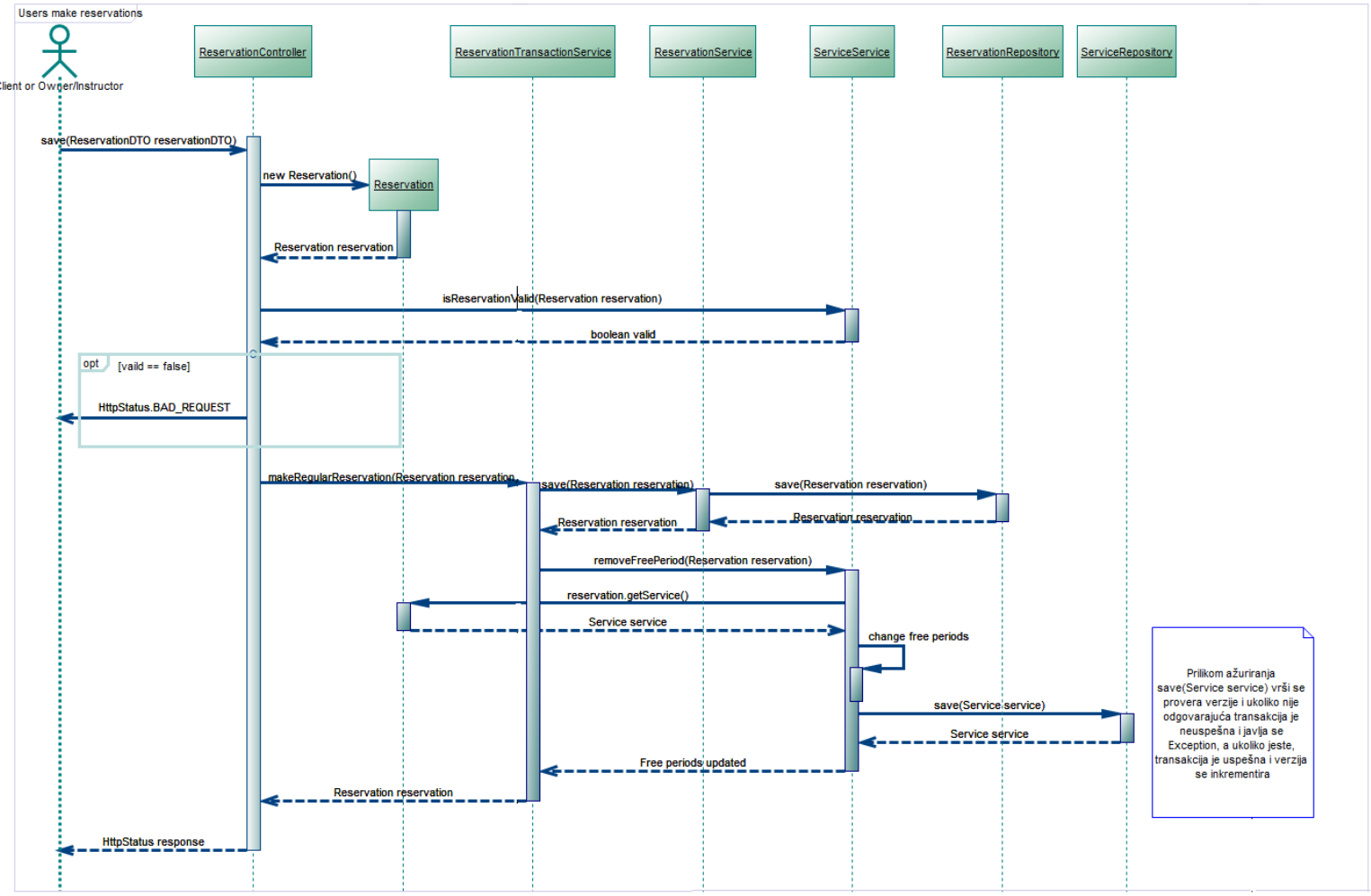
Konkurentan pristup resursima u bazi: Student 2

