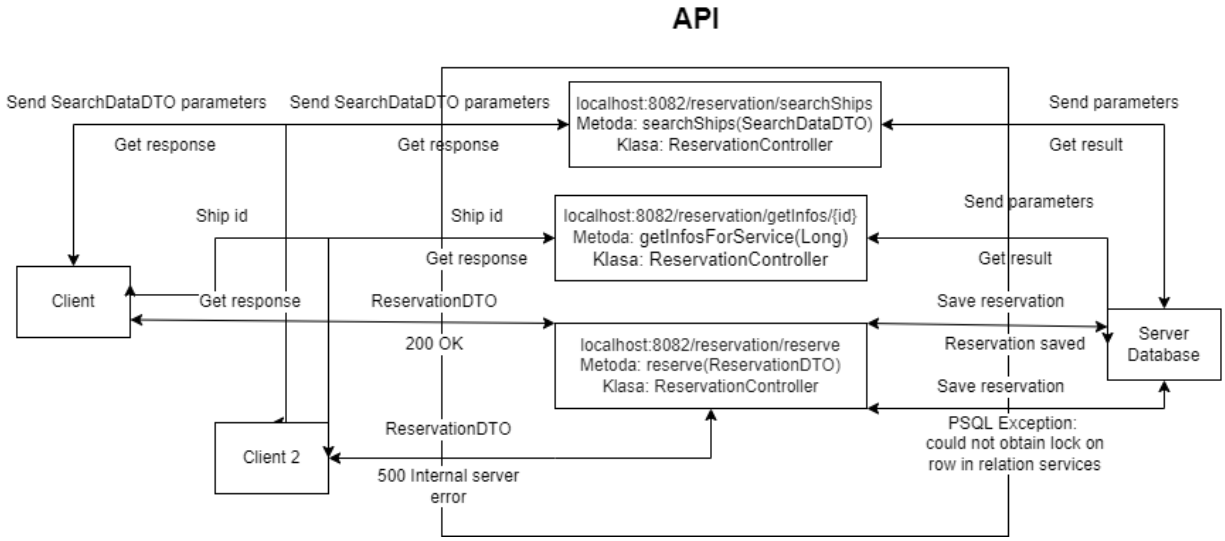
Rezervacija

Do konflikta dolazi, ako dva ili više korisnika pokušaju da kreiraju rezervaciju nad istim servisom (vikendice, brodovi…) u preklapajuće vreme.

Pošto prilikom sačuvavanja rezervacije, vršim čuvanje servisa, odnosno menjam period dostupnosti servisa, odlučio sam da metodu getServiceById sa anotacijom @Lock(LockModeType.PESSIMISTIC_WRITE) učinim metodom koja će zaključati servis, koji se rezerviše. Tj. koristio sam pesimističko zaključavanje. Izabrao sam ovaj tip zaključavanja jer sprečava druge transakcije da vrše bilo kakve izmene nad servisom. Da bih se dodatno osigurao postavio sam timeout na 0 tako da odma dobijam grešku, bez ikakvog čekanja, ukoliko neka druga transakcija pokuša da vrši izmene. Pored toga dodao sam i polje version u Service klasu tako da imam duplu protekciju. Takođe u samoj metodi za rezervaciju imam proveru da li rezervacija sa datim periodom za dati servis već postoji, ukoliko je to slučaj metoda se prekida, vraća Bad Request.

