Konfliktne situacije – student 2 – Matija Uskoković RA76/2018

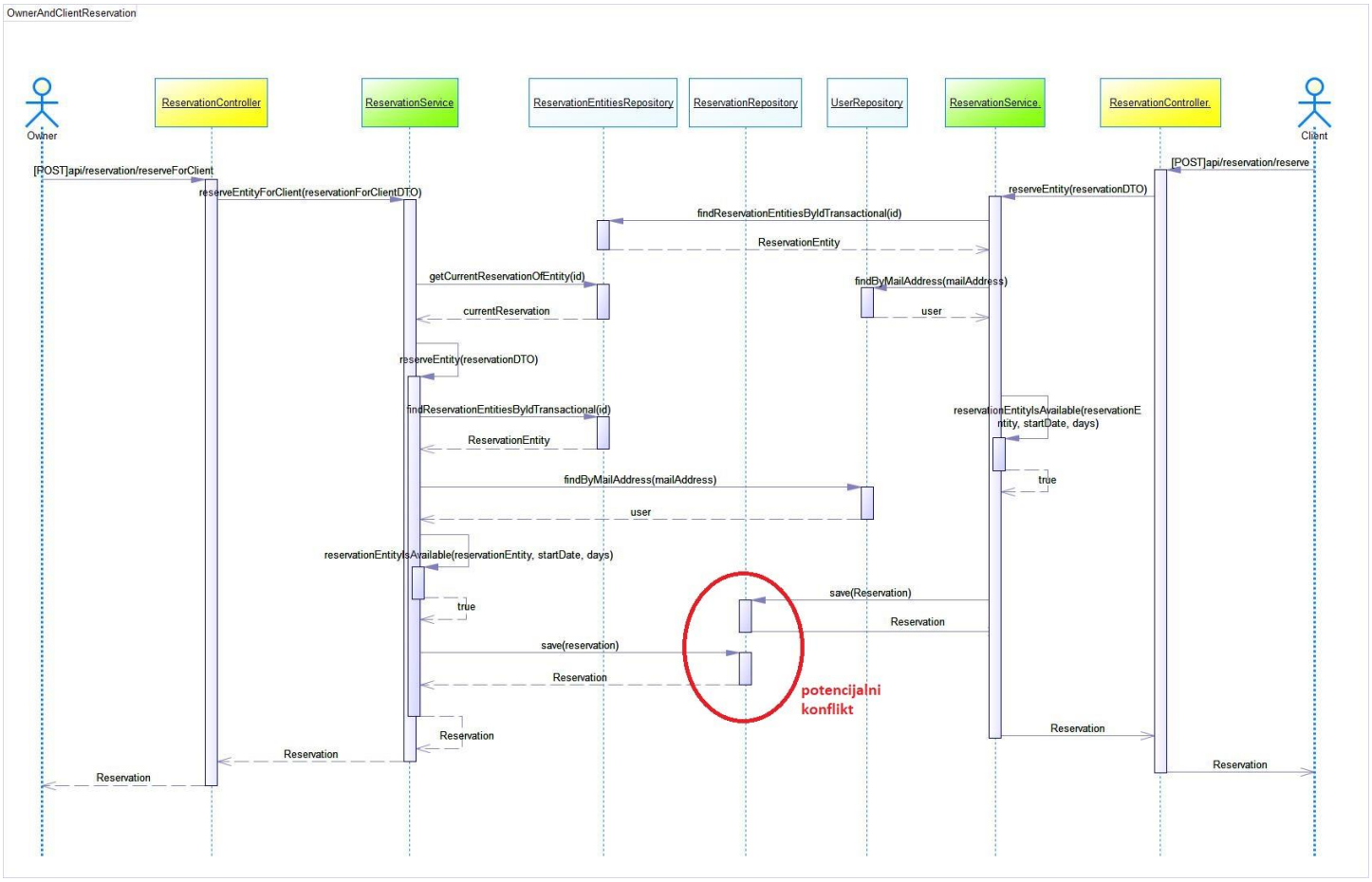
1. Vlasnik vikendice/broda ne može da napravi rezervaciju u isto vreme kad i drugi klijent

Opis konfliktne situacije:

Vlasnici vikendica ili brodova mogu da naprave novu rezervaciju za klijenta, kojem trenutno traje rezervacija, i time da rezervišu taj isti entitet za neko buduće vreme. Problem može nastati ukoliko neki drugi korisnik želi da rezerviše taj isti entitet u neko isto ili preklapajuće buduće vreme. Iako pre poziva metode za rezervisanje postoji provera da li je entitet za dato vreme već rezervisan do konflikta može doći ukoliko se za oba zahteva u isto vreme odradi čitanje iz baze i provera da li je entitet slobodan što bi na kraju propustilo i jedan i drugi zahtev da rezervišu entitet.

