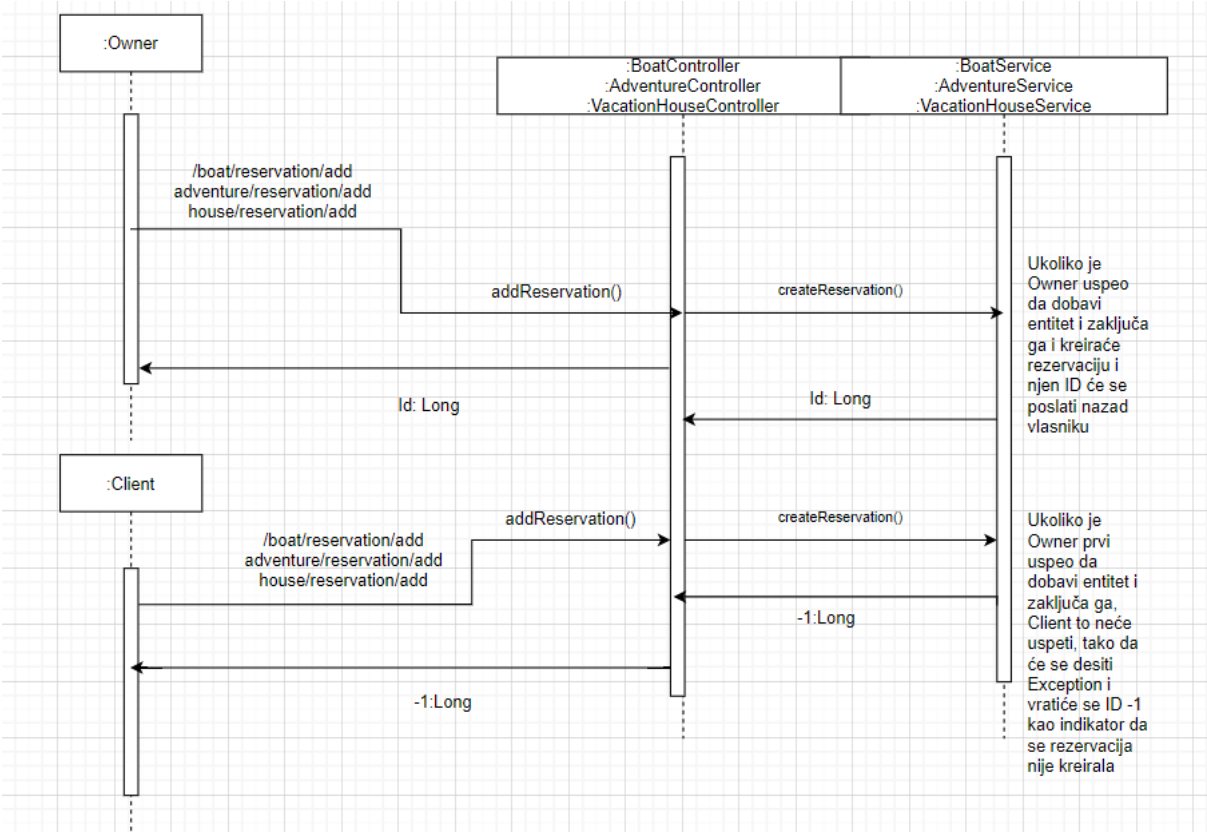
Student 2 - Aleksandra Nedić SW 27/2019

1. Konflikt

a. Opis konfliktne situacije

Vlasnik vikendice/broda ili instruktor pravi rezervaciju u isto vreme kad i drugi klijent

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.

