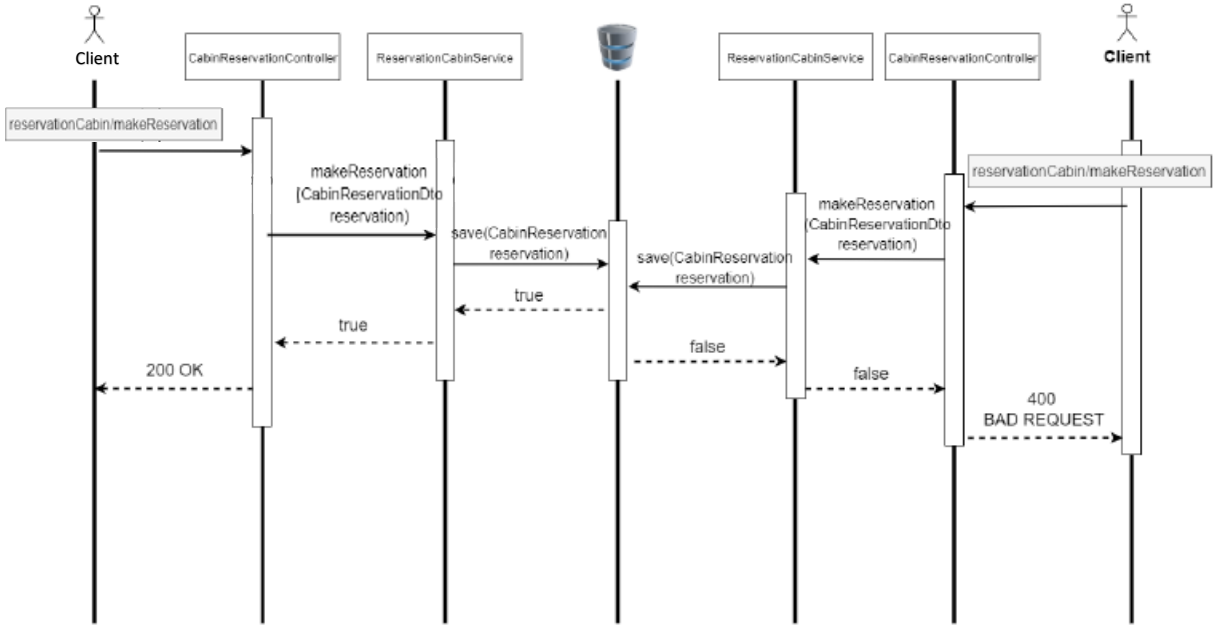
KONKURENTNI PRISTUP BAZI - Student 1

1. Kreiranje rezervacije od strane više klijenata u isto ili preklapajuće vreme

Suština ovog sistema je mogućnost rezervacije entiteta koju klijenti poseduju. Pri procesu rezervisanja određenog entiteta, ukoliko dva klijenta pokušaju da naprave rezervaciju za isti ili preklapajući period, dolazi do konfliktne situacije. Kako bismo izbegli kreiranje dve rezervacije za isti entitet u istom ili preklapajućem terminu, potrebno je rešiti ovu konfliktnu situaciju. Na slici 1 prikazana je situacija kada dva klijenta u isto vreme pokušavaju da rezervišu vikendicu u istom terminu, pri čemu je levi pokrenuo transakciju nekoliko sekundi ranije. (*Napomena: na dijagramu su izostavljene metode logike servisa i validacije kako bi se pokazala suština transakcija*)



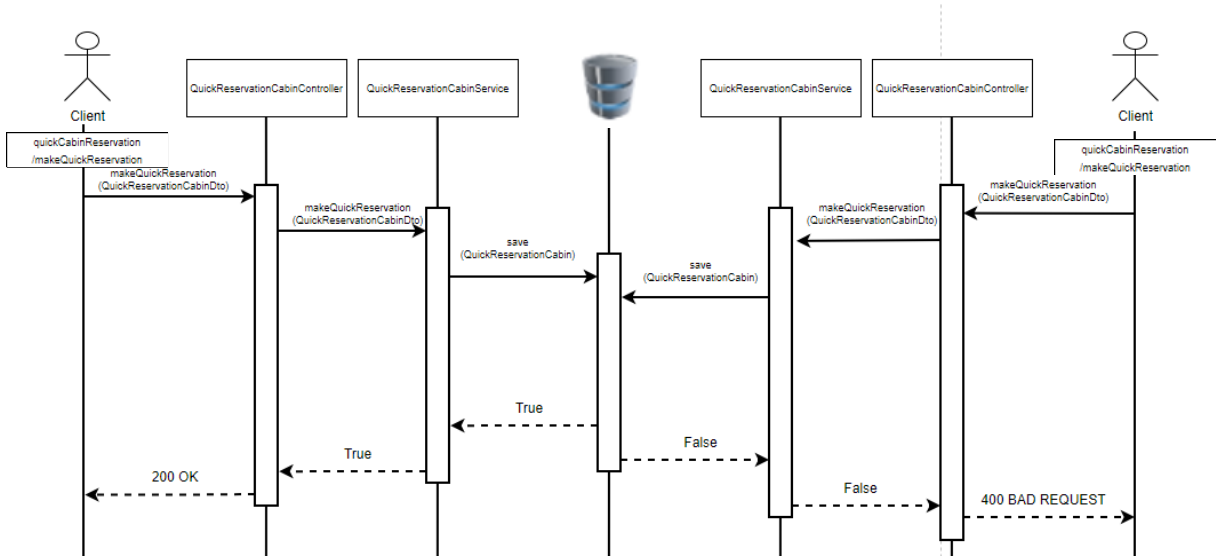
Slika 1 Kreiranje rezervacije

Problem je rešen upotrebom optimističkog zaključavanja. U klasu Cabin dodato je polje version sa anotacijom @Version koja nam omogućava da izvršimo verzionisanje torke u bazi. Za kreiranje rezervacije korišćena je metoda servisa CabinReservationService makeReservation. Metoda je označena kao transakciona anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE (rešena su četiri glavna problema konkurentnog pristupa bazi), atribut readonly postavljen na false jer vršimo izmenu u bazi, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje. U slučaju da klijenti u isto vreme pokrenu

transakciju vezanu za rezervaciju istog entiteta, ali te rezervacije nisu u istom ili preklapajućem terminu, prvo će se izvršiti rezervacija koja je prva stigla, a nakon što ona bude sačuvana sačuvaće se druga rezervacija.

2. Kreiranje brze rezervacije od strane klijenata u isto vreme

Klijenti imaju mogućnost brze rezervacije, odnosno rezervacije entiteta na akciji. Ukoliko klijenti pokušaju da izvrše brzu rezervaciju istog entiteta za isti termin, dolazi do konfliktne situacije. Kako bismo izbegli izvršavanje brze akcije za isti entitet u istom terminu, potrebno je rešiti ovu konfliktnu situaciju. Na slici 2 prikazana je situacija kada klijenti izvršavaju brzu akciju, pri čemu je levi klijent pokrenuo transakciju nekoliko sekundi ranije. (Napomena: na dijagramu su izostavljene metode logike servisa i validacije kako bi se pokazala suština transakcija)

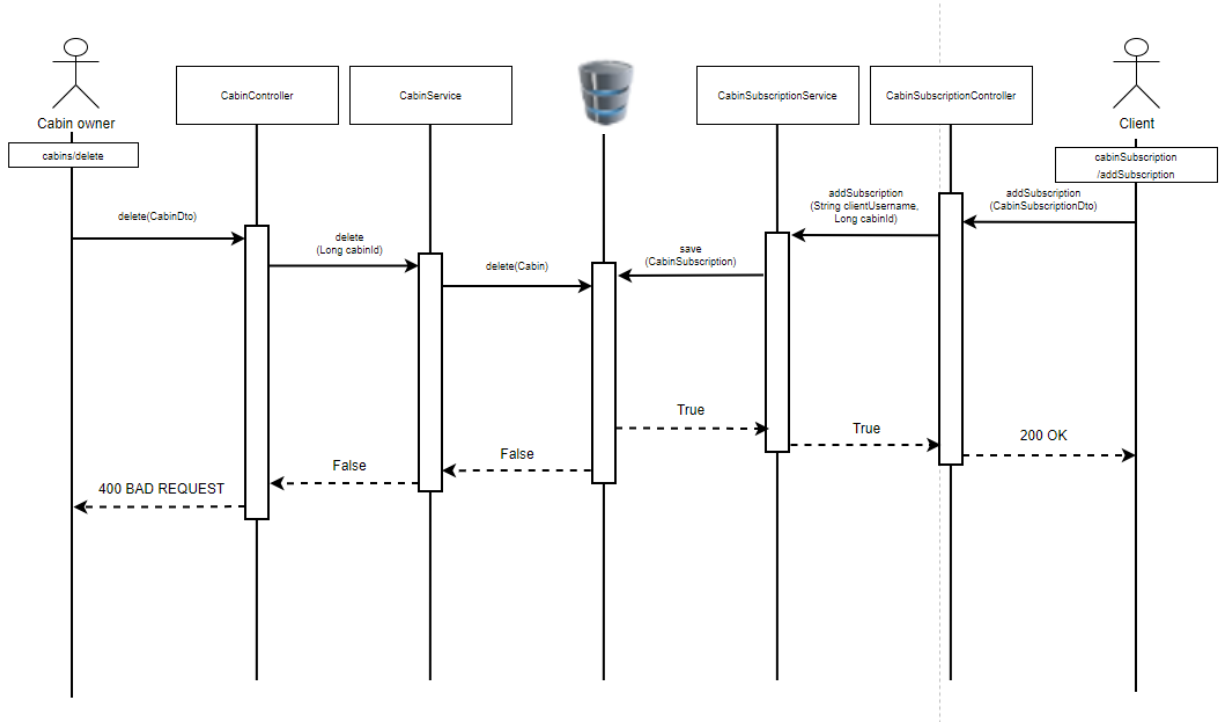


Slika 2 Brza rezervacija

Problem je rešen upotrebom optimističkog zaključavanja. U klasu Cabin dodato je polje version sa anotacijom @Version koja nam omogućava da izvršimo verzionisanje torke u bazi. Za izvršavanje brze rezervacije korišćena je metoda makeQuickReservation servisa QuickReservationCabinService. Metoda je označena kao transakciona anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE (rešena su četiri glavna problema konkurentnog pristupa bazi), atribut readonly postavljen na false jer vršimo izmenu u bazi, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje. U slučaju da klijenti u isto vreme pokrenu transakciju vezanu za brzu rezervaciju istog entiteta, ali te rezervacije nisu u istom terminu, prvo će se izvršiti rezervacija koja je prva stigla, a nakon što ona bude sačuvana sačuvaće se druga rezervacija.

2. Pretplata na akcije entiteta I brisanje entiteta

Vlasnici vikendica, brodova ili instruktori pecanja imaju mogućnost brisanja entiteta. Ukoliko vlasnik vikendice pokuša da obriše entitet u trenutku kada neki klijent pokuša da se pretplati, dolazi do konfliktne situacije. Kako bismo ovo izbegli, potrebno je rešiti ovaj konflikt. Na slici 3 prikazana je situacija kada vlasnik vikendice pokušava da obriše vikendicu, a klijent pokušava da se pretplati. (*Napomena: na dijagramu su izostavljene metode logike servisa i validacije kako bi se pokazala suština transakcija*)



Slika 3 Pretplata i brisanje entiteta

Problem je rešen upotrebom optimističkog zaključavanja. U klasu Cabin dodato je polje version sa anotacijom @Version koja nam omogućava da izvršimo verzionisanje torke u bazi. Za brisanje vikendice od strane vlasnika korišćena je metoda delete servisa CabinService, a za pretplatu od strane klijenta metoda addSubscription servisa CabinSubscriptionService. Obe metode označene su kao transakcione anotacijom @Transactional. Nivo izolacije podešen je na SERIALIZABLE (rešena su četiri glavna problema konkurentnog pristupa bazi), atribut readonly postavljen na false jer vršimo izmenu u bazi, a atribut propagation na REQUIERS_NEW. Ovim smo obezbedili da metoda uvek pokreće novu transakciju, a ako postoji tekuća transakcija ona se suspenduje.

Napomena: Navedeni problemi rešeni su na isti način u slučaju brodova i avantura, ali je zbog jednostavnosti detaljno opisana vikednica. Metode, servisi i endpointi vezani za ove entitete nazvani su isto, samo imaju drugačije prefikse (Boat/ Adventure).