Slika:

Na slici je prikazana rešena situacija koja je gore opisana sa označenim mestom gde se pre rešenja nalazio potencijalni konflikt.



Rešenje:

Za rešenje ovog problema koristi se pesimističko zaključavanje entiteta za rezervaciju što dovodi do toga da će samo onaj koji je prvi učitao iz baze entitet moći da vrši proveru da li je moguće rezervisati entitet i na kraju izvršiti samu rezervaciju.

U kodu je implementirano tako što je u ReservationService iznad metode reserveEntityForClient(ReservationForClientDTO reservationForClientDTO) postavljena anotacija @Transactional(readonly = false) koja označava da će se cela metoda izvršavati kao jedna transakcija, a ona

poziva metodu reserveEntity(ReservationDTO reservationDTO) koja iz repozitorijuma za entitete koji se rezervišu (ReservationEntitiesRepository) poziva metodu findReservationEntitiesByIdTransactional(Long id) iznad koje se nalazi anotacija @Lock(LockModeType.*PESSIMISTIC_WRITE*) koja će zabraniti i čitanje i izmenu sve dok se ne završi transakcija koja je prva pristupila toj metodi, tako da će jedino ta prva transakcija moći da vrši proveru da li je željeni entitet slobodan za rezervaciju i nakon toga ga rezervisati.

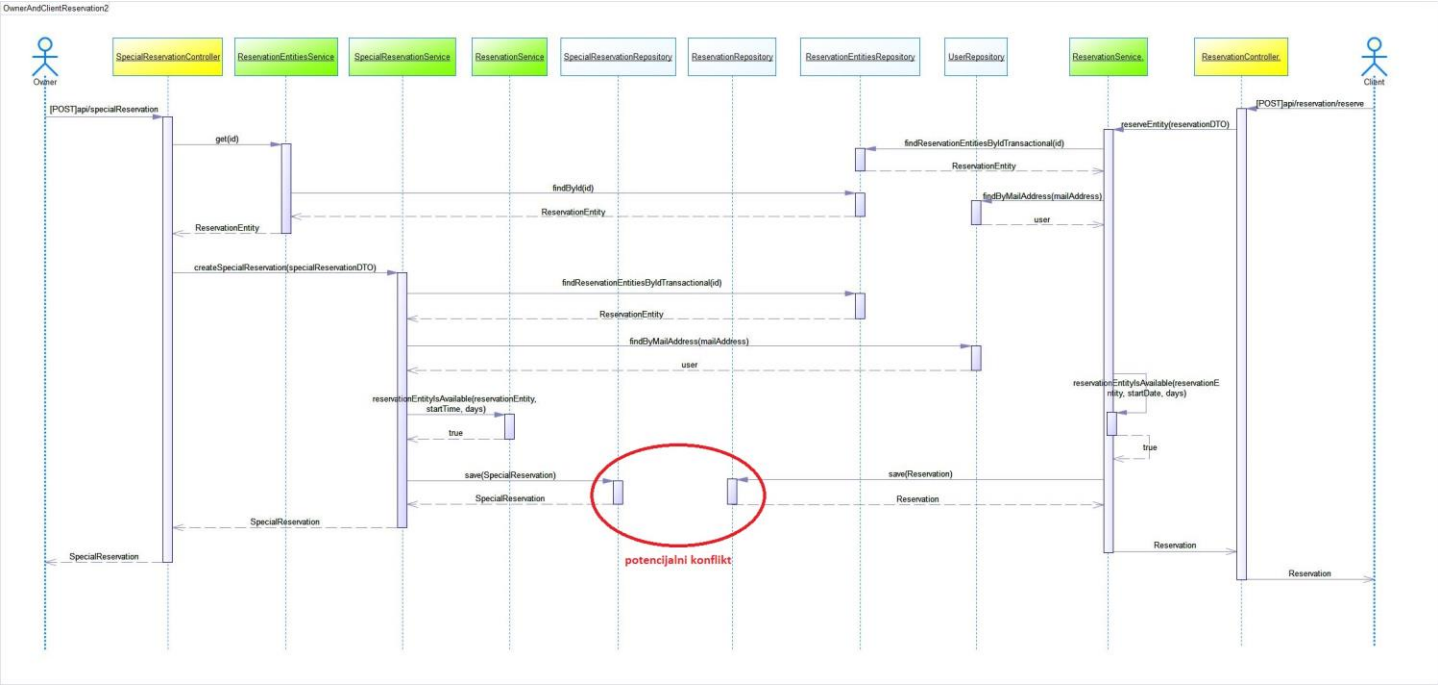
2. Vlasnik vikendice/broda ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta

