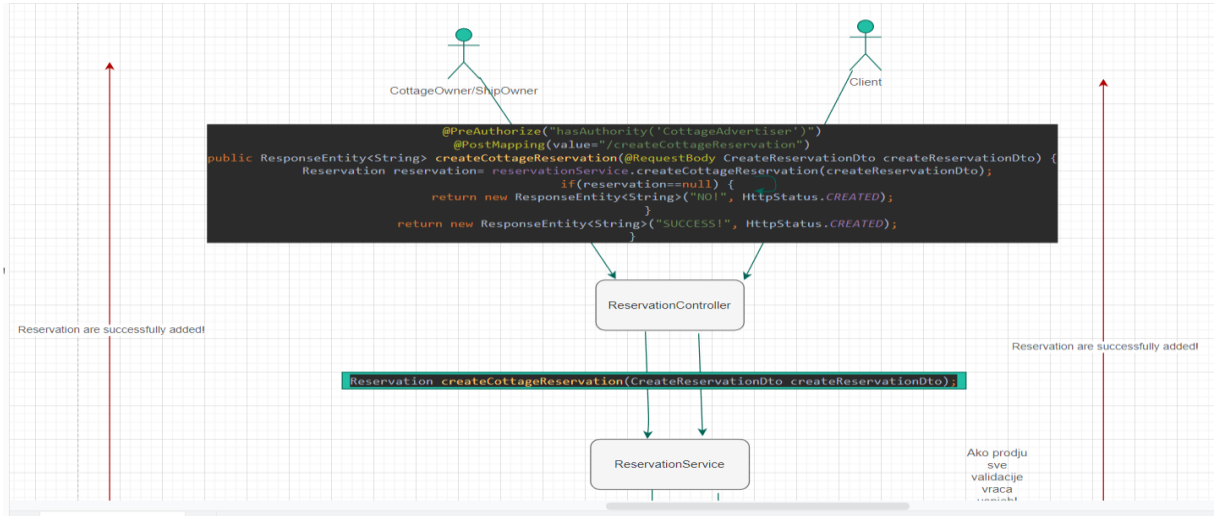
Student 2:Vedrana Mijatović

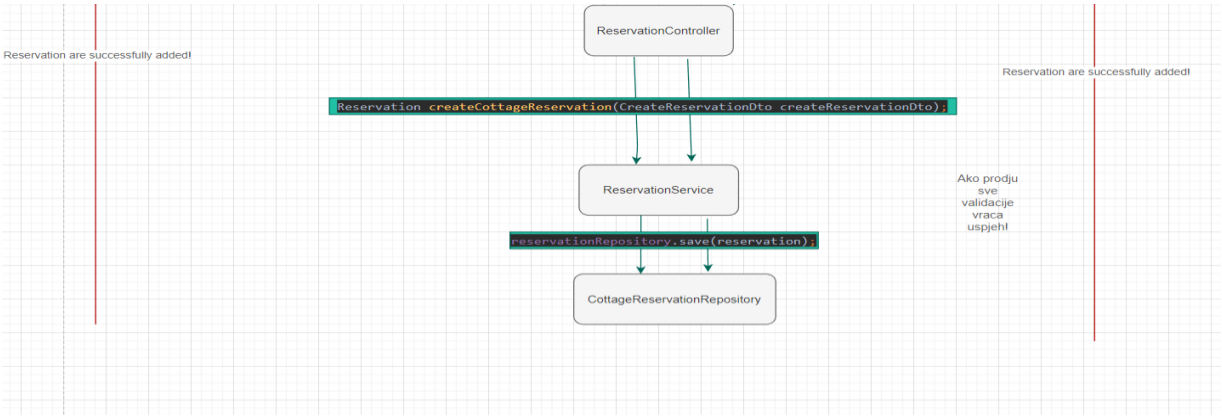
1.Vlasnik vikendice/broda ne može da napravi rezervaciju u isto vrijeme kad i drugi klijent

Opis problema:

Vlasnik vikendice/broda može u ime klijenta koji trenutno ima rezervaciju da napravi rezervaciju za vikendicu ili brod u budućnosti, dok takođe i drugi klijenti mogu sami da izvrše rezervaciju. Može da dođe do situacija kada obojica u isto vrijeme odaberu isti termin ili se termini preklope i izvrše rezervaciju istog entiteta. Pošto je obojici entitet u tom trenutku slobodan oni će uspješno izvršiti rezervaciju i rezervacija će nekom određenih validacija biti sačuvana u bazi. Ovim se narušava uslov da se ne može rezervisati već zauzet termin. Potrebno je riješiti ovu konfliktnu situaciju.

