Konkurentni pristup resursima u bazi

Nevena Radešić SW12/2019
Student 1

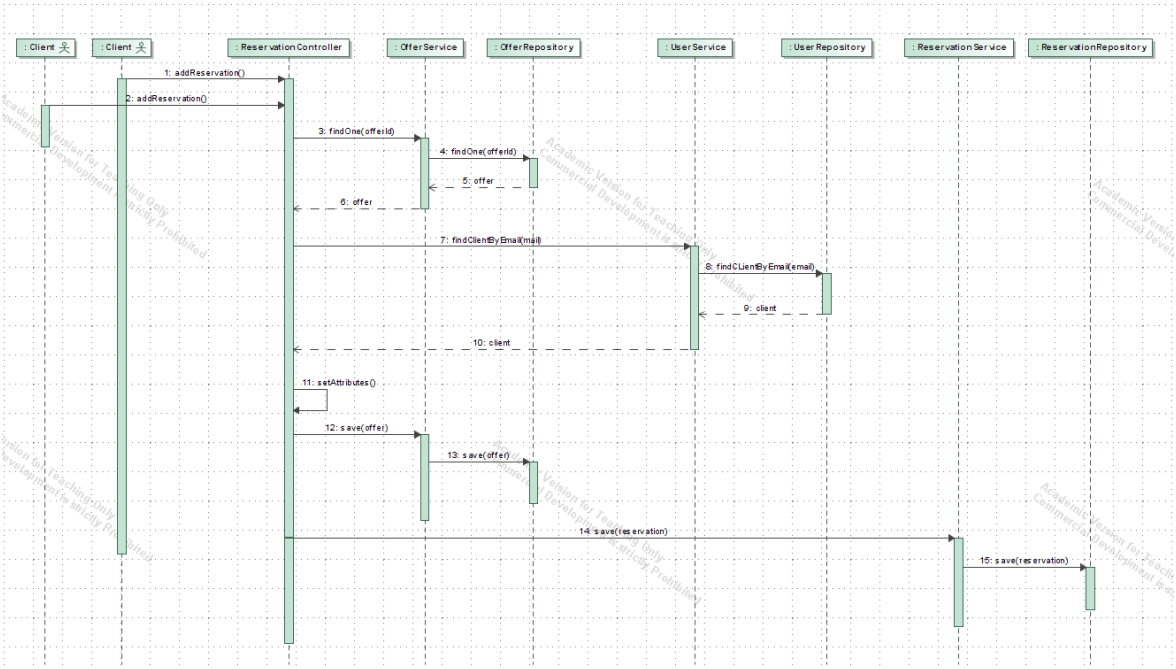
Konfliktna situacija 1

Opis problema

Potrebno je riješiti konfliktnu situaciju u okviru koje dva klijenta istovremeno prave rezervaciju istog entiteta za isti ili preklapajući vremenski period. Ova situacija je problematična jer može dovesti do toga da se u istom terminu za istu ponudu kreira više rezervacija, što nije dozvoljeno.

Model situacije

Dva klijenta istovremeno iniciraju kreiranje rezervacije. Tada se poziva metoda *addReservation* iz klase *ReservationController*, te se iz baze dobavljaju ponuda i klijent za kojeg se rezervacija pravi. Nakon postavljanja svih potrebnih atributa novokreirane rezervacije, ona se čuva u bazi. Problem je u tome što se sada može desiti da su za ponudu kreirane dve rezervacije u istom terminu.



Dijagram 1 – dva klijenta rezervišu etitet u isto vrijeme

Rješenje

Navedeni problem je riješen upotrebom optimističkog zaključavanja ponude (vikendice, broda ili avanture), tako što je u klasi *Offer* dodat atribut *version*, koji predstavlja verziju objekta. Klasi *Offer* je takođe dodata promenljiva koja predstavlja brojač rezervacija. Kada se nova rezervacija doda, brojač se inkrementira i promena se sačuva u bazi, što dovodi do povećanja vrijednosti atributa *version*. Zahvaljujući tome, u slučaju da dva klijenta pokušaju napraviti rezervaciju u isto vrijeme, samo će se jedna sačuvati. Prilikom pokušaja druge izmjene doći će do *OptimisticLockingFailureException*-a. Dakle, ovaj mehanizam nam je samo obezbijedio provjeru da li su se desile dve rezervacije istovremeno, ali nismo provjerili da li su same rezervacije u istom periodu. Bilo je potrebno dozvoliti rezervacije koje su se desile istovremeno u slučaju da se ne odnose na isti period. Stoga smo nakon bacanja *OptimisticLockingFailureException*-a, u *catch* blok ubacili nove provjere. U slučaju da se rezervacije ipak nisu odnosile na isti period, rezervacija se dodavala u bazu.