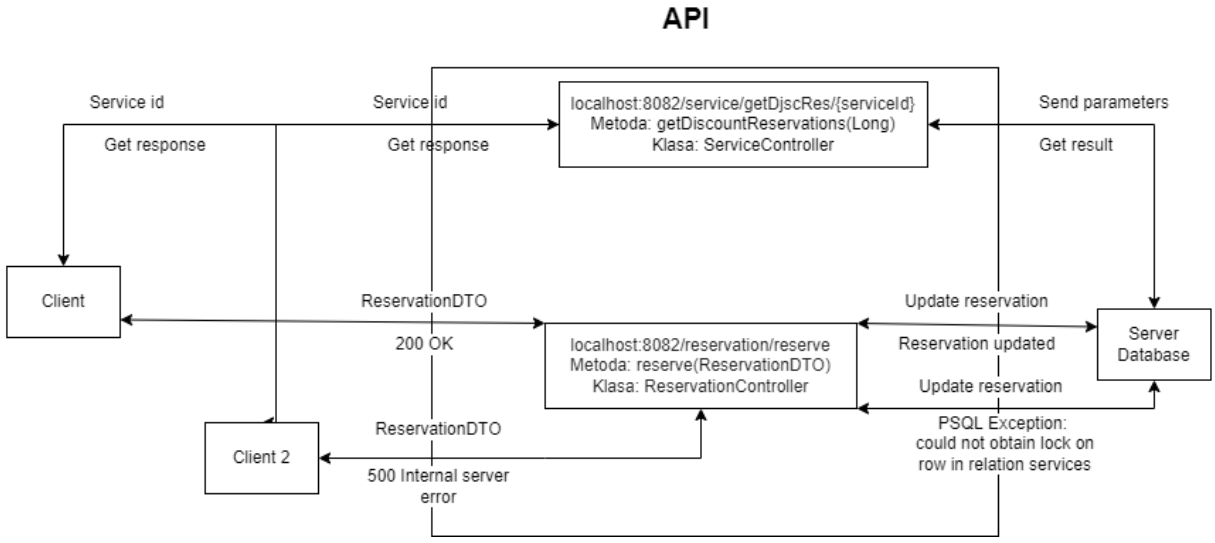


Tok podataka za rezervaciju servisa

Izabrao sam pristup pesimističkog zaključavanja, jer time sprečavam bilo kakve neželjene promene nad servisom koji predstavlja okosnicu cele metode za rezervaciju.Napisao sam jedan test za rezervaciju, ali samo sa version atributom, bez pesimističkog zaključavanja, što takođe samo po sebi radi, baca ObjectOptimisticLockingFailureException.

Rezervacija akcija

S obzirom da i akciju i običnu rezervaciju vršim u istoj metodi, pristup je vrlo sličan kao za običnu rezervaciju. I dalje imam getServiceById metodu kojom zaključavam servis za koji pravim rezervaciju. Jedina novina je što sam i u samu rezervaciju dodao version atribut. Pošto rezervacija akcija već postoji u bazi, samo vršim ažuriranje nekih polja, pa sam zbog toga dodao version atribut. Ali pošto i ovde menjam period dostupnosti servisa odlučio sam se za pesimističko zaključavanje.