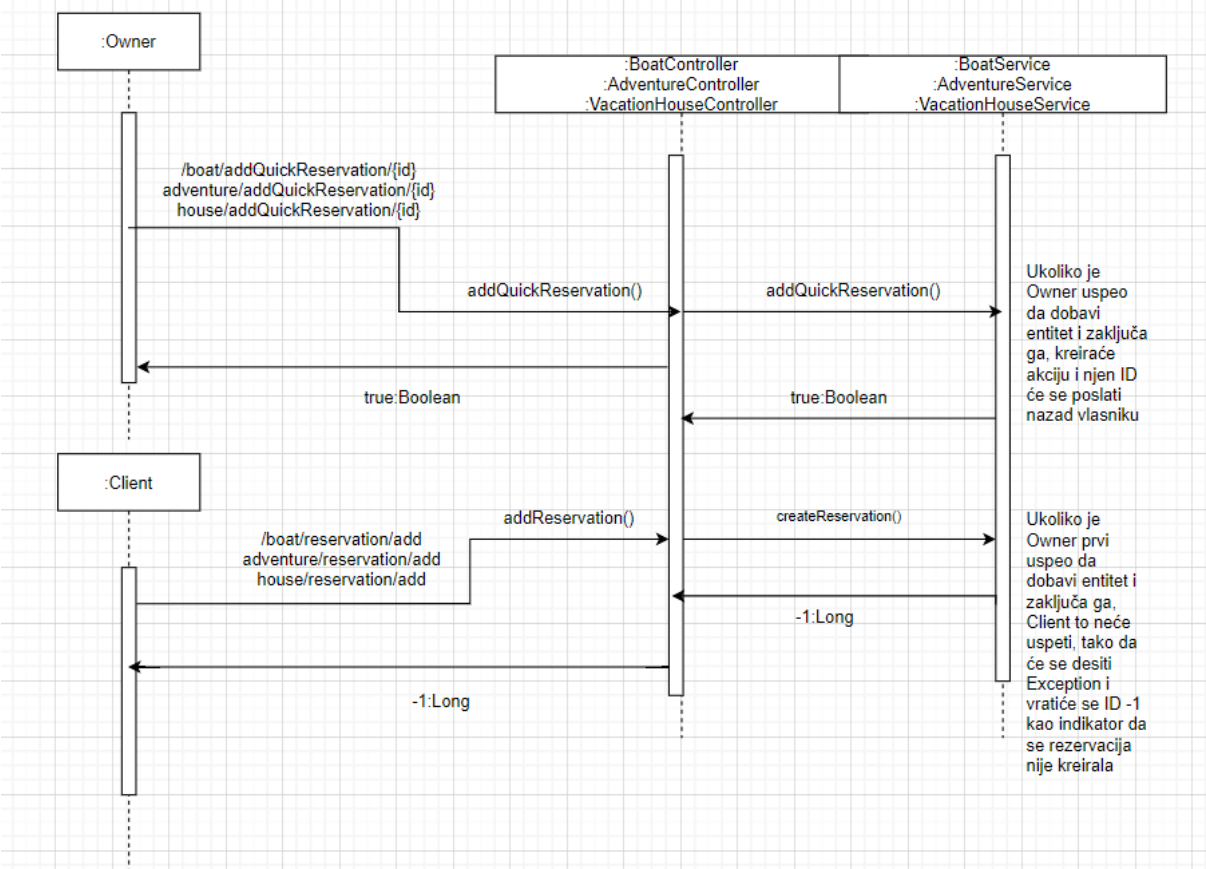
Na ovaj način rešen je i problem kada više klijenata rezerviše entitet u isto vreme s obzirom na to da se pozivaju identične funkcije.

2. Konflikt

d. Opis konfliktne situacije

Vlasnik vikendice/broda ili instruktor pravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta

e. Tok zahteva



f. Rešenje

Za rešavanje ovog problema odabrano je pesimističko zaključavanje. Razlog za to je što se prilikom pravljenja akcije kreira novi objekat *AdventureReservation/BoatReservation/VacationHouseReservation* koji se dodaje entitetu koji se rezerviše. Razlika u ovom objektu kod akcija i regularnih rezervacija jeste ta što akcija ima atribut *isQuickReservation* postavljen na *true*, i atribut *Client* postavljen na *null*. Kako se ažurira entitet time što se atributu *reservations:list* dodaje objekat i pri tom se dodaje novi red u bazu podataka, čitav entitet je potrebno zaključati.

Prilikom kreiranja akcije, poziva se funkcija *addQuickReservation* i ona je transakciona. U ovoj funkciji dobavlja se entitet iz baze kome se dodaje akcija. 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 vlasnika vikendice/broad ili instruktora pecanja da trenutno nije moguće dodati akciju. Ovaj entitet će biti zaključan u slučaju da klijent pravi rezervaciju,