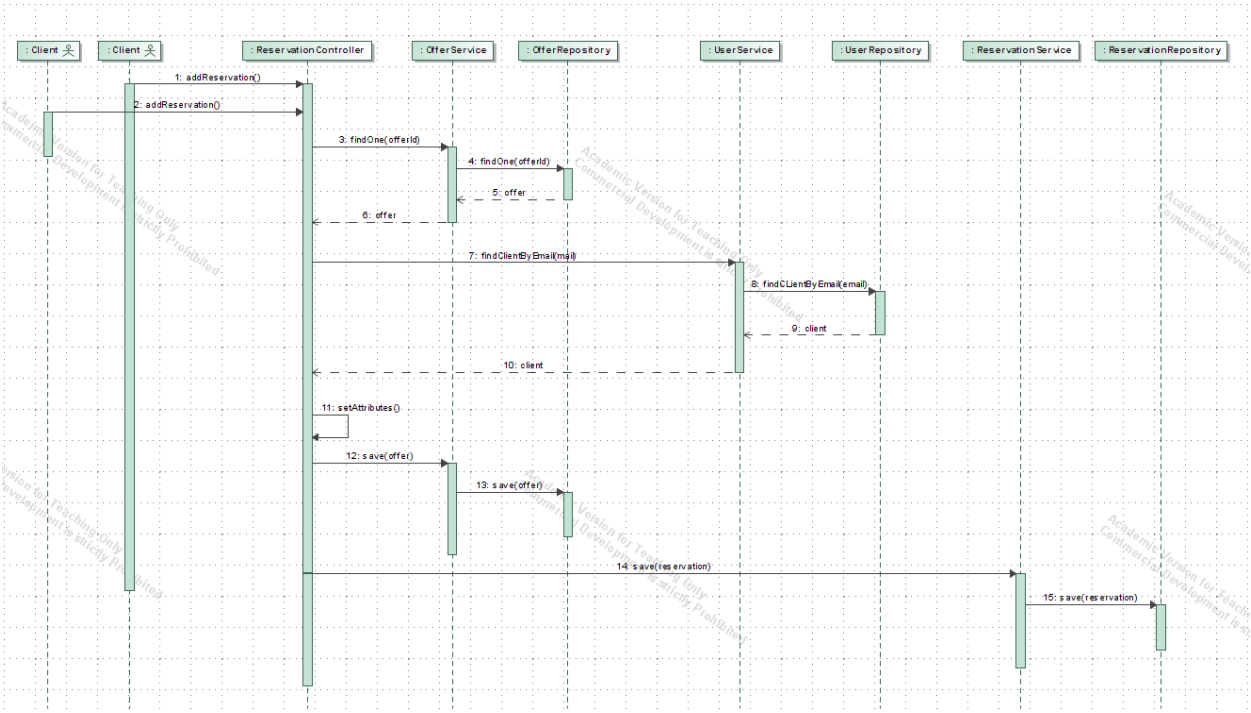
Konfliktna situacija 2

Opis problema

Potrebno je riješiti konfliktnu situaciju u okviru koje dva klijenta istovremeno rezervišu entitet na akciji. Ova situacija je problematična, jer može dovesti do toga da za isti termin, istog entiteta, dva različita klijenta imaju rezervacije.

Model situacije

Dva klijenta istovremeno iniciraju kreiranje brze rezervacije. Tada se poziva metoda *addReservation* iz klase *ReservationController*, te se iz baze dobavljaju ponuda i klijent za kojeg se rezervacija pravi. Nakon postavljanja svih potrebnih atributa novokreirane rezervacije, ona se čuva u bazi. Problem je u tome što se sada može desiti da su za ponudu kreirane dve rezervacije u istom terminu.



Dijagram 2 – dva klijenta rezervišu entitet na akciji u isto vrijeme

Rješenje

Navedeni problem je riješen upotrebom optimističkog zaključavanja ponude (vikendice, broda ili avanture), tako što je u klasi Offer dodat atribut *version*, koji predstavlja verziju objekta. Klasi Offer je također dodata promjenljiva koja predstavlja brojač rezervacija. Kada se nova rezervacija doda, brojač se inkrementira i promjena se sačuva u bazi, što dovodi do povećanja vrijednosti atributa *version*. Zahvaljujući tome, u slučaju da dva klijenta pokušaju napraviti rezervaciju u isto vrijeme, samo će se jedna sačuvati. Prilikom pokušaja druge izmjene doći će do OptimisticLockingFailureException-a. Dakle, ovaj mehanizam nam je samo obezbijedio provjeru da li su se desile dve rezervacije istovremeno, ali nismo provjerili da li su same rezervacije u istom periodu. Bilo je potrebno dozvoliti rezervacije koje su se desile istovremeno u slučaju da se ne odnose na isti period. Stoga smo nakon bacanja *OptimisticLockingFailureException*-a, u *catch* blok ubacili nove provjere. U slučaju da se rezervacije ipak nisu odnosile na isti period, rezervacija se dodavala u bazu.

