# Rešavane konfliktne situacije

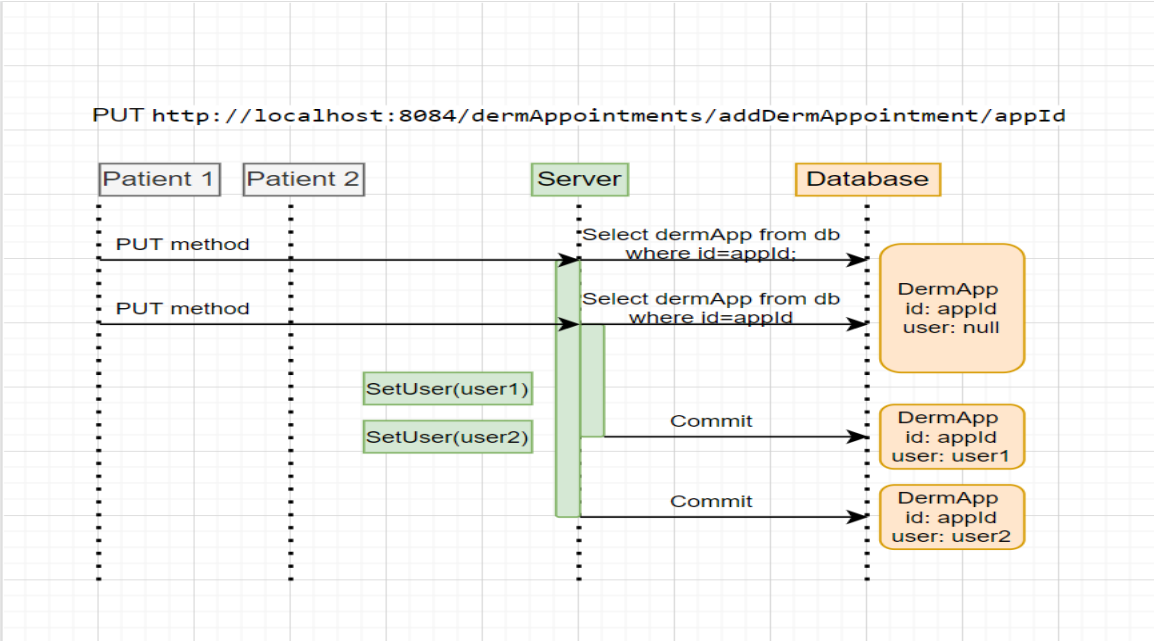
## 1. Zakazivanje pregleda kod dermatologa

### Funkcionalnost

Na stranici apoteke pacijent može videti slobodne termine pregleda kod dermatologa u toj apoteci. Jednim klikom zakazuje izabrani termin. Na email mu stiže obaveštenje o detaljima zakazanog pregleda. Pregledi kod dermatologa su unapred definisani.

### Opis konfliktne situacije

Dva korisnika ne mogu istovremeno zakazati isti termin pregleda.




### Rešenje konfliktne situacije

Za rešavanje ove konfliktne situacije korišćen je optimistički pristup zaključavanju (Optimistic locking). U svakoj transakciji kada se modifikuje objekat entiteta njegov broj verzije se uvećava za jedan. Tokom commit-a (i flush-a) porede se brojevi verzija objekata u bazi i u memoriji. Ako se ne poklapaju nastaje OptimisticLockException, što nam govori da je objekat izmenio drugi korisnik koristeći drugi Entity Manager.

U našem slučaju, proverićemo pre rezervisanja termina da li je zauzet u međuvremenu i ako jeste nastaće izuzetak.

```java
@Override
public DermAppointment addDermAppointment(User user, Long appId) {
    //nadjem pregled koji hoce da zakaze
    Optional<DermAppointment> dermApp = dermAppointmentRepository.findById(appId);

    //proverim da li ga je u medjuvremenu rezervisao drugi pacijent
    if(dermApp.get().getUser() != null){
        throw new OptimisticLockException();
    }

    //termin je zauzet kada je user setovan
    dermApp.get().setUser(user);

    return dermAppointmentRepository.save(dermApp.get());
}
```