Opis konfliktne situacije:

Vlasnici vikendica ili brodova mogu da prave akcije za svoje entitete za neko buduće vreme u kojem je taj entitet dostupan za rezervacije. Poblem može nastati ukoliko neki drugi običan korisnik želi da rezerviše taj entitet u isto ili preklapajuće vreme. Iako pre poziva metode za rezervisanje i pravljenje akcije postoji provera da li je entitet za dato vreme već rezervisan do konflikta može doći ukoliko se za oba zahteva u isto vreme odradi čitanje iz baze i provera da li je entitet slobodan što bi na kraju propustilo i jedan i drugi zahtev pa bi se kreirala i rezervacija i akcija za isto ili preklapajuće vreme.

Slika:

Na slici je prikazana rešena situacija koja je gore opisana sa označenim mestom gde se pre rešenja nalazio potencijalni konflikt.



Rešenje:

Za rešenje ovog problema koristi se pesimističko zaključavanje entiteta za rezervaciju što dovodi do toga da će samo onaj koji je prvi učitao iz baze entitet moći da vrši proveru da li je entitet slobodan za traženo vreme i na kraju izvršiti samu rezervaciju ili kreiranje akcije.

U kodu je implementirano tako što je u SpecialReservationService iznad metode createSpecialReservation(SpecialReservationDTO specialReservationDTO) postavljena anotacija @Transactional(readonly = false) koja označava da će se cela metoda izvršavati kao jedna transakcija i u njoj se

poziva metoda findReservationEntitiesByIdTransactional(Long id) , iz repozitorijuma ReservationEntitiesRepository, iznad koje se nalazi anotacija @Lock(LockModeType.*PESSIMISTIC_WRITE*) koja će zabraniti i čitanje i izmenu sve dok se ne završi transakcija koja je prva pristupila toj metodi, tako da će jedino ta prva transakcija moći da vrši proveru da li je željeni entitet slobodan za rezervaciju ili akciju i nakon toga ga rezervisati ili kreirati akciju.

Isto je urađeno i na strani rezervacije entiteta od strane klijenta koji koristi ReservationService gde je iznad metode reserveEntity(ReservationDTO reservationDTO) postavljena anotacija @Transactional(readonly = false) u kojoj se poziva metoda findReservationEntitiesByIdTransactional(Long id) , iz repozitorijuma ReservationEntitiesRepository.