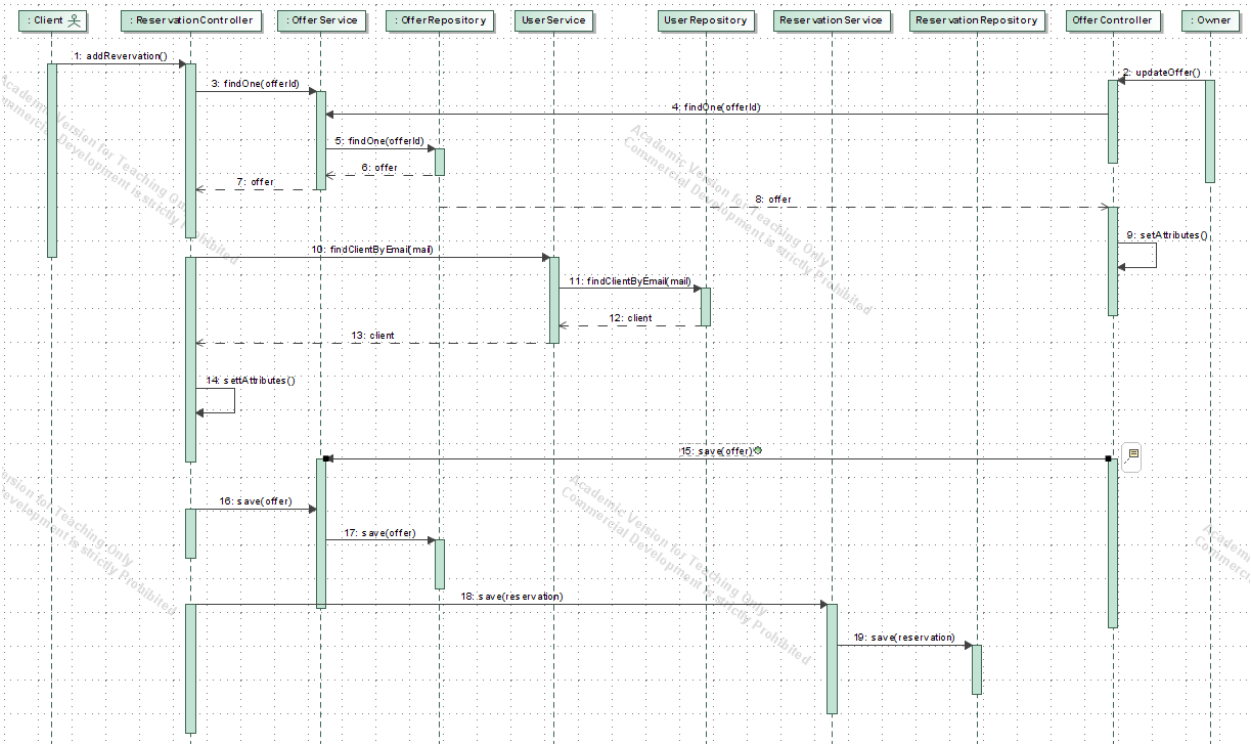
Konfliktna situacija 3

Opis problema

Potrebno je riješiti konfliktnu situaciju u okviru koje vlasnik vikendice ili broda dodaje novi cjenovnik za svoju ponudu isto vrijeme kad i klijent rezerviše tu istu ponudu. Ova situacija je problematična, jer može dovesti do toga da klijent rezerviše ponudu po drukčijoj cijeni od one koju će on da plati ili do toga da vlasnik želi da promijeni podatke koje ne bi trebalo, jer je rezervacija već izvršena.

Model situacije

Vlasnik vikendice ili broda inicira izmjenu cijene ponude u isto vrijeme kada klijent inicira kreiranje rezervacije za istu ponudu. Prilikom izmjene cijene, prvo se poziva metoda *updateOffer* klase *OfferService*. Zatim se iz baze dobavlja ponuda, te se postavljaju njena izmjenjena svojstva. Na kraju se izmjenjena ponuda čuva u bazi. Dodavanje rezervacije klijenta se obavlja na isti način kao u prethodne dve situacije. Na ovaj način je došlo do toga da je klijent rezervisao ponudu sa cijenom koja više ne važi.



Dijagram 3 – korisnik rezerviše entitet čiji cjenovnik vlasnik mijenja

Rješenje

Problem je riješen upotrebom optimističkog zaključavanja ponude (vikendice, broda) tako što je u klasi *Offer* dodat atribut *version*, koji predstavlja verziju objekta.

Klasi *Offer* je takođe dodata promenljiva koja predstavlja brojač istorije cijena. U slučaju dodavanja nove cijene, brojač se inkrementira i promjena se sačuva u bazi, što dovodi do povećanja vrijednosti atributa *version*. Zahvaljujući tome, u slučaju da klijent pokuša napraviti rezervaciju u isto vrijeme kad vlasnik pokuša izmijeniti cjenovnik, samo jedna od tih promjena će se sačuvati. Prilikom pokušaja druge izmjene doći će do *OptimisticLockingFailureException*-a. Nije izabrano pesimističko zaključavanje, jer bi ono više narušilo performanse aplikacije.