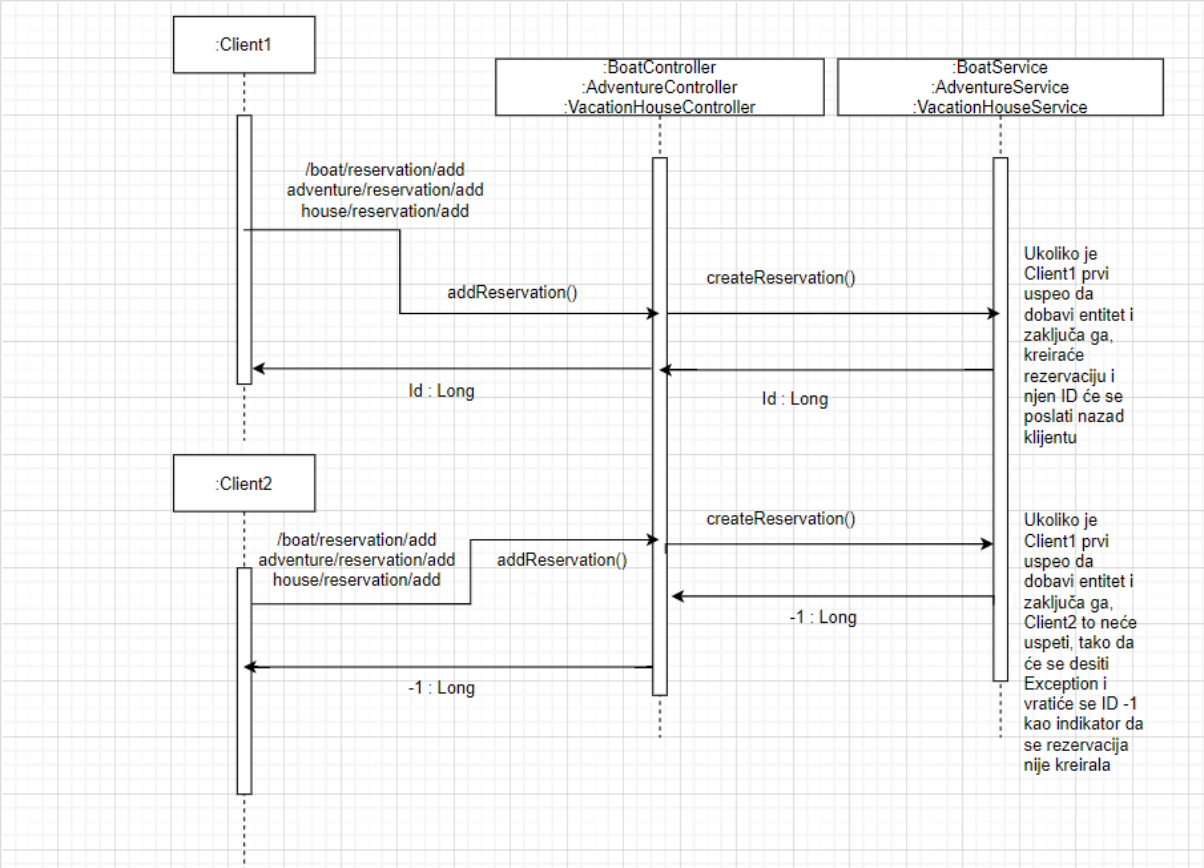
Student 1 - Miloš Milovanović SW 17/2019

1. Konflikt

a. Opis konfliktne situacije

Više istovremenih klijenata pravi rezervaciju istog entiteta u isto (ili preklapajuće) vreme

b. Tok zahteva



c. Rešenje

Za rešavanje ovog problema odabrano je pesimističko zaključavanje. Razlog za to je što se prilikom svake rezervacije kreira novi objekat *AdventureReservation/BoatReservation/VacationHouseReservation* koji se dodaje entitetu koji se rezerviše. Kako se ažurira entitet i pri tom se dodaje novi red u bazu podataka, čitav entitet je potrebno zaključati.

Prilikom kreiranja rezervacije, poslati DTO je potrebno pretvoriti u konkretnu rezervaciju. Funkcija koja to obavlja naziva se *createFromDTO* i ona je transakciona. U ovoj funkciji dobavlja se entitet iz baze za koji se vrši rezervacija. Funckija koja dobavlja entitet iz baze naziva se *getByIdConcurrent* i ona poziva funkciju repozitorijuma *findOneById* koja je zaključana ključem koji je tipa *PESSIMISTIC_WRITE*. Ukoliko je entitet zaključan ova funkcija će izazvati *PessimisticLockingFailureException* koji je potrebno uhvatiti I obavestiti klijenta da trenutno nije moguće obaviti rezervaciju.

