

## Konfliktne situacije – Student 1

### Konfliktna situacija 1:

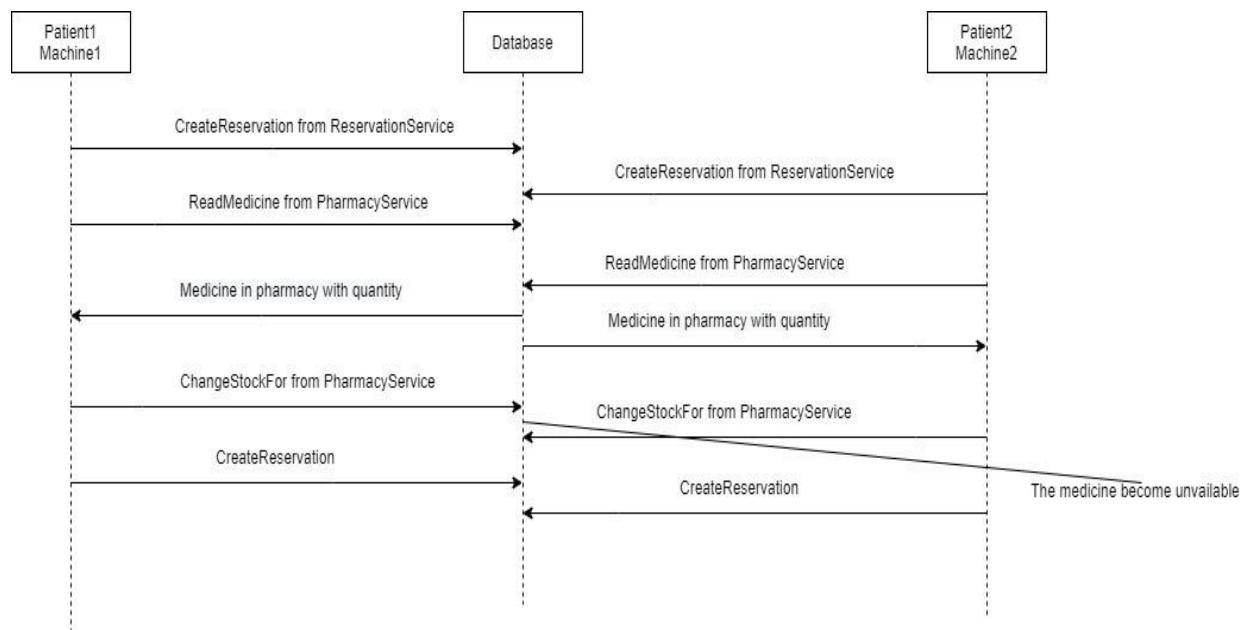
#### Opis konfliktne situacije

Više istovremenih korisnika aplikacije ne može da rezerviše lek koji je u međuvremenu postao nedostupan.

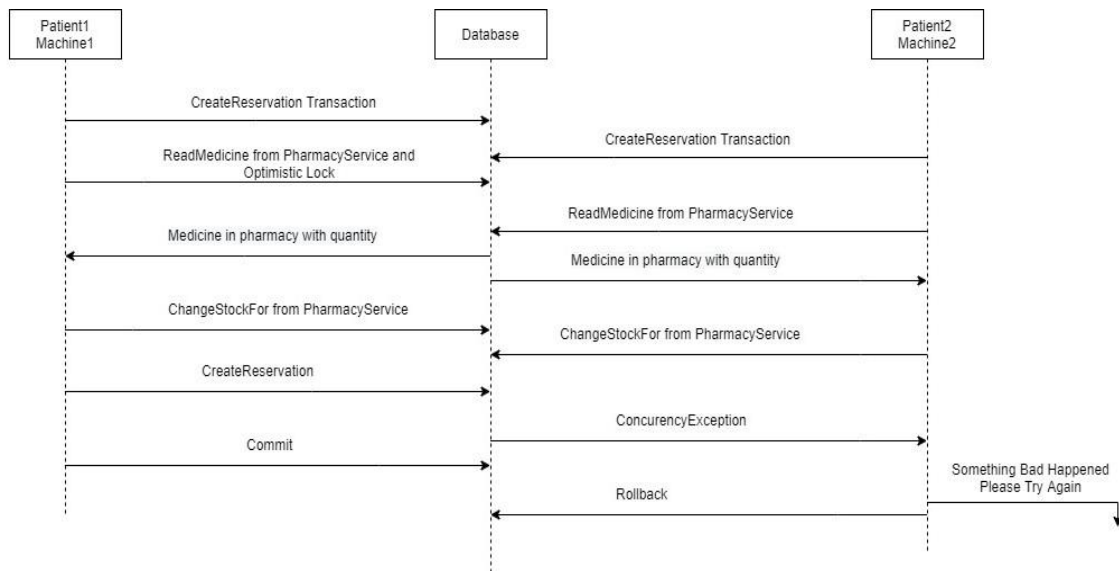
#### Kako dolazi do konfliktne situacije

