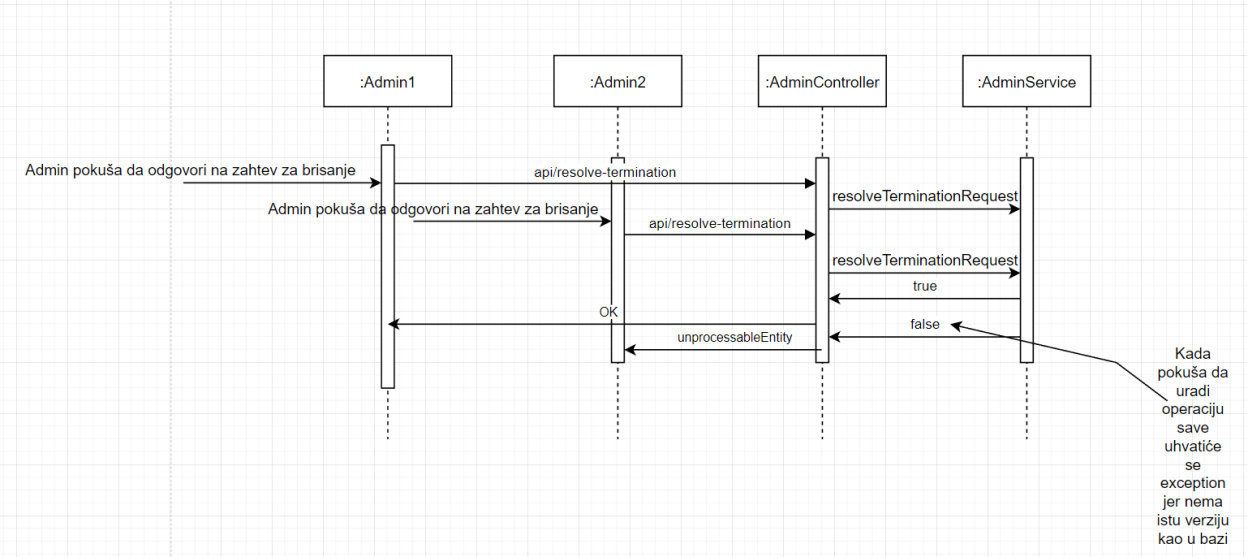
Student 3 –Pavle Glušac SW 19/2019

Situacija 1: na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema

Ukoliko stigne zahtev za brisanje naloga admini koji trenutno rade mogu odgovoriti na taj zahtev. Moguće je da više njih odjednom to uradi, što se mora sprečiti.

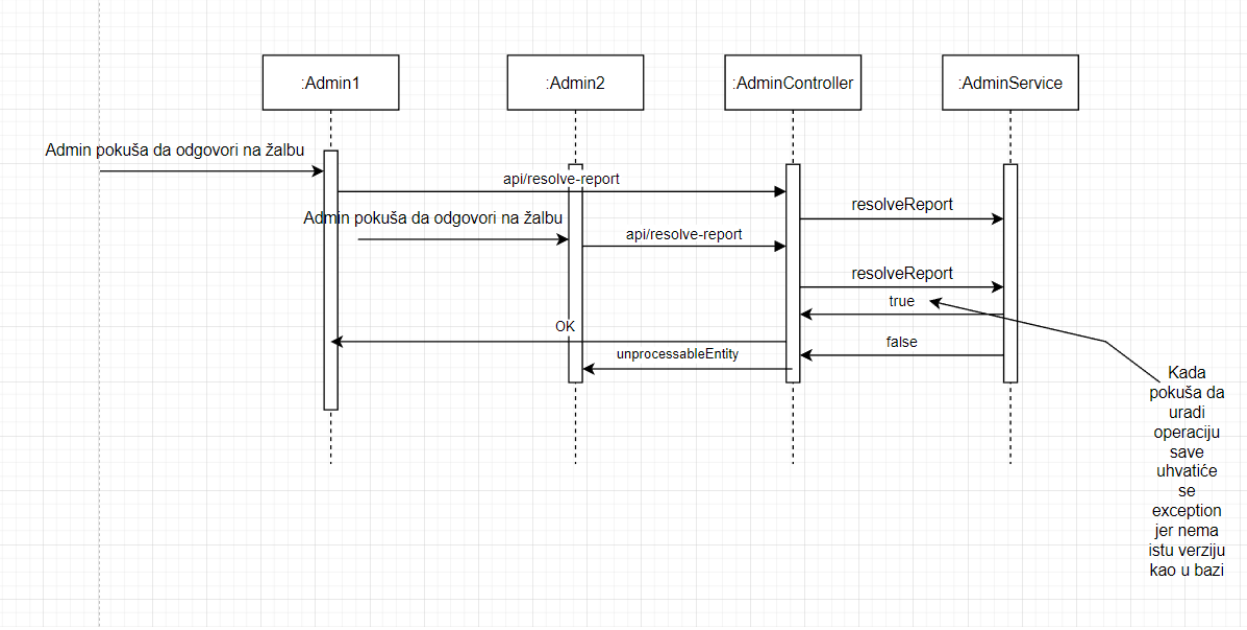
Obzirom da su zahtevi za brisanje ne toliko česti, kao i data situacija, i uzevši da je dovoljno proveriti samo verzije, odabran je optimistic lock. Kao što se vidi na dijagramu, dva admina u isto vreme pokušaju da odgovore na zahtev za brisanje. Nakon što prvi uspešno odgovori, u drugom zahtevu će se desiti OptimistickLockException pri operaciji save() jer verzija u entitetu nije ista.