Konflikt 1: Vlasnik vikendice/broda ili instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent

Problem: Funkcionalnost rezervisanja vikendice, broda ili avantura je omogućena klijentima, ali i vlasnicima/instruktorima odgovarajućih entiteta. Moguće je da dođe do konfliktne kada klijent izvrši rezervaciju u istom trenutku kada i vlasnik/instruktor nad istim periodom kada je entitet slobodan. Ovakav konfliktni slučaj treba da se koriguje tako da korisnik koji pošalje zahtev za rezervaciju ranije uspešno rezerviše entitet, a da korisniku koji kasnije pošalje zahtev rezervacija ne bude odobrena.

Rešenje: Klasa *Service* je dopunjena poljem *version* anotiranim sa *@Version* čime se omogućava inktementiranje verzije entiteta prilikom svake izmene. Vrši se verzionisanje entiteta nad kojima je moguće napraviti rezervaciju (svih klasa koje nasleđuju *Service* klasu). Prilikom poziva metode *makeRegularReservation* poziva se metoda *removeFreePeriod* koja dalje poziva *save* metodu nad *Service* objektom, ažurirajući taj entitet. U ovom trenutku se vrši provera verzije i u slučaju kada se verzije ne poklapaju dobija se izuzetak *ObjectOptimisticLockingFailureException*. U slučaju da je verzija odgovarajuća, inkrementira se *version* polje i transakcija se uspešno izvršava.

Dijagram:



Konflikt 2: Vlasnik vikendice/broda ili instruktor ne može da napravi akciju u isto vreme kad klijent vrši rezervaciju postojećeg entiteta

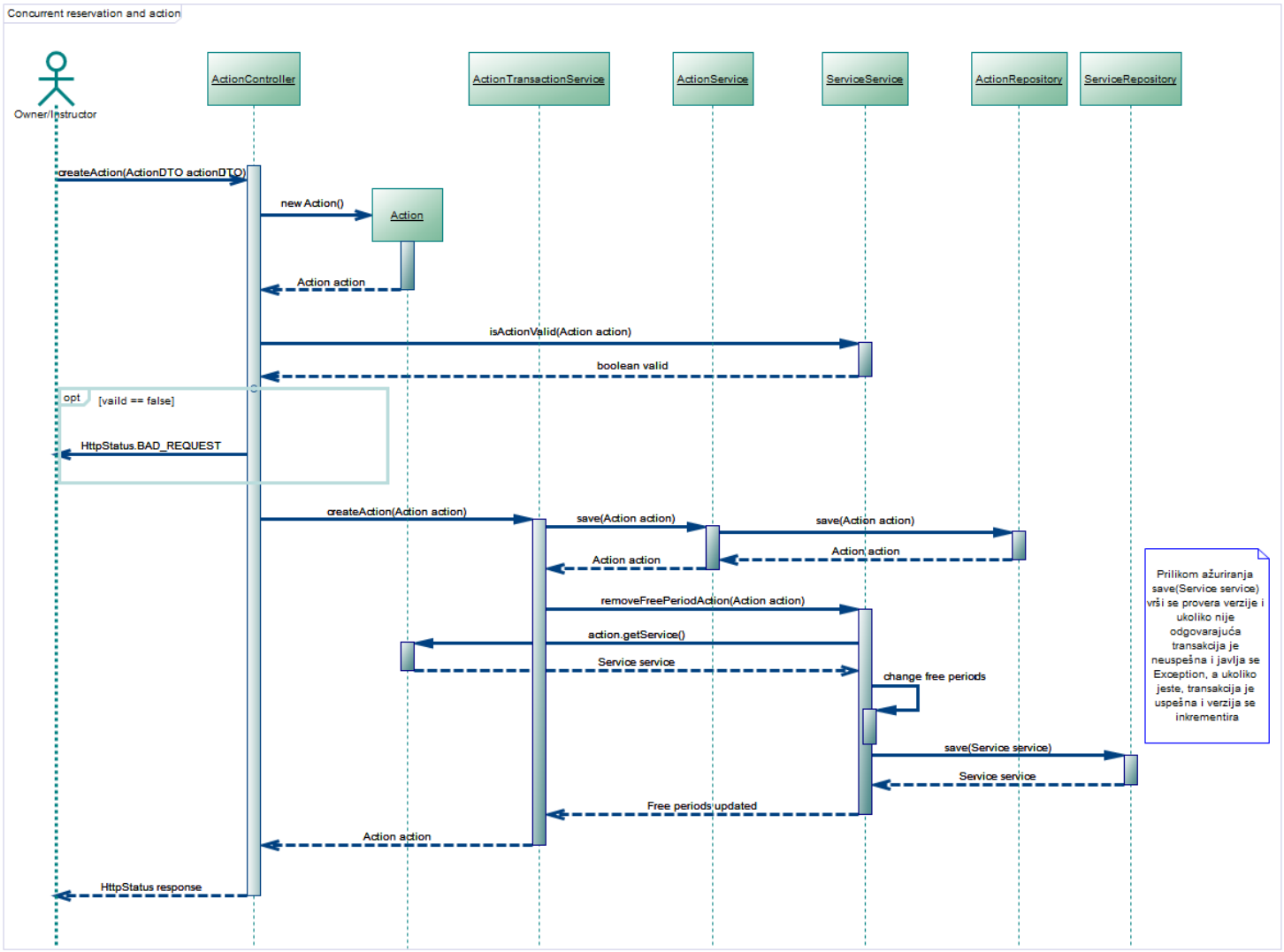
Problem:

Akcije su, poput rezervacija, modelovane kao entiteti koji se mogu kreirati samo u okviru već definisanog slobodnog perioda. Akcije mogu da kreiraju samo vlasnici odnosno instruktori za svoje odgovarajuće entitete. Do konfliktne situacije dolazi kada klijent vrši rezervaciju nad entitetom u određenom slobodnom periodu, a vlasnik/instruktor istovremeno kreira akciju nad istim entitetom u istom slobodnom periodu.

Rešenje:

Problem se rešava na gotovo identičan način kao u prethodnom slučaju, mehanizmom optimističkog zaključavanja. Verzionišu se entiteti klasa koje nasleđuju *Service*, prethodno je opisano funkcionisanje metode makeRegularReservation, a analogno njoj je kreirana i metoda createAction koja poziva metodu RemoveFreePeriodAction u kojoj se na kraju odvija ažuriranje entiteta pozivom save metode. U ovom trenutku se vrši provera verzije i u slučaju kada se verzije ne poklapaju dobija se izuzetak *ObjectOptimisticLockingFailureException*. U slučaju da je verzija odgovarajuća, inkrementira se *version* polje i transakcija se uspešno izvršava.

Dijagram:



Konflikt 3: Vlasnik vikendice/broda ili instruktor ne može da obriše slobodan period u isto vreme kada klijent vrši rezervaciju u tom periodu