Do konfliktne situacije dolazi ukoliko dva korisnika u isto vreme naručuju isti lek. Endpoint backend servera koji se gađa POST zahtevom je /reservations. Nakon uspešne autorizacije poziva se metoda CreateReservation iz ReservationService. U metodi se prvo radi validacija, odnosno proverava se da li korisnik postoji u sistemu, koliko ima negativnih poena, koji rok je izabrao za preuzimanje lekova. Zatim se učitavaju lekovi i proverava se da li se lek prodaje samo na recept i da li ga u određenoj apoteci ima dovoljno na stanju. Ukoliko provere prođu za sve lekove, onda se smanjuje njihova količina u apoteci i zapravo tu nastaje problem. Prilikom učitavanja lekova u oba zahteva može se desiti da leka ima dovoljno na stanju za pojedinačne zahteve, ali da ga nema dovoljno za oba. Zbog toga bi količina leka u apoteci postala negativna, što je naravno loše. Na sledeće dve slike prikazan je problem, a zatim i rešenje.

#### Problem



#### Solution



### Način na koji je konfliktna situacija rešena

Problem je moguće rešiti pomoću optimističkog zaključavanja. Atribut objekta PharmacyMedicine koji je potrebno nadgledati zbog konkurentnosti je Quantity i zbog toga će on imati anotaciju *ConcurrencyCheck*. Pored toga treba koristiti transakciju da bi se rezervacija lekova izvršila atomično. Transakcija počinje sa *BeginTransaction* i ukoliko nijedna druga nit ne promeni stanje leka u međuvremenu izvršiće se *Commit*. Ako se ipak desi da neka druga nit promeni vrednost količine leka u apoteci izazvaće se *DbUpdateConcurrencyException* i uraditi *Rollback* i korisniku poslati poruka da je nešto pošlo po zlu i da pokuša ponovo.

### Napomena

Konfliktne situacije: količina leka na stanju se mora ispravno ažurirati nakon rezervacije leka od strane korisnika i izdavanja leka preko eRecepta su rešene prethodnim rešenjem, jer se ista funkcija poziva u svakom od navedenih slučaja.