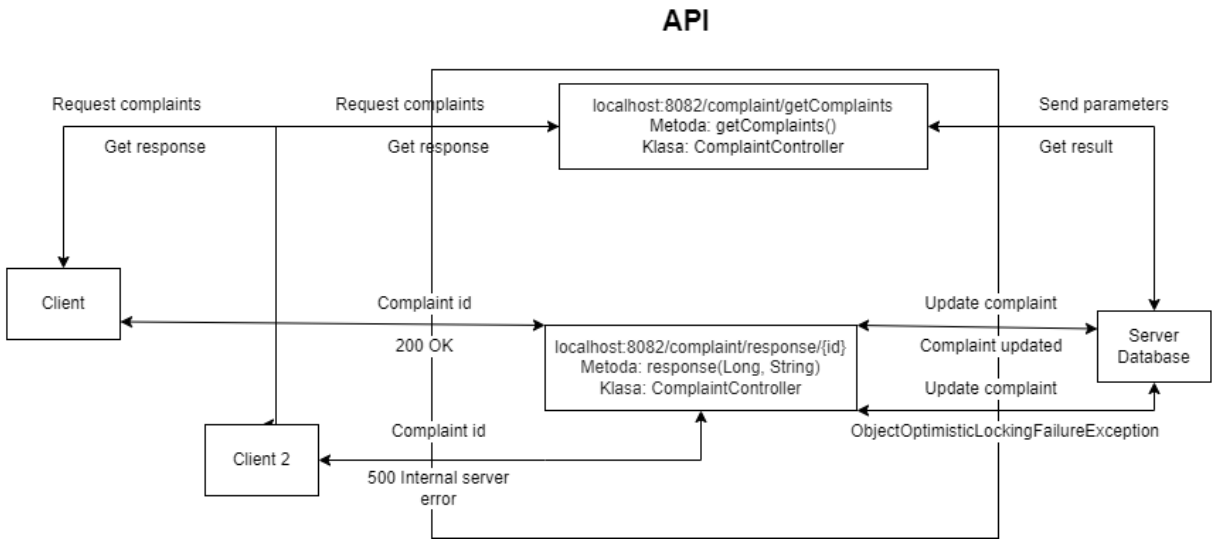


Tok podataka za rezervaciju akciju

Žalba

Konflikt nastaje, ako dva korisnika, admina u ovom slučaju pokušaju da u isto vreme odgovore na žalbu, tj. da ažuriraju polje u bazi.

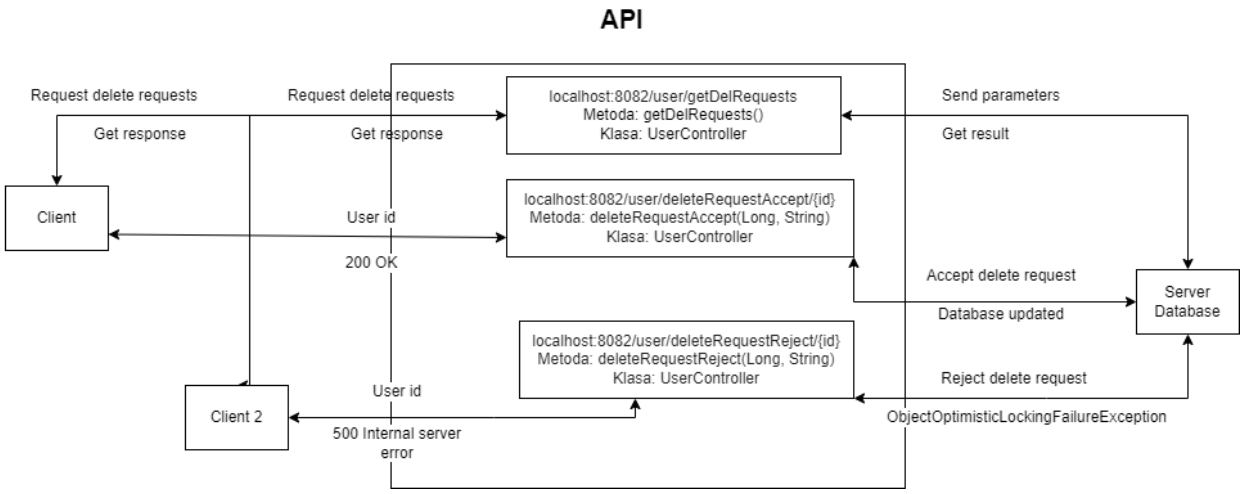
Žalba u mom slučaju predstavlja vrlo jednostavnu funkcionalnost. Sve se svodi na ažuriranje jedne kolone u bazi i slanje maila. Iz tog razloga smatram da je dovoljno samo dodati version atribut u klasu Complaint. Takođe sam napisao jedan test i za žalbe i dobijam isti izuzetak kao i kod rezervacije ObjectOptimisticLockingFailureException.