Tok zahtjeva:





Rješenje problema:

Za rješenje problema sam koristila pesimistično zaključavanje resursa. Rješenje je implementarino u ReservationServiceImpl, CottageReservationRepository i ShipReservationRepository. Za metode koje služe za kreiranje rezervacija za vikendicu, brod kao i na sve povezane metode van ovog service dodata je anotacija *Transactional*, tj sve su transakcione metode. Iz metode CottageRepository zaključavamo metodu findById kojom se dobavlja vikendica,te će vikendica biti zaključava sve dok se ne izvrši validacija odabranog termina i dati termin na sačuva u bazu podataka. Kao tip zaključavanja koristi se *PESSIMISTIC_WRITE* kojom se ne dozvoljava ni čitanje datog reda.Ukoliko pristigne novi zahtjev za dodavanje rezervacije,korisnik će dobiti poruku da je termin već zauzet. Ovim je riješena konfliktna situacija i onemogućeno rezervisanje istog termina u isto vrijeme.

Zaključavanje je uređeno u repozitorijima CottageRepository and ShipRepository i prikazano je na fotografiji.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Cottage c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Cottage findCottageById(Long id);
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Ship c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Ship findShipById(Long id);
```

Dodavanje anotacije na metodama u ReservationServiceImpl.

```
@Override
@Transactional
public Reservation createShipReservation(CreateReservationDto createReservationDto) {
```

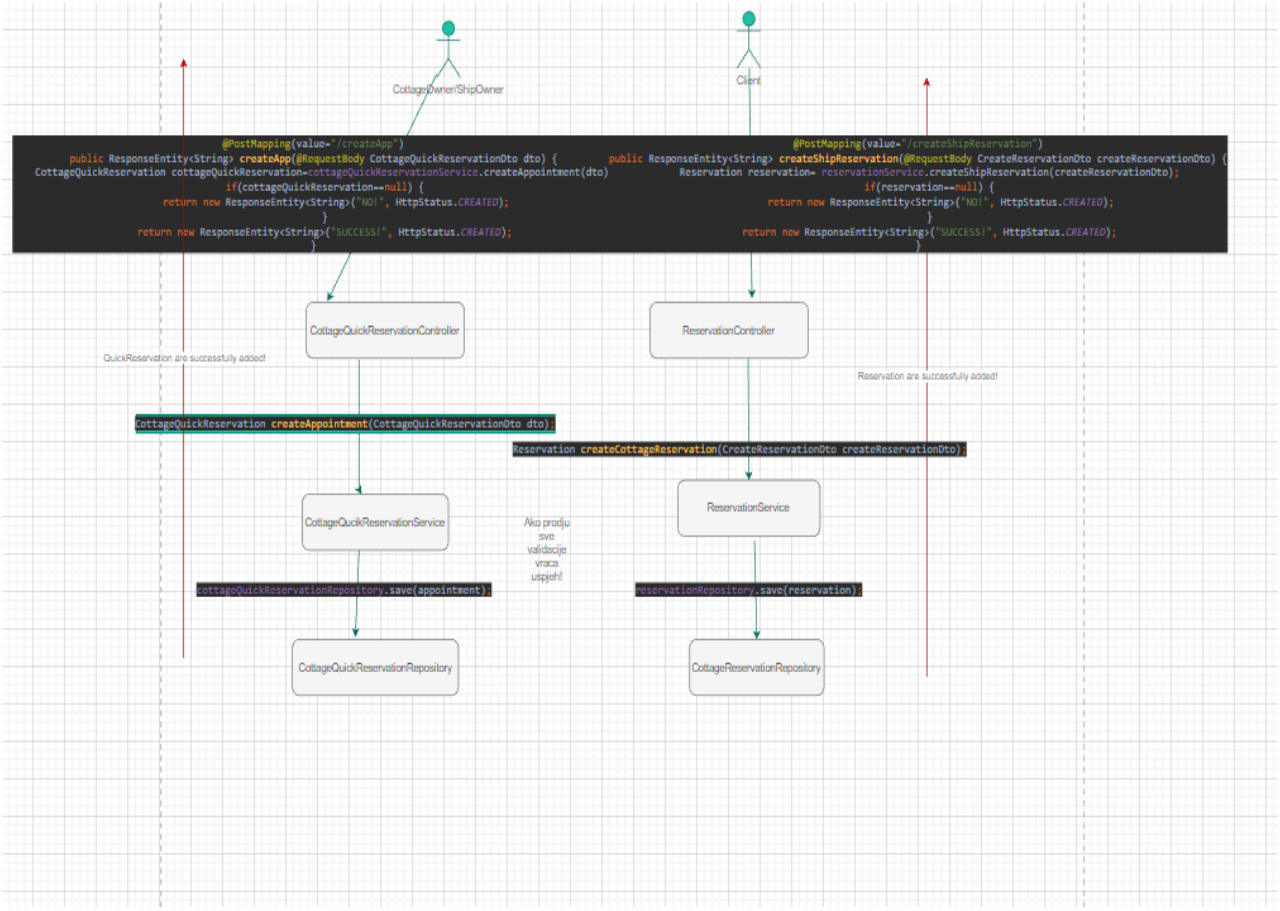
```
@Override
@Transactional
public Reservation createCottageReservation(CreateReservationDto createReservationDto) {
```

2.Vlasnik vikendice/broda ne može da napravi akciju u isto vrijeme kad i drugi klijent vrši rezervaciju datog entiteta

Opis problema:

Vlasnik vikendice/broda može da kreira brze rezervacije koje kasnije klijent jednim klikom može da rezerviše. Prilikom kreiranja akcije vrši se validacija do li je dati termin već zauzet. Ukoliko nije, akcija će se uspješno izvršiti. Ono što se može desiti je da i klijent i vlasnik i isto vrijeme rezervišu dati termin, klijent dodavanjem rezervacije, a vlasnik dodavanjem akcije sa preklapajućim terminom. Ovim se narušava uslov da ne može u isto vrijeme da postoji i akcija i rezervacija sa istim terminom. Potrebno je riješiti datu konfliktnu situaciju.

Tok zahtjeva:



Rješenje problema:

Za rješenje problema sam koristila pesimistično zaključavanje resursa. Rješenje je implementarino u ShipQuickReservationServiceImpl, CottageReservationRepository, ShipReservationRepository i CottageQuickReservationImpl. Za metode koje služe za kreiranje objekata vikendice , broda kao i na sve povezane metode van ovog service dodata je anotacija *Transactional*, tj sve su transakcione metode. U repozitorijumu CottageRepository i

ShipRepository zaključavamo metodu findCottageById/findShipById k,te će vikendica/brod biti zaključani sve dok se ne izvrši validacija odabranog termina i dati termin na sačuva u bazu podataka. Kao tip zaključavanja koristi se *PESSIMISTIC_WRITE* kojom se ne dozvoljava ni čitanje datog reda. Ukoliko pristigne novi zahtjev za dodavanje rezervacije,korisnik će dobiti poruku da je termin već zauzet. Ovim je riješena konfliktna situacija i onemogućeno rezervisanje istog termina u isto vrijeme.

Zaključavanje je uređeno u repozitorijima CottageRepository and ShipRepository i prikazano je na fotografiji.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Ship c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Ship findShipById(Long id);
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Cottage c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Cottage findCottageById(Long id);
```

Dodata anotacija u klase ShipQuickReservationImpl and CottageQuickReservationImpl.

```
@Override
@Transactional
public CottageQuickReservation createAppointment(CottageQuickReservationDto dto) {
```