Situacija 2: na jednu žalbu može da odgovori samo jedan administrator sistema.

Ukoliko stigne žalba admini koji trenutno rade mogu odgovoriti na tu žalbu. Moguće je da više njih odjednom to uradi, što se mora sprečiti.

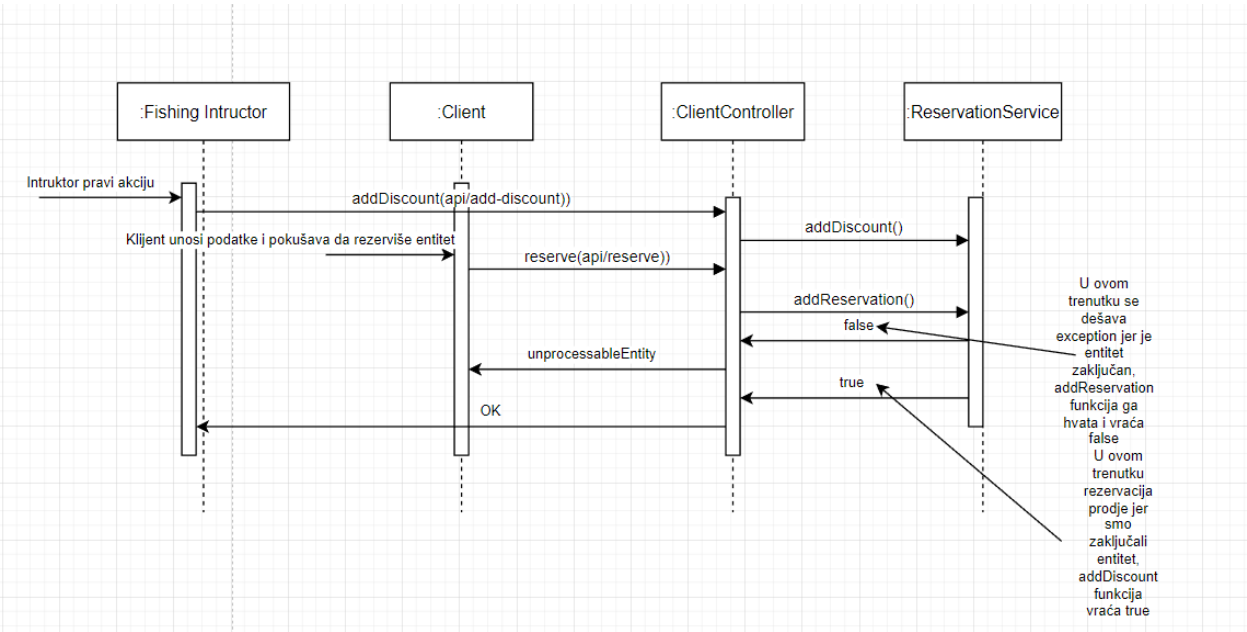
Ista logika se primenjuje kao i kod zahteva za brisanje, odabran je optimistic lock. Kao što se vidi na dijagramu, dva admina u isto vreme pokušaju da odgovore na žalbu. Nakon što prvi uspešno odgovori, u drugom zahtevu će se desiti OptimistickLockException pri operaciji save() jer verzija u entitetu nije ista.