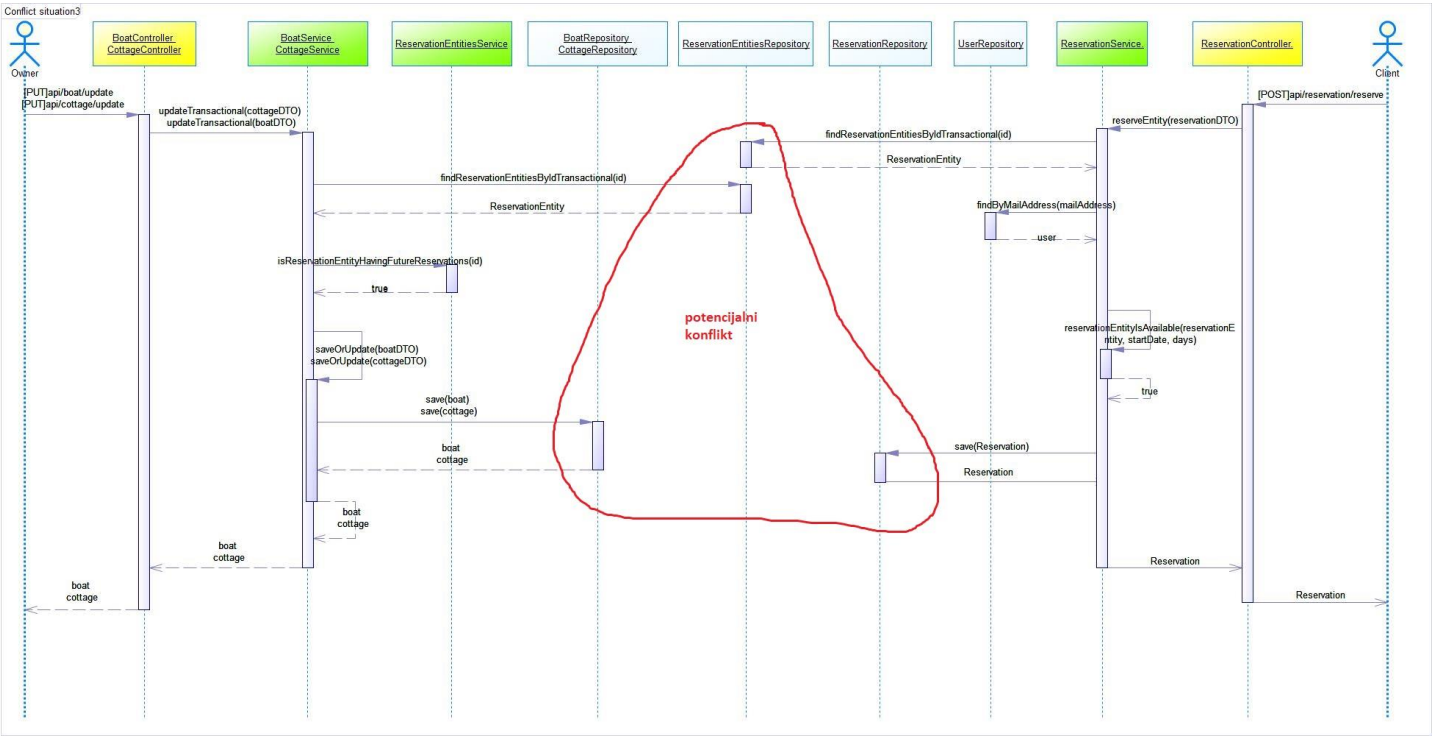
3. Vlasnik vikendice/broda ne može da izmeni svoj entitet u isto vreme kad i drugi klijent vrši rezervaciju dok još nije izmenjen entitet

Opis konfliktne situacije:

Vlasnici vikendica imaju mogućnost da menjaju svoje entitete ukoliko nema budućih rezervacija. Problem može nastati ukoliko neki klijent želi da rezerviše entitet po prethodno prikazanim informacijama u isto vreme kada vlasnik želi da izmeni neke od informacija nakon čega dolazi do konflikta gde je vlasnik izmenio informacije entiteta koji ima buduće rezervacije ili je korisnik rezervisao entitet po nevalidnim informacijama.

Slika:

Na slici je prikazana rešena situacija koja je gore opisana sa označenim mestom gde se pre rešenja nalazio potencijalni konflikt. Označeno mesto prikazuje da je ranije obična metoda za dobavljanje entiteta bez ikakve provere vraćala entitet koji je vlasnik kasnije mogao da izmeni ili klijent da rezerviše i ukoliko bi to uradili u isto vreme došlo bi do konflikta.



Rešenje:

Za rešenje ovog problema koristi se pesimističko zaključavanje entiteta za izmenu ili rezervaciju što dovodi do toga da će samo onaj koji je prvi učitao iz baze entitet moći da radi sa njim i neće doći do konflikta.

U kodu je implementirano tako što je u BoatService ili CottageService iznad metode updateTransactional(BoatDTO boatDTO)/updateTransactional(CottageDTO cottageDTO) postavljena anotacija @Transactional(readonly = false) koja označava da će se cela metoda izvršavati kao jedna transakcija i u njoj se poziva metoda findReservationEntitiesByIdTransactional(Long id) (iz repozitorijuma ReservationEntitiesRepository), koja se takođe poziva i iz ReservationService, iznad koje se nalazi anotacija @Lock(LockModeType.*PESSIMISTIC_WRITE*) koja će zabraniti i čitanje i izmenu sve dok se ne završi transakcija koja je prva pristupila toj metodi, tako da će jedino ta prva transakcija moći da radi nad dobijenim entitetom.