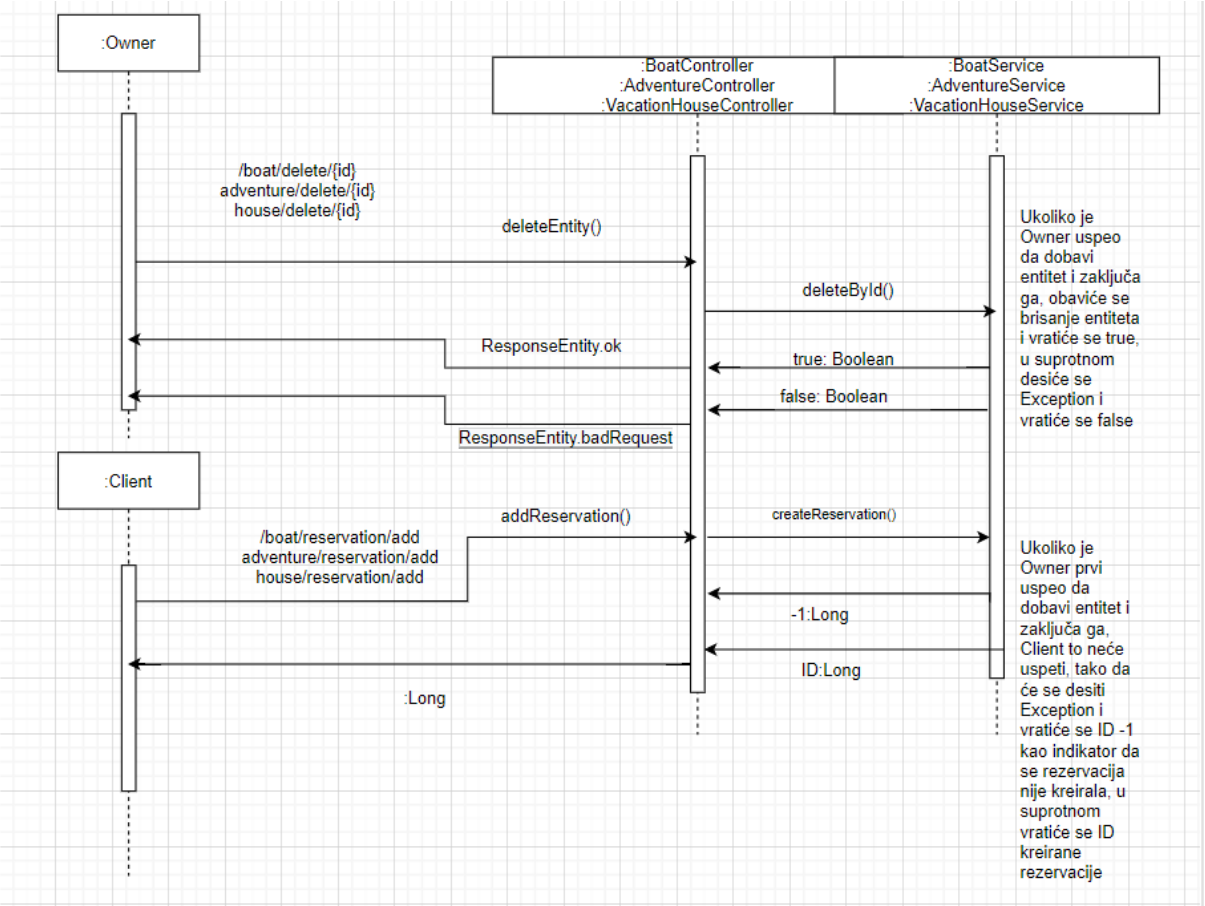
pozivajući funkciju *createReservation* koja u sebi sadrži transakcionu metodu *createFromDTO*.

3. Konflikt

g. Opis konfliktne situacije

Vlasnik vikendice/broda ili instruktor briše entitet u isto vreme kad je klijent rezerviše

h. Tok zahteva



i. Rešenje

Iako se brisanje entiteta obavlja logički, tj potrebno je izmeniti kolonu *deleted* u bazi podataka, što bi predstavljalo optimističko zaključavanje, za rešavanje ovog problema odabrano je pesimističko zaključavanje. Razlog za to je što rezervisanje entiteta predstavlja dodavanje novog reda u bazu, pa samim tim se zaključava ceo entitet. Iz tog razloga, pre brisanja entiteta, potrebno je proveriti da li je entitet koji vlasnik vikendice/broda ili instruktor pecanja moguće dobaviti.

Prilikom brisanja entiteta poziva se funckija *deleteById* i ona je transakciona. U ovoj funkciji dobavlja se entitet iz baze koji želimo obrisati. 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 vlasnika vikendice/broad ili instruktora pecanja da trenutno nije moguće brisanje entiteta.

Entitet će biti zaključan u slučaju da klijent pravi rezervaciju, pozivajući funkciju *createReservation* koja u sebi sadrži transakcionu metodu *createFromDTO*.