## Konfliktna situacija 2:

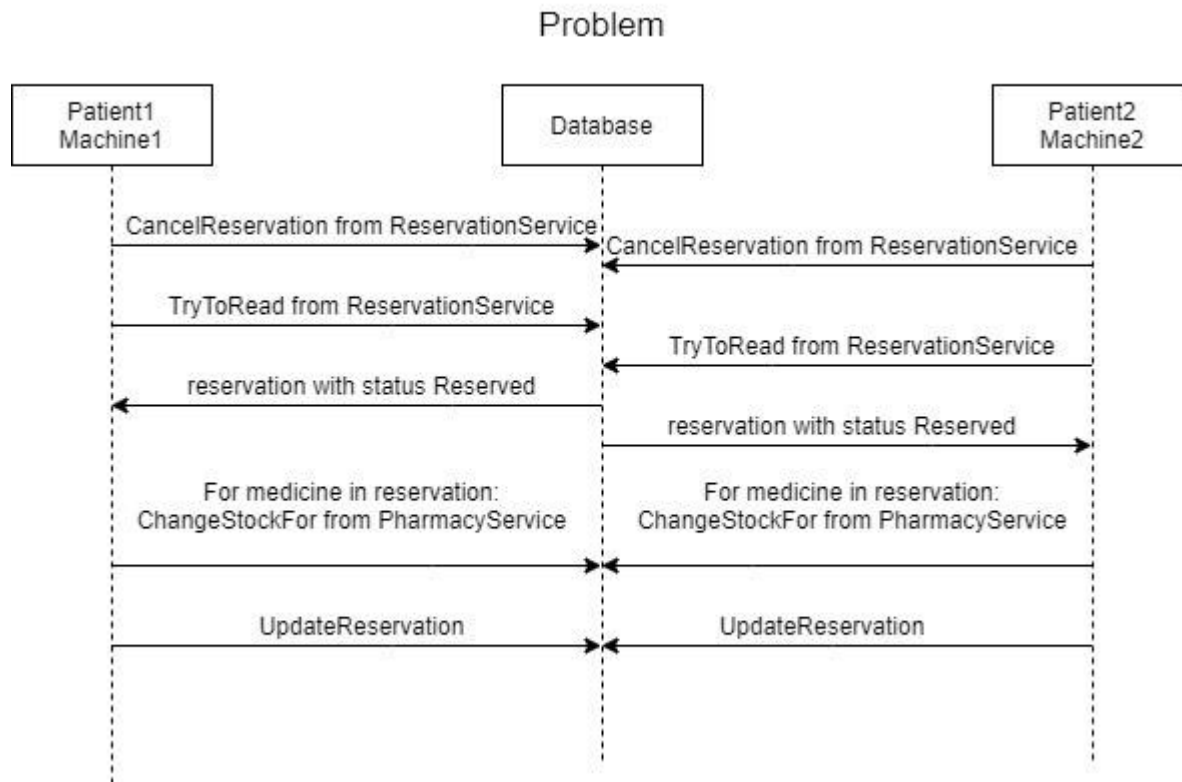
### Opis konfliktno situacije

Količina leka na stanju se mora ispravno ažurirati nakon otkazivanja rezervacije leka.

### Kako dolazi do konfliktno situacije

Do konfliktno situacije dolazi ukoliko dva korisnika u isto vreme otkazuju rezervacije lekova. Gađa se isto endpoint `/reservations/cancel/{reservationId}` Delete zahtevom. Nakon uspešne autorizacije pozva se metoda `CancelMedicineReservation` iz `ReservationService`. Nakon provere da li rezervacija postoji, da li je već otkazana i da li je ostalo više od 24h do preuzimanja, menja se količina lekova u apotekama i zbog tog

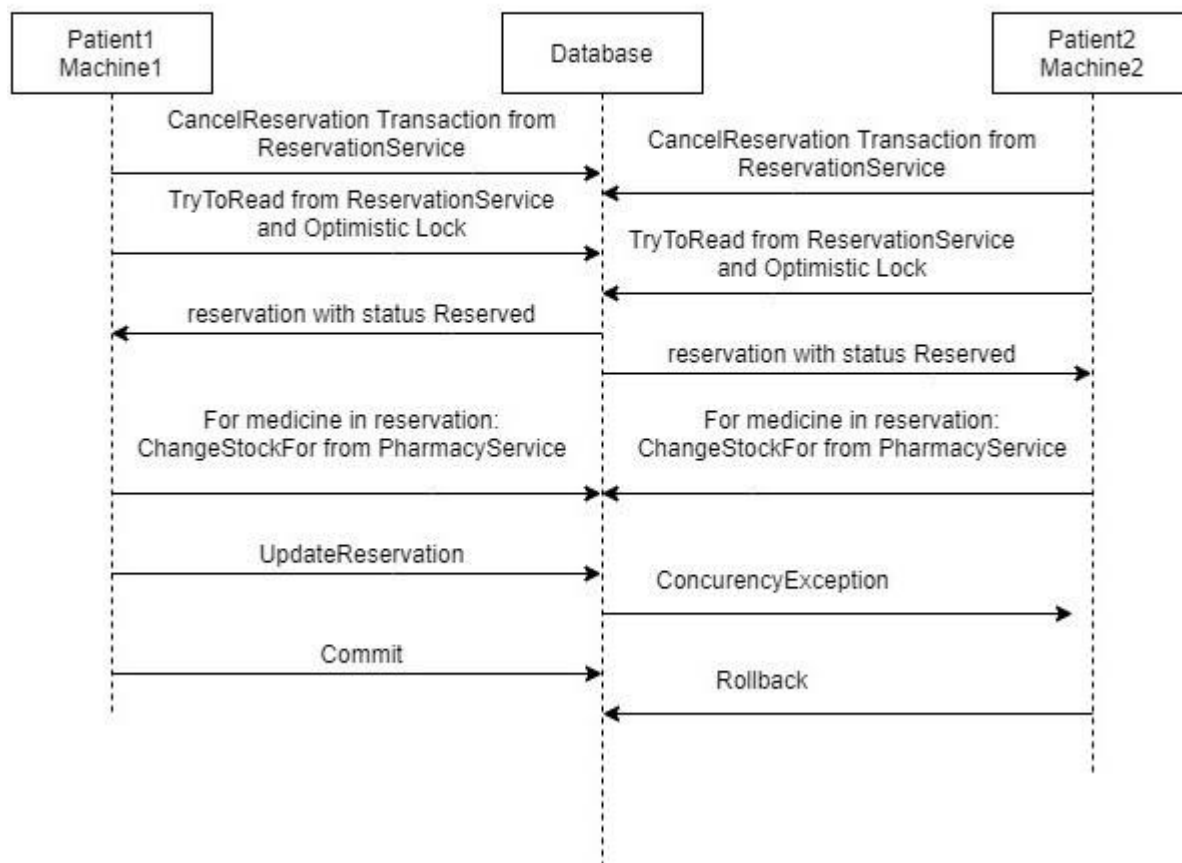
dela je ova situacija konkurentna. Ukoliko bi u isto vreme bila pozvana metoda `ChangeStockFor` sa dve različite mašine, jedna promena ne bi bila uvažena, što naravno predstavlja problem. Na sledećoj slici ilustrovan je problem.



### Način na koji je konfliktna situacija rešena

Problem je moguće rešiti pomoću optimističkog zaključavanja. Atribut objekta `PharmacyMedicine` koji je potrebno nadgledati zbog konkurentnosti je `Quantity` i zbog toga će on imati anotaciju `ConcurrencyCheck`. Pored toga treba koristiti transakciju da bi se rezervacija lekova izvršila atomično. Transakcija počinje sa `BeginTransaction` i ukoliko nijedna druga nit ne promeni stanje leka u međuvremenu izvršiće se `Commit`. Ako se ipak desi da neka druga nit promeni vrednost količine leka u apoteci izazvaće se `DbUpdateConcurrencyException` i uraditi `Rollback` i korisniku poslati poruka da je nešto pošlo po zlu i da pokuša ponovo. Rešenje je ilustrovano na sledećoj slici.

## Solution



### Konfliktna situacija 3:

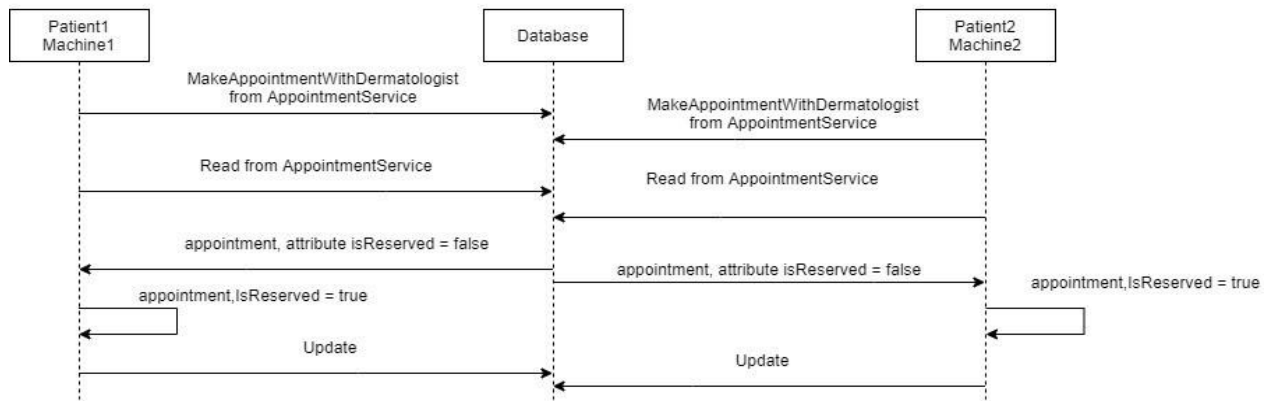
#### Opis konfliktne situacije

Pregledi koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika.

#### Kako dolazi do konfliktne situacije

Do konfliktne situacije dolazi ukoliko dva korisnika sa dve različite mešine u isto vreme žele da zakažu unapred kreiran pregled kod dermatologa. Gađa se /appointments/make-appointment endpoint POST zahtevom. Nakon uspešne autorizacije poziva se metoda MakeAppointmentWithDermatologist iz AppointmentService. U metodi se čita traženi pregled i vrši se sva potrebna validacija. Zatim se menja status pregleda na rezervisan i id korisnika na id korisnika koji šalje zahtev. Nakon toga pregled se ažurira u bazi. Upravo tu se krije problem jer će oba korisnika dobiti poruku da su rezervisali termin, a samo će jedan rezervisati zapravo. Na sledećoj slici ilustrovan je problem.

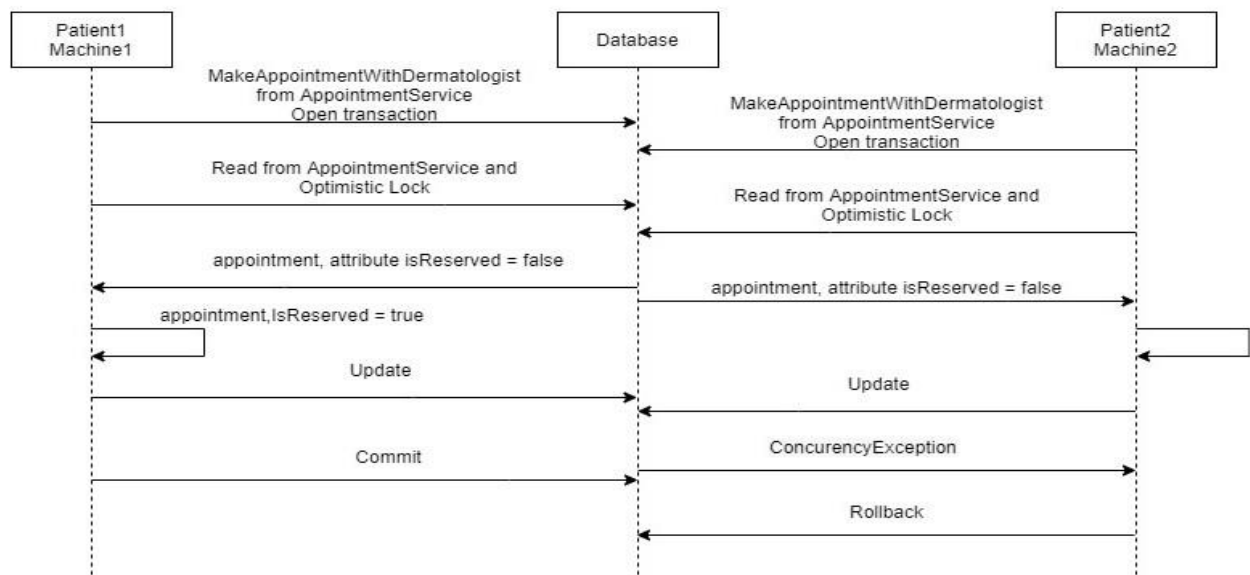
## Problem



## Način na koji je konfliktna situacija rešena

Problem je moguće rešiti pomoću optimističkog zaključavanja. Atribut objekta Appointment koji je potrebno nadgledati zbog konkurentnosti je IsReserved i zbog toga će on imati anotaciju *ConcurrencyCheck*. Atribut PatientId nije potrebno nadgledati jer nadgledanje prethodno pomenutog atributa rešava sve. Pored toga treba koristiti transakciju da bi se rezervacija pregleda izvršila atomično. Transakcija počinje sa BeginTransaction i ukoliko nijedna druga nit ne promeni status pregleda u međuvremenu izvršiće se Commit. Ako se ipak desi da neka druga nit promeni status pregleda izazvaće se *DbUpdateConcurrencyException* i uraditi Rollback i korisniku poslati poruka da je nešto pošlo po zlu i da pokuša ponovo. Rešenje je ilustrovano na sledećoj slici.

## Solution



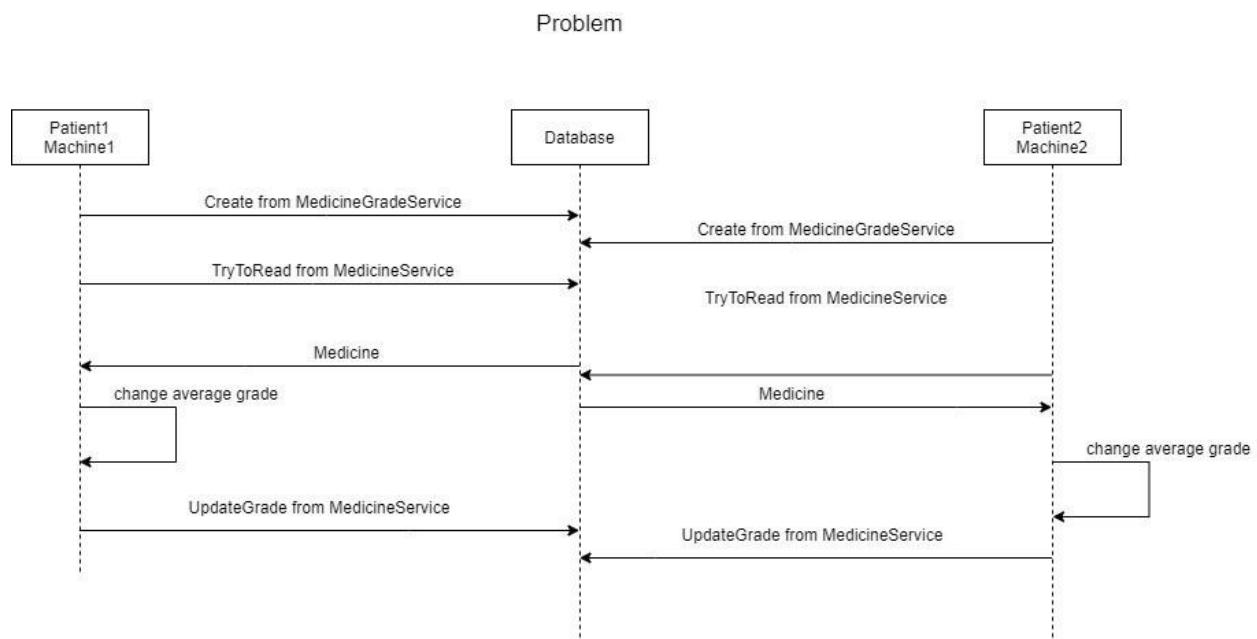
## Konfliktna situacija 4:

### Opis konfliktne situacije

Srednja ocena i broj ocena moraju se ispravno ažurirati nakon ocenjivanja lekova i menjanja ocene leka.

### Kako dolazi do konfliktne situacije

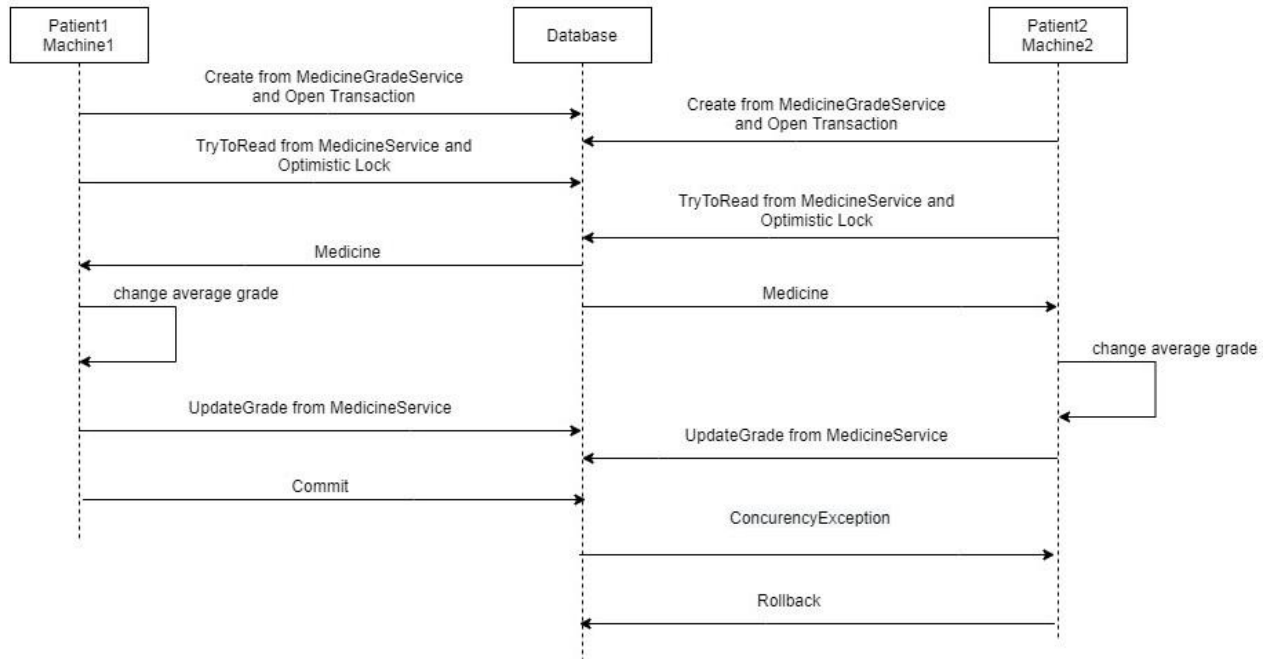
Do konfliktne situacije dolazi kada dva korisnika sa različitih mašina u isto vreme žele da ocene isti lek. Gađa se `/grades/rate-medicine` endpoint. Nakon uspešne autorizacije poziva se metoda `Create` iz `MedicineGradeService`. Prvo se učitava lek i vrše validacije. Nakon njih potrebno je izračunati novu prosečnu ocenu, povećati broj ocena leka i ažurirati to u bazi. Upravo tu se krije problem jer jedna ocena može biti pregažena, odnosno neće biti uvažena. Ilustracija problema je na sledećoj slici.



### Način na koji je konfliktna situacija rešena

Problem je moguće rešiti pomoću optimističkog zaključavanja. Atributi objekta `Medicine` koji su konkurentni su `AverageGrade` i `NumberOfGrades`. Međutim dovoljno je nadgledati samo jedan od njih i zbog toga će `AverageGrade` imati anotaciju `ConcurrencyCheck`. Pored toga treba koristiti transakciju da bi se ocenjivanje leka izvršilo atomično. Transakcija počinje sa `BeginTransaction` i ukoliko nijedna druga nit ne promeni srednju ocenu leka u međuvremenu izvršiće se `Commit`. Ako se ipak desi da neka druga nit promeni srednju ocenu leka izazvaće se `DbUpdateConcurrencyException`, uraditi `Rollback` i korisniku poslati poruka da je nešto pošlo po zlu i da pokuša ponovo. Rešenje je ilustrovano na sledećoj slici.

### Solution



### Napomena

Na isti način je u kodu rešena i konfliktna situacija menjanja ocene leku.

### Korisni linkovi:

1. [LOCK IN SHARE MODE - MariaDB Knowledge Base](#)
2. [Concurrency Management in Entity Framework Core | Learn Entity Framework Core](#)
3. [Transaction in Entity Framework 6 & Core \(entityframeworktutorial.net\)](#)