Situacija 3: instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent

Ukoliko instruktor i klijent istovremeno žele da rezervišu isti entitet u isto vreme, to moramo sprečiti. Ovo je veoma realan problem naročito za vrlo popularno entitete koji su retko dostupni.

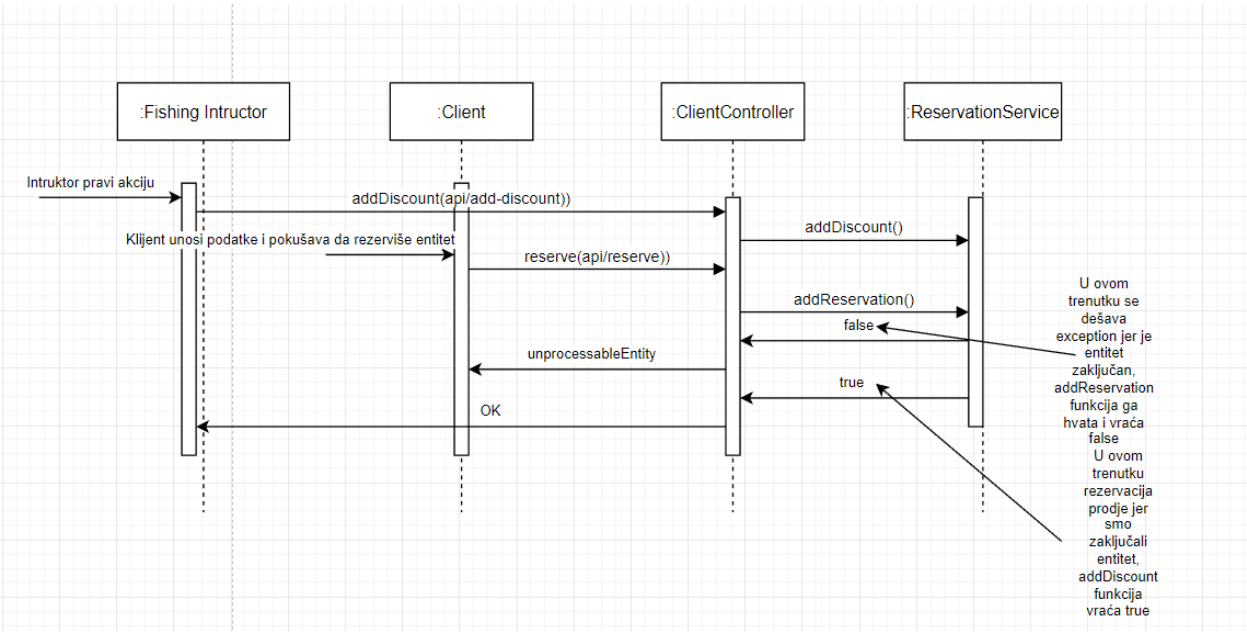
Ovo ćemo rešiti pomoću pessimistic locka. Polazimo od pretpostavke da će do kolizije dolaziti često jer entiteti mogu biti veoma popularni, i naša aplikacija može imati mnogo korisnika. Zaključavamo red u tabeli Rentable, dok insertujemo u tabelu Reservations. Takođe znamo da je pessimistic lock bolji ukoliko očekujemo mnogo konflikata. Zbog mogućnosti deadlocka izabran je pessimistic_write.