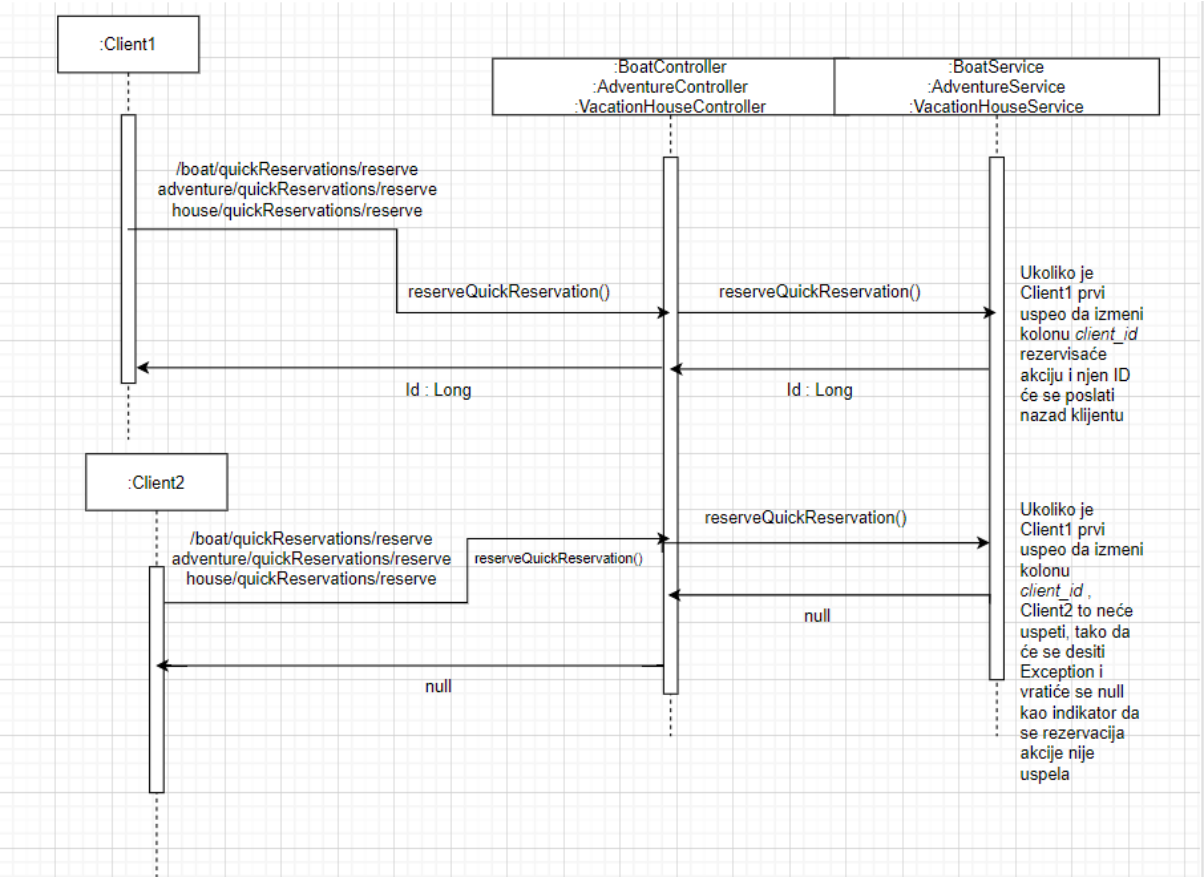
Na ovaj način rešen je i problem kada vlasnik vikendice/broad ili instructor pecanja rezerviše u isto vreme kad I klijent s obzirom na to da se pozivaju identične funkcije.

2. Konflikt

d. Opis konfliktne situacije

Više istovremenih klijenata prave rezervaciju istog entiteta na akciji u isto vreme

e. Tok zahteva



f. Rešenje

Za rešavanje ovog problema odabrano je optimističko zaključavanje. Razlog za to je što prilikom rezervacije akcije, akcija već kreirana i postoji u bazi. Ono što je potrebno odraditi jeste izmeniti kolonu *client_id* i iz tog razloga objektima *AdventureReservation/BoatReservation/VacationHouseReservation* dodat je poseban atribut *Version* koji predstavlja mehanizam za rešavanje konflikata u konkuretnom pristupu entitetima baze.

