**KONKURENTANI PRISTUP RESURSIMA U BAZI**

1. *Vlasnik vikendice/broda ne može da napravi rezervaciju u isto vreme kad i drugi klijent*
   1. **Vlasnik vikendice ne može da napravi rezervaciju u isto vreme kad i drugi klijent**

1.1.1 Opis konfliktne situacije

Vlasnik vikendice ima mogućnost da za klijenta za kojeg trenutno traje rezervacija kreira novu rezervaciju, tako da se termin ove rezervacije, ne poklapa sa terminima postojećih rezervacija.

Konfliktna situacija nastaje ako i klijent i vlasnik, u isto vreme, izaberu isti period u kom žele da ostvare rezervaciju ili preklapajuće periode. Na taj način bi vikendica bila rezervisana za dva korisnika u isto vreme, što nije dozvoljeno.

1.1.2 Tokovi klijentskih zahteva i odgovora servera

*Slika i dijagram: KonfliktaSituacija1*

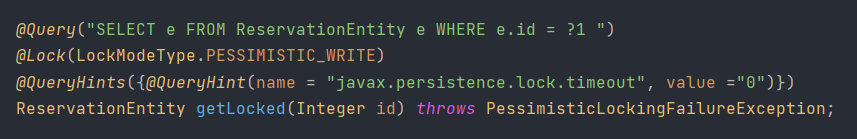
1.1.3 Opis načina na koji je konfliktna situacija rešena

Zaštita od mogućnosti ostvarenja ove situacije je realizovana principom pesimističkog zaključavanja. Sama implementacija je ostavarena u klasama *IReservationEntityRepository* i *ReservationService.*

Metoda koja se koristi za dobavljanje entiteta priliko kreiranja rezervacije je *getLocker(Integer id)* IReservationEntityRepository repozitorijuma. Korisnik čiji zahtev za kreiranje rezervacije stigne prvi će zaključati ovaj entitet, i kada drugi korisnik zatraži ovaj entitet dobiće *PessimisticLockingFailureException*, odnosno poruku da je entitet već rezervisan.

Implementacija:

* IReservationEntityRepository



Takođe, na nivou servisa metode koje pozivaju prethodno navedenu metodu repozitorijuma su označene kao transakcione, a sami pozivi repozitoriju se izvršavaju u okviru try/catch bloka.

* ReservationEntityService



Mana ovog pristupa: korisnik dobija poruku da je entitet već zauzet, uvek kada uporedo sa njim neko vrši rezervaciju istog entiteta, bez obzira da li su izabrali preklapajuće termine ili ne.

* 1. **Vlasnik broda ne može da napravi rezervaciju u isto vreme kad i drugi klijent**

1.2.1 Opis konfliktne situacije

Vlasnik broda ima mogućnost da za klijenta za kojeg trenutno traje rezervacija kreira novu rezervaciju, takođe, termin ove rezervacije, ne sme da se poklapa sa terminima postojećih rezervacija.

Razlika u načinu rezervacije u odnosu na rezervaciju vikendice:

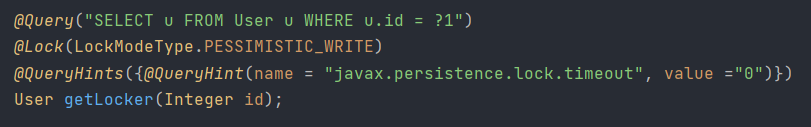
* Korisnik pored izbora broda za koji želi da izvrši rezervaciju bira da li želi da koristi usluge kapetana broda. Ukoliko ne želi ove usluge sam postupak rezervacije, kao i konfliktna situacija je identična kao za vikendicu.
* Međutim, ako korisnik ipak izabere ovu opciju može doći do još jedne konfliktne situacije, a to je da usluge kapetana broda budu rezervisane u isto vreme od strane dva različita korisnika(klijenta i vlasnika samog broda).