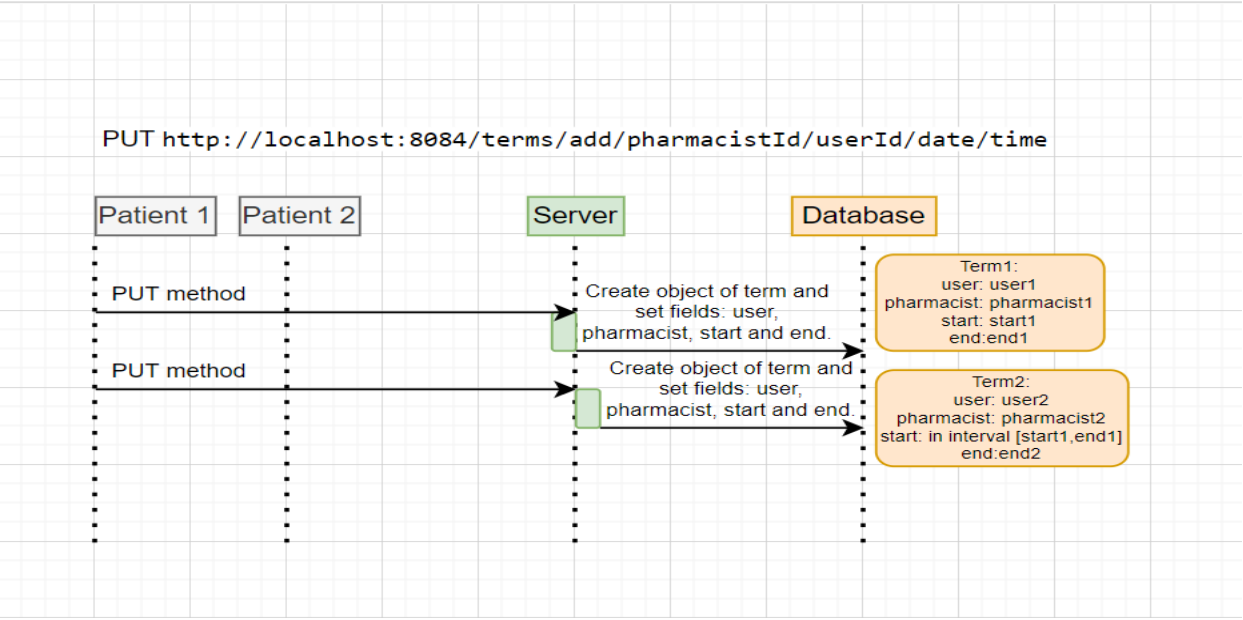
## 2. Zakazivanje pregleda kod farmaceuta

### Funkcionalnost

Na početnoj stranici ulogovani korisnik ima mogućnost da zakaže pregled kod farmaceuta. Prvo bira datum i vreme. Zatim mu se za izabrani termin prikazuju apoteke koje u tom terminu imaju slobodnog farmaceuta. Kada izabere apoteku, prikazuju mu se farmaceuti koji u izabranom terminu u izabranoj apoteci imaju slobodan termin. Pacijent zatim jednim klikom bira farmaceuta i pregled je zakazan. Na email mu stiže obaveštenje o detaljima zakazanog pregleda.

### Opis konfliktne situacije

Dva korisnika ne mogu istovremeno zakazati pregled čija vremena se poklapaju.



### Rešenje konfliktne situacije

Za rešavanje ove konfliktne situacije korišćen je optimistički pristup zaključavanju, koji je već opisan za prethodni konfliktni slučaj zakazivanja pregleda kod dermatologa.

U našem slučaju, proverićemo pre kreiranja pregleda kod farmaceuta da li se izabrani termin pregleda ne preklapa sa već postojećim terminima u bazi.

```
@Override
public Object add(Date dateValue, Long userId, Long pharmacistId) {

    Term term = new Term();

    Optional<User> user = userRepository.findById(userId);

    //izabrano vreme plus 30 min koliko traje pregled
    long ONE_MINUTE_IN_MILLIS=60000; //millisecs
    long t= dateValue.getTime();
    Date date1=new Date(t + (30 * ONE_MINUTE_IN_MILLIS));

    Pharmacist pharmacist = new Pharmacist();
    pharmacist.setId(pharmacistId);
    term.setPharmacist(pharmacist);
    term.setStart(dateValue);
    term.setEnd(date1);
    term.setUser(user.get());
```

```
    //da li je neko u medjuvremenu zakazao termin
    Boolean termNotFree = false;

    List<Term> scheduledTerms = termRepository.findAllByPharmacistId(pharmacistId);
    for(int i=0; i<scheduledTerms.size(); i++){
        if(dateValue.after(scheduledTerms.get(i).getStart()) && dateValue.before(scheduledTerms.get(i).getEnd())){
            termNotFree = true;
        }else if (dateValue.equals(scheduledTerms.get(i).getStart())){
            termNotFree = true;
        }
    }

    if(termNotFree){
        throw new OptimisticLockException();
    }

    return termRepository.save(term);
}
```