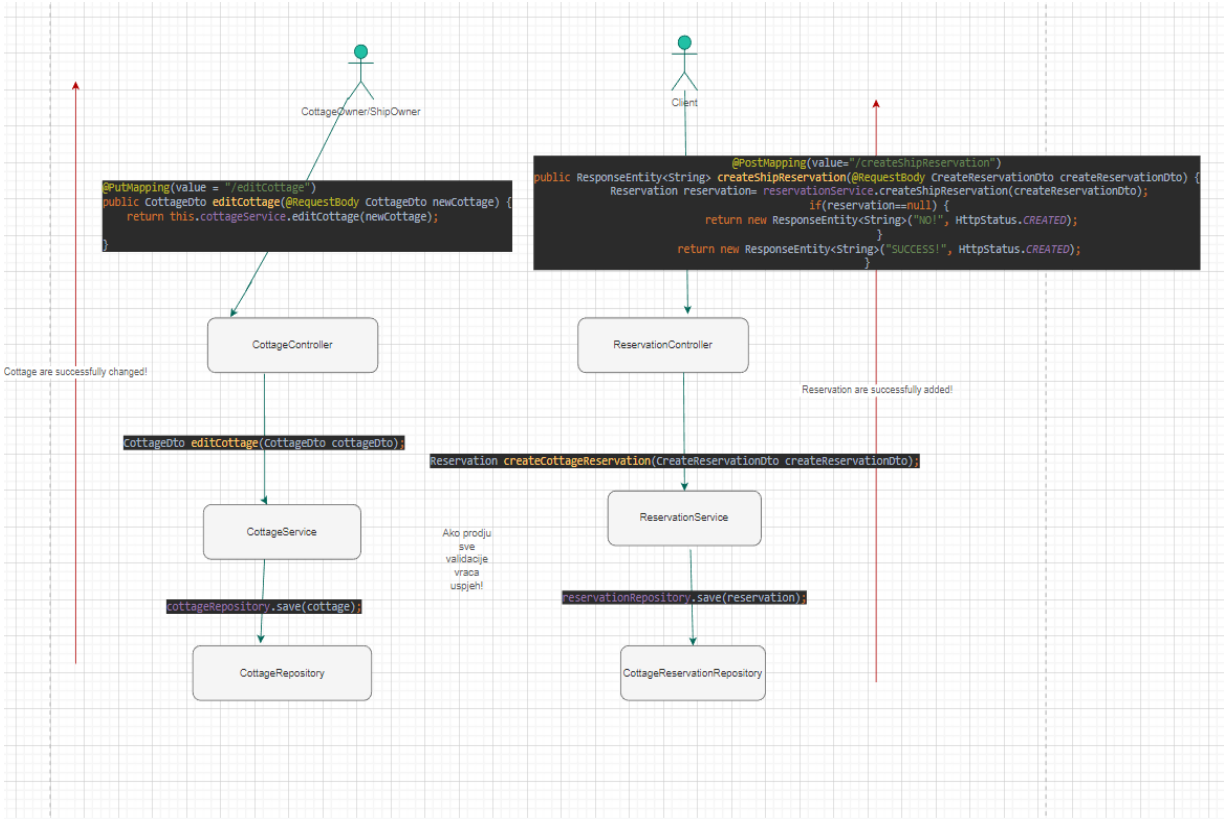
```
@Override
@Transactional
public ShipQuickReservation createAppointment(ShipQuickReservationDto dto)
```

3.Vlasnik vikendice/broda ne može da ažurira profil vikendice/broda u isto vrijeme kad drugi klijent dodaje rezervaciju datog entiteta

Opis problema:

Pretpostavimo da vlasnik vikendice/broda želi da ažurira podatke vikendice/broda dok klijent želi da u isto vrijeme doda rezervaciju za datu vikendicu/brod. Da ne bi došlo do situacije da se podaci o vikedici/brodu kao što su usluge i cijena promijene odmah po rezervisanju entiteta, potrebno je da riješimo dati konflikt.

Tok zahtjeva:



Rješenje problema:

Za rješenje problema sam koristila pesimistično zaključavanje resursa. Rješenje je implementarino u CottageServiceImpl,ShipServiceImpl, CottageRepository, i Shipepository. Za metode koje služe za editovanje vikendice , broda kao i na sve povezane metode van ovog service dodata je anotacija *Transactional*, tj sve su transakcione metode. U repozitoriju CottageRepository i ShipRepositoty zaključavamo metodu findCottageById/findShipById ,te će vikendica/brod biti zaključani sve dok se ne izvrši validacija odabranog termina i dati termin na sačuva u bazu podataka. Kao tip zaključavanja koristi se *PESSIMISTIC_WRITE* kojom se ne dozvoljava ni čitanje datog reda.. Ovim je riješena konfliktna situacija i onemogućeno je editovanje datog entiteta.

Zaključavanje je odrađeno u repozitorijima CottageRepository i ShipRepository.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Cottage c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Cottage findCottageById(Long id);
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Cottage c WHERE c.id = :id")
@QueryHints({@QueryHint(name ="javax.persistance.lock.timeout",value="0")})
Cottage findCottageById(Long id);
```

Dodata anotacija u CottageServiceImpl and ShipServiceImpl.

```
@Override
@Transactional
public CottageDto editCottage(CottageDto cottageDto) {
```

```
@Override
@Transactional
public ShipDto editShip(ShipDto shipDto) {
```