Prilikom obavljanja rezervacije akcije, potrebno je dobaviti rezervaciju na koju se akcija odnosi i ona predstavlja objekat klase *AdventureReservation/BoatReservation/VacationHouseReservation*. Nakon što se objektu postavi atribut *Client* na odgovarajući način, ove izmene potrebno je ponovo sačuvati u bazu podataka. Ažuriranje kolone u bazu obavlja se funkcijom *saveQuickReservationAsReservation* i ona je transakciona. Ukoliko se atribut rezervacije *Version* ne poklapa na odgovarajući način sa atributom *Version* rezervacije koju

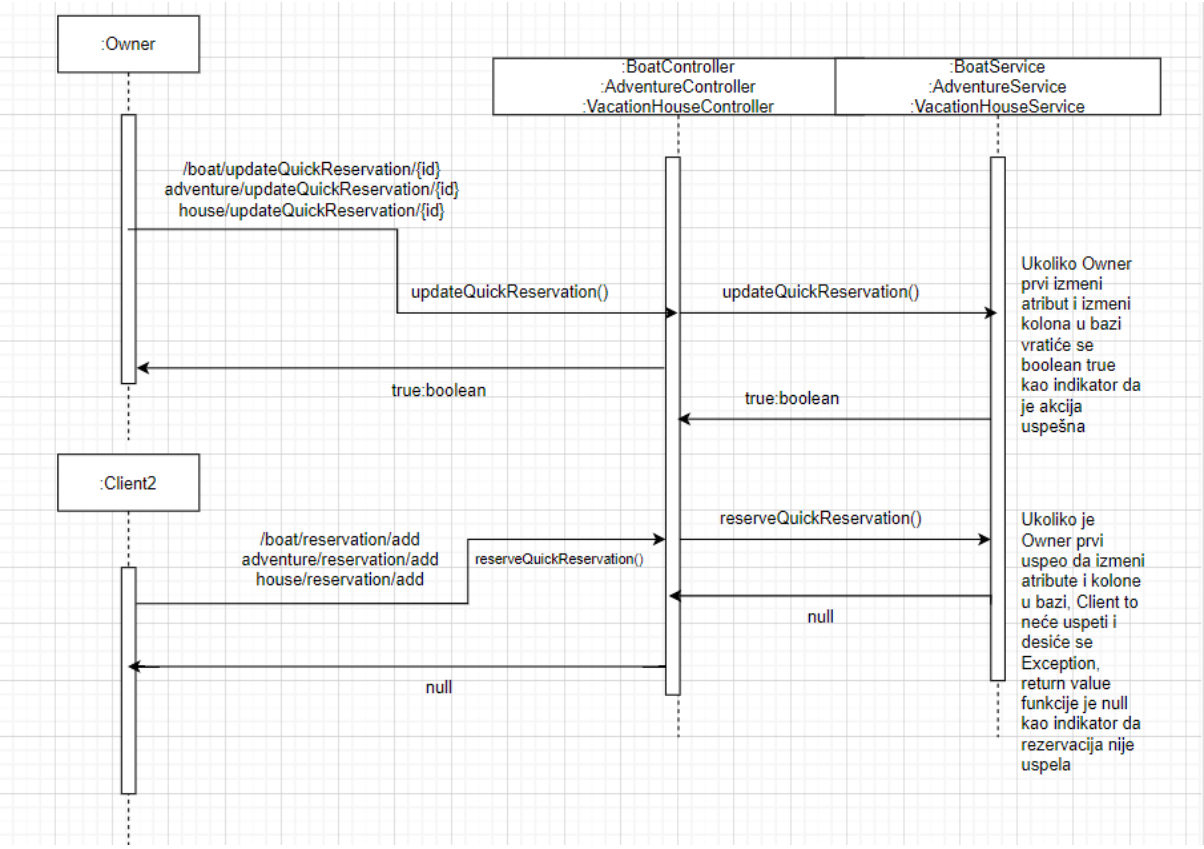
pokušavamo trenutno sačuvati u bazu, desiće se *ObjectOptimisticLockingFailureException* koji je potrebno uhvatiti i obavestiti klijenta da trenutno nije moguće obaviti rezervaciju akcije.

3. Konflikt

g. Opis konfliktne situacije

Klijent rezerviše akciju koju poseduje vikendica/brod/avantura dok je njihov vlasnik uređuje

h. Tok zahteva



i. Rešenje

Kako i uređivanje akcije predstavlja izmenu njenih atributa koje se obavlja izmenom kolona entiteta u bazi podataka, i ovde je vršeno optimistično zaključavanje.

Funkcija *updateQuickReservation* čuva izmenjenu akciju pozivom funckije *saveQuickReservationAsReservation* koja je opisana u prethodnom konfliktu. Ukoliko vlasnik vikendice/broda ili instruktor pecanja ažurira akciju, klijent koji je pokušao da rezerviše akciju prilikom poziva funkcije *saveQuickReservationAsReservation* iz funkcije *reserveQuickReservation* neće uspeti zato što se verzije objekata koji predstavljaju akciju neće poklapati iz razloga što je vlasnik već izvršio izmenu nad entitetom.

Ovakvom implementacijom rešen je i problem kada klijent prvi rezerviše akciju, a zatim vlasnik vikendice/broda ili instruktor pecanja proba da izmeni tu akciju.