Situacija 4: klijent ne može da napravi rezervaciju u isto vreme kada i instruktor napravi akciju

Ukoliko instruktor želi da napravi akciju i klijent želi da rezerviše za isti entitet u isto vreme, to moramo sprečiti. Ovo je veoma realan problem naročito za vrlo popularno entitete koji su retko dostupni.

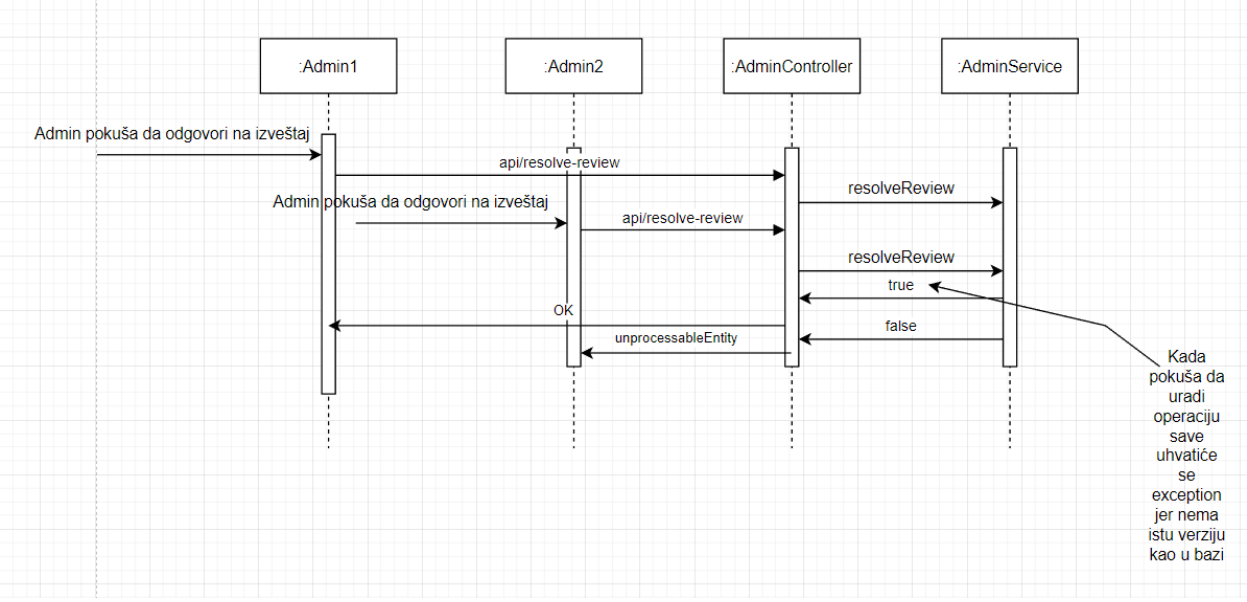
Ovo ćemo takođe rešiti pomoću pessimistic locka. Logika je ista kao i kod rezervacija. Očekujemo da će akcije biti vrlo tražene i da će mnogo korisnika želeti da ih zakažu čim dobiju obaveštenje, te je izabran pessimistic lock. Zbog mogućnosti deadlocka izabran je pessimistic_write.



Situacija 5: na jednu izveštaj može da odgovori samo jedan administrator sistema.

Ukoliko stigne izveštaj admini koji trenutno rade mogu odgovoriti na tu žalbu. Moguće je da više njih odjednom to uradi, što se mora sprečiti.

Ista logika se primenjuje kao i kod zahteva za brisanje, odabran je optimistic lock. Kao što se vidi na dijagramu, dva admina u isto vreme pokušaju da odgovore na žalbu. Nakon što prvi uspešno odgovori, u drugom zahtevu će se desiti OptimistickLockException pri operaciji save() jer verzija u entitetu nije ista.


:Admin1
:Admin2
:AdminController
:AdminService
Admin pokuša da odgovori na izveštaj
api/resolve-review
resolveReview
Admin pokuša da odgovori na izveštaj
api/resolve-review
resolveReview
true
OK
false
unprocessableEntity
Kada pokuša da uradi operaciju save uhvatiće se exception jer nema istu verziju kao u bazi