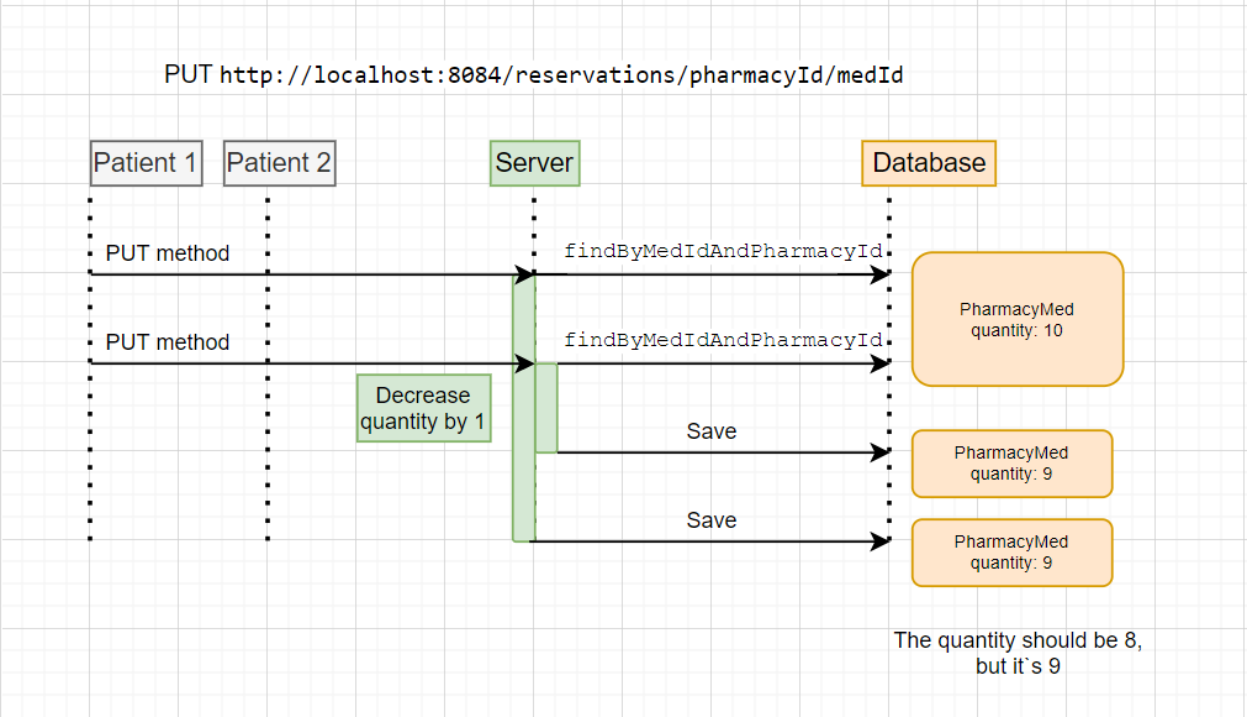
## 3. Rezeracija leka

### Funkcionalnost

Iz liste apoteka pacijent ima opciju da vidi listu lekova koje ta apoteka ima. Iz liste lekova koje apoteka ima pacijent može izabrati lek koji želi da rezerviše. Tada mu se otvara prozor u kome može izabrati datum preuzimanja leka. Kada izabere datum, lek je uspešno rezervisan.

