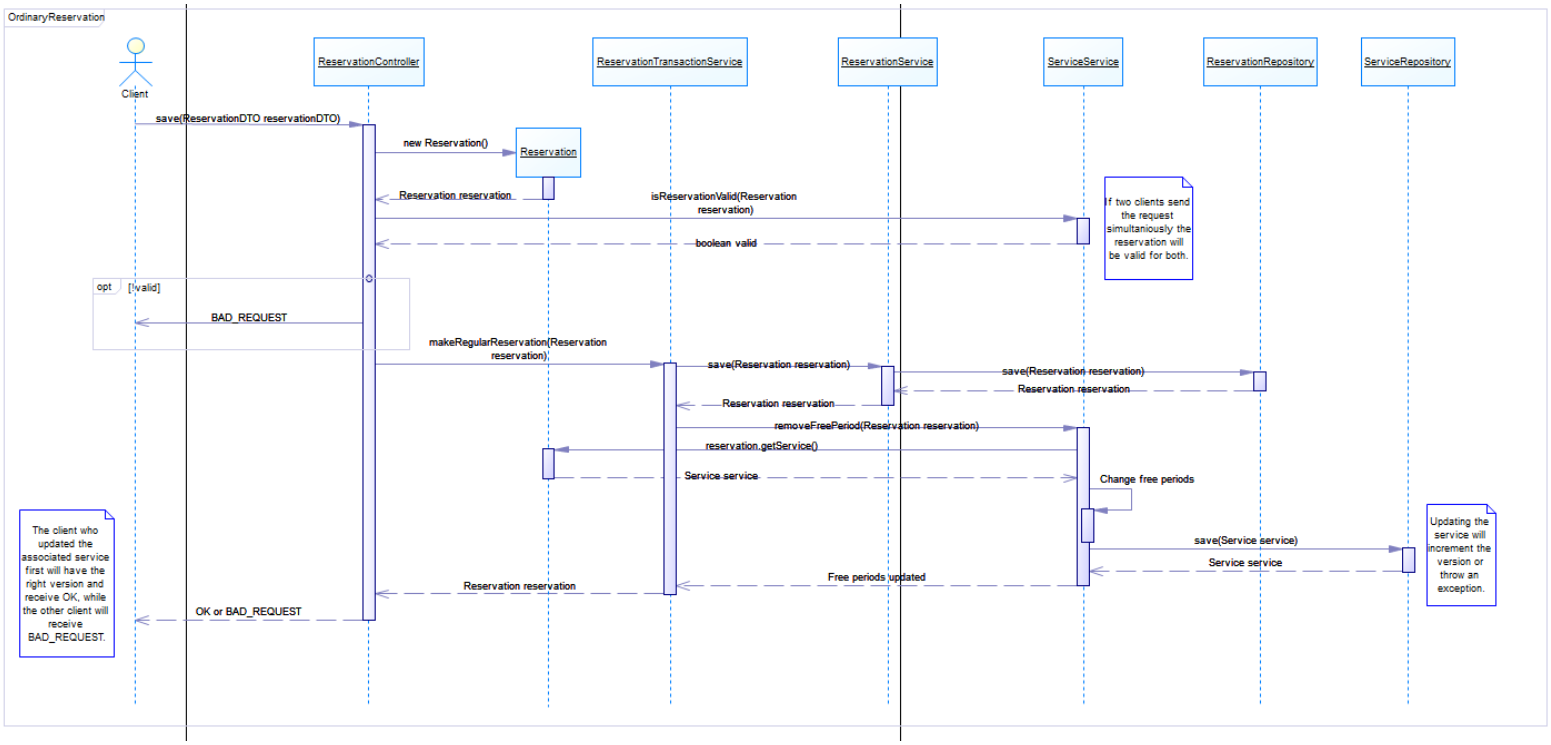
Konkurentnost – Student 1

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

Može doći do konfliktne situacije u slučaju da više korisnika istovremeno pokuša da rezerviše istu uslugu. U ovakvom slučaju, zahtevi korisnika bi bi bili uspešno validirani i na klijentskoj i na serverskoj strani, te bi došlo do dvostruke rezervacije istog entiteta. U idealnom slučaju, korisnik čiji zahtev je ranije obrađen bi uspešno obavio rezervaciju i bio obavešten o uspešnosti transakcije, dok bi drugom korisniku zahtev bio odbijen, o čemu bi on bio obavešten.

Rešenje:

Problem se rešava uvođenjem mehanizma optimističkog zaključavanja. U klasi *Service* (koju nasleđuju svi entiteti) dodato je polje *version* koje označava trenutnu verziju entiteta u bazi, pošto je rezervacija povezana sa entitetom koji se rezerviše. Prilikom kreiranja rezervacije, poziva se ažuriranje povezanog *Service* entiteta u bazi podataka. Ovom prilikom se proverava da li je trenutna verzija odgovarajuća verziji koja se nalazi u bazi i ukoliko jeste, transakcija se izvršava i verzija se inkrementira, a u suprotnom dolazi do *OptimisticLockingFailureException*-a, koji se obrađuje u kontroleru. U svakom slučaju klijent na odgovarajući način biva obavešten o uspešnosti transakcije. Dodatno, u klasi *ReservationTransactionService*, metoda *makeRegularReservation*, koja je zadužena za kreiranje rezervacije i podešavanje slobodnih perioda, je anotirana sa *@Transactional*, kako bi se izvršio rollback prilikom neuspešne transakcije.

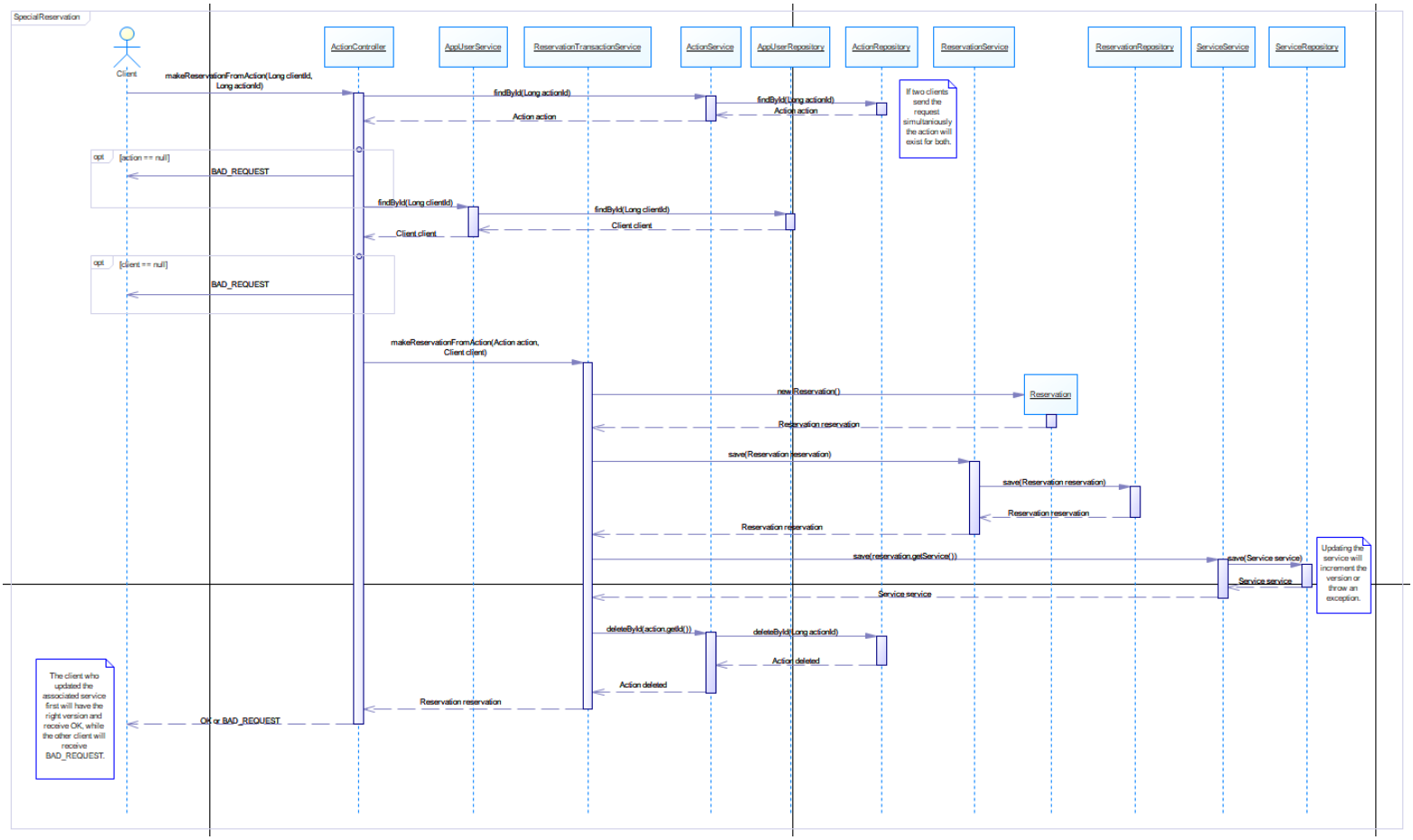


Konflikt 2: Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vreme

Kako su ponude na akciji dostupne svim prijavljenim korisnicima, lako može doći do konfliktne situacije u kojoj dva klijenta približno istovremeno pokušaju da rezervišu isti entitet. U navedenom scenariju, moglo bi se desiti da odgovarajuci *Action* entitet postoji u bazi podataka u vreme prihvatanja zahteva oba korisnika, što bi učinilo njihove zahteve validnim i došlo bi do dvostruke rezervacije. U idealnom slučaju, korisnik čiji zahtev je ranije obrađen bi uspešno obavio brzu rezervaciju i bio obavešten o uspešnosti transakcije, dok bi drugom korisniku zahtev bio odbijen, o čemu bi on bio obavešten.

Rešenje:

Opisani problem se takođe rešava uvođenjem mehanizma optimističkog zaključavanja. U klasi *Service* postoji polje *version* koji označava trenutnu verziju entiteta u bazi, pošto je rezervacija povezana sa entitetom koji se rezerviše. Prilikom kreiranja rezervacije, poziva se ažuriranje povezanog *Service* entiteta u bazi podataka. Ovom prilikom se proverava da li je trenutna verzija odgovarajuća verziji koja se nalazi u bazi i ukoliko jeste, transakcija se izvršava i verzija se inkrementira, a u suprotnom dolazi do *OptimisticLockingFailureException*-a, koji se obrađuje u kontroleru. U svakom slučaju klijent na odgovarajući način biva obavešten o uspešnosti transakcije. Dodatno, u klasi *ReservationTransactionService*, metoda *makeReservationFromAction*, koja je zadužena za brisanje akcije i kreiranje njoj odgovarajuče rezervacije, je anotirana sa *@Transactional*, kako bi se izvršio rollback prilikom neuspešne transakcije, što bi sačuvalo konzistentnost sitema.



Konflikt 3: Otkazivanje rezervacije ne sme da prouzrokuje neželjene nuspojave u sistemu

Kako su obična i brza reakcija implementirane kao transakcije radi konzistentnosti sistema, očekivano je da i otkazivanje rezervacija bude tretirano analogno. Otkazivanje postojeće rezervacije ažurira stanje rezervisanog entiteta, što može dovesti do brojnih nuspojava u podacima. Povraćanje slobodnog perioda rezervisanog entiteta može dovesti do konflikta u slučaju da vlasnik vikendice istovremeno ažurira slobodne periode ili ukoliko drugi klijent napravi rezervaciju. Ažuriranje rezervisanih entiteta treba da ih ostavi u konzistentnom stanju, te bi bilo poželjno da se izvrše samo transakcije koje operišu sa trenutnom verzijom entiteta. Svaki korisnik koji pokuša da izmeni zastarelu verziju entiteta biva o tome obavešten.

Rešenje:

Kao i u slučaju pravljenja rezervacija, uvodi se mehanizam optimističkog zaključavanja. U klasi *Service,* polje *version* označava trenutnu verziju entiteta u bazi pošto se ažurira prilikom izmene slobodnih perioda. Prilikom brisanja, poziva se ažuriranje povezanog *Service* entiteta u bazi podataka. Ovom prilikom se proverava da li je trenutna verzija odgovarajuća verziji koja se nalazi u bazi i ukoliko jeste, transakcija se izvršava i verzija se inkrementira, a u suprotnom dolazi do *OptimisticLockingFailureException*-a, koji se obrađuje u kontroleru. U svakom slučaju, korisnik na odgovarajući način biva obavešten o uspešnosti transakcije. Dodatno, u klasi *ReservationTransactionService*, metoda *cancelReservation*, koja je zadužena za brisanje rezervacije i povraćanje slobodnog perioda, je anotirana sa *@Transactional*, kako bi se izvršio rollback prilikom neuspešne transakcije.