Tok podataka za ažuriranje statusa žalbe

Zahtev za brisanje naloga

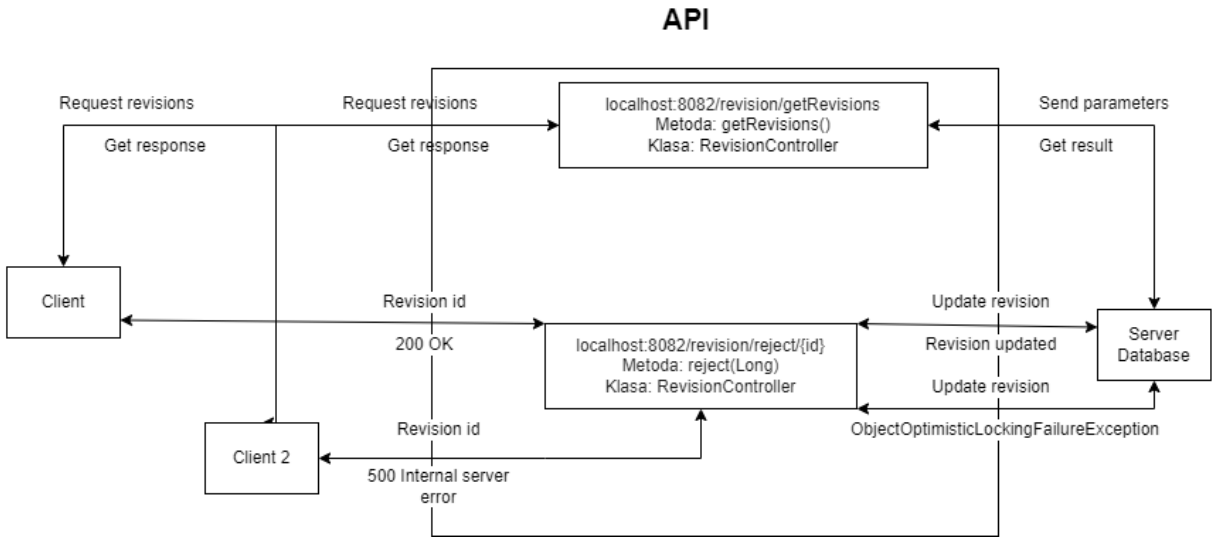
Slično kao u prethodnom slučaju konflikt se javlja ukoliko više admina pokuša da odobri ili odbije zahtev za brisanjem naloga. Ipak ovo je malo složenija funkcionalnost jer prilikom brisanja recimo nekog prodavca (vlasnik vikendice, broda...) brišu se svi njegovi servisi koji nemaju nikakvih rezervacija, ako je zahtev za brisanjem odobren, ili prilikom brisanja klijenta brišu se sve njegove rezervacije. Ali pošto to nisu neke izmene koje utiču na neke druge entitete kao što je slučaj prilikom kreiranja rezervacije, odlučio sam se za pristup optimističkog zaključavanja dodavanjem version atributa u klasu DeleteRequest. Svakako se sve izmene nad entitetima poništavaju ukoliko transakcija baci bilo kakav izuzetak. Takođe sam napisao jedan test i za ovu funkcionalnost, samo što sam stavio da jedna nit radi prihvatanje, a druga odbijanje. Rezultat je opet ObjectOptimisticLockingFailureException.



Tok podataka prihvatanja/odbijanja zahteva za brisanje naloga

Ocenjivanje

Funkcionalnost koju sam izabrao sam da ispitam je ocenjivanje. Budući da je dosta slična prethodnoj opet sam se odlučio za pristup optimističkog zaključavanja dodavanjem atributa version u klasu Revision. Opet imamo dve funkcije za prihvatanje odnosno odbijanje revizije. Razlike između dve funkcije su što se kod prihvatanja vrši ažuriranje prosečne ocene prodavca odnosno servisa i slanje maila, a zajedničko im je ažuriranje statusa revizije. Napisao sam jedan test koji baca ObjectOptimisticLockingFailureException u slučaju kada dva korisnika pokušaju da odbiju istu reviziju.



Tok podataka odbijanja revizije

Problemi

Verovatno najveći problem, bio je izbor tipa zaključavanja i entiteta koje bi trebalo zaključati. Recimo, u početku sam pokušavao da zaključam rezervaciju, prilikom njenog kreiranja, dok nisam shvatio da to nije dobar pristup. Još jedan od problema bilo je određivanje minimalnog vremena uspavljivanja niti za učitavanje podataka iz druge niti i radi preklapanja radnog vremena dve niti kako bi uspešno simulirali dva korisnika koji pristupaju istoj funkcionalnosti.

Zavisno od brzine izvršavanja metoda za dobavljanje odnosno čuvanje podataka, kao i brzine slanja maila dobijaju se različiti rezultati za različito vreme uspavljivanja niti.