U trenutku kada korisnik zahteva slobodne termine za rezervaciju broda sa uslugama kapetana, njemu se nudi presek slobodnih termina samog broda i njegovog kapetana, unutar kojih on može da odabere period svoje rezervacije.

Konfliktna situacija nastaje ako i klijent i vlasnik, u isto vreme, izaberu isti period u kom žele da ostvare rezervaciju i unutar tog perioda izaberu isti početak i kraj rezervacije, ili preklapajuće periode. Na taj način bi, pored broda, i kapetan bio rezervisan od strane dva korisnika u isto vreme, što nije dozvoljeno.

Sam postupak implementacije je odrađen na isti način kao i kod rezervacije vikendice, primenom pesimističkog zaključavanja.

Metoda na kojoj je primenjeno zaključavanje je *getLocker(Integer id) IUserRepository* klase, čime bi se pri dobavljanju user-a, odnosno kapetana za kog se pravi rezervacija izvršilo samo zaključavanje korisnika.



1. *Vlasnik vikendice/broda ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg termina*
   1. Opis konfliktne situacije

Vlasnik broda/vikendice ili instruktor avanture, i klijent mogu u isto vreme da prave akciju i kreiraju rezervaciju za isti entitet. Ako se termini koje su odabrali poklapaju ili preklapaju, dolazi se do situacije da će entitet biti rezervisan u isto vreme od strane dva različita klijenta, što nije dozvoljeno.

* 1. Tokovi klijentskih zahteva i odgovara

*Slika i dijagram: KonfliktaSituacija2*

* 1. Opis načina na koji je konfliktna situacija rešena

U našem sistemu je postupak kreiranja rezervacije i postupak kreiranja akcije odrađen na sličan način, tj. prvi korak u postupku kreiranja i jedne i druge su identični, pa je samim tim i postupak zaštite ove konfliktne situacije obuhvaćen zaštitom od preklapanja rezervacija.

Implementacija zaštite je odrađena primenom pesimističkog zaključavanja samog entiteta za koji se vrši kreiranj same akcije ili rezervacije.

1. *Vlasnik vikendice/broda ne može da dodeljuje penale za nepojavljivanje u zakazanom terminu u isto vreme kada penale tom klijentu dodeljuje administrator*
   1. Opis konfliktne situacije

Vlasnik vikendice/broda ima mogućnost da direktno dodeli penal klijentu ako se nije pojavio u terminu zakazanom rezervacijom. Administrator sistema u okviru analize izveštaja o rezervacijama koje dobija od vlasnika entiteta, ako je vlasnik tražio da klijent bude sankcionisan, ima mogućnost da dodeli penal klijentu. Ako vlasnik entiteta želi da dodeli penal za nepojavljivanje u terminu, i u istom trenutku administrator sistema tom istom klijentu dodelju penal za neku drugu rezervaciju, klijentu će biti dodeljen samo jedan penal, umesto dva.

* 1. Tokovi klijentskih zahteva i odgovora

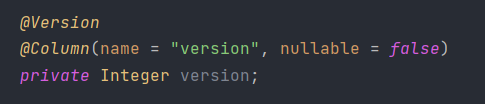
*Slika i dijagram: KonfliktaSituacija3*

* 1. Opis načina na koji je konfliktna situacija rešena

Zaštite od ove konfliktne situacije je realizovana primenom optimističkog zaključavanja. U model klasi Penalty je dodato novo polje koje predstavlja verziju, jer se samo dodeljivanje penala vrši ažuriranjem trenutnog broja penala, što će verzija ispratiti.

Implementacija:

* Penalty – model klasa



Sama metoda servisa u kojoj se ostvaruje ažuriranje penala je realizovana unutar try/catch bloka, zarad mogućnosti pojavljivanja OptimisticEntityLockException-a, koja je rezultat optimističkog zaključavanja.

* PenaltyService