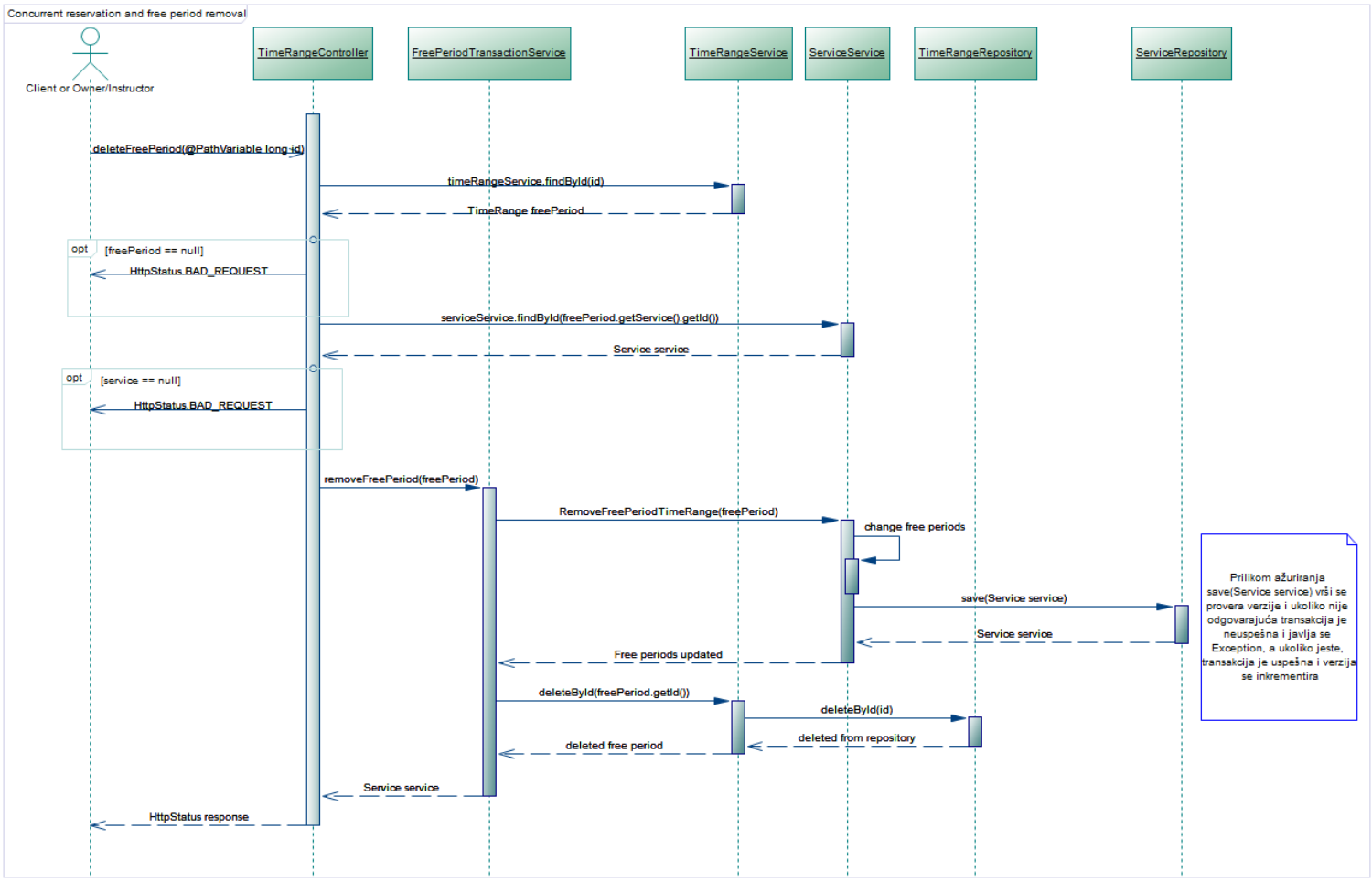
Problem:

Vlasnici/instruktori imaju mogućnost brisanja slobodnih perioda nad svojim entitetima vikendica/brodova/avantura. Ovo predstavlja problem u slučaju kada klijent želi da izvrši rezervaciju nad entitetom u slobodnom periodu koji se istovremeno briše od strane vlasnika/instruktora.

Rešenje:

Problem je rešen kao i u prethodnim slučajevima, verzionisanjem entiteta koji nasleđuju *Service* i primenom optimističkog zaključavanja. Pozivom metode removeFreePeriod lančano se pozivaju metode RemoveFreePriodTimeRange i save nad entitetom, prilikom čega se vrši provera vezija. U slučaju kada se verzije ne poklapaju dobija se izuzetak *ObjectOptimisticLockingFailureException*. U slučaju da je verzija odgovarajuća, inkrementira se *version* polje i transakcija se uspešno izvršava.

Dijagram:



Napomena: Metode makeRegularReservation, createAction i removeFreePeriod iz servisnih klasa ReservationTransactionService, ActionTransactionService i FreePeriodTransactionService su anotirane sa @Transactional, što omogućava da se izvrši rollback u slučaju neuspešne transakcije i time očuva konzistentnost.