### Opis konfliktne situacije

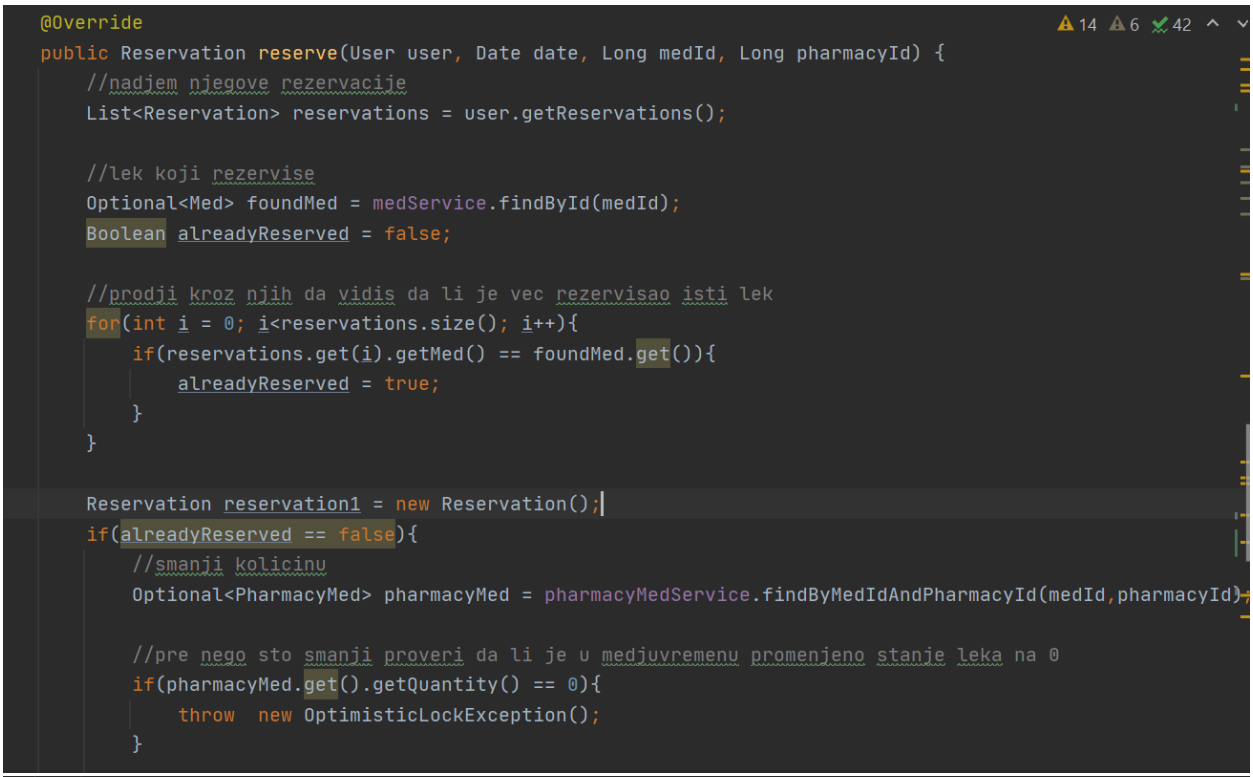
Dva korisnika ne mogu istovremeno rezervisati isti lek. Stanje leka u određenoj apoteci se mora pravilno ažurirati.




### Rešenje konfliktne situacije

Za rešavanje ove konfliktne situacije korišćen je optimistički pristup zaključavanju, koji je već opisan za prethodne konfliktne slučajeve zakazivanja pregleda kod dermatologa i farmaceuta.

U našem slučaju, proverićemo pre rezervisanja leka da li je stanje leka promenjeno u međuvremenu, odnosno da li je leka nestalo.



```
@Override                                                                    14 6 42
public Reservation reserve(User user, Date date, Long medId, Long pharmacyId) {
    //nadjem njegove rezervacije
    List<Reservation> reservations = user.getReservations();

    //lek koji rezervise
    Optional<Med> foundMed = medService.findById(medId);
    Boolean alreadyReserved = false;

    //prodji kroz njih da vidis da li je vec rezervisao isti lek
    for(int i = 0; i<reservations.size(); i++){
        if(reservations.get(i).getMed() == foundMed.get()){
            alreadyReserved = true;
        }
    }

    Reservation reservation1 = new Reservation();
    if(alreadyReserved == false){
        //smanji kolicinu
        Optional<PharmacyMed> pharmacyMed = pharmacyMedService.findByMedIdAndPharmacyId(medId,pharmacyId);

        //pre nego sto smanji proveri da li je u medjuvremenu promenjeno stanje leka na 0
        if(pharmacyMed.get().getQuantity() == 0){
            throw  new OptimisticLockException();
        }
```

```
        pharmacyMed.get().setQuantity(pharmacyMed.get().getQuantity()-1);
        pharmacyMedService.update(pharmacyMed.get());

        //napravi rezervaciju
        Optional<Pharmacy> pharmacy = pharmacyService.findById(pharmacyId);
        Reservation reservation = new Reservation();
        reservation.setMed(foundMed.get());
        reservation.setUser(user);
        reservation.setPickUpDate(date);
        reservation.setPickedUp(false);
        reservation.setPharmacy(pharmacy.get());

        //dodaj je u bazu i povratnu vrednost dodeli useru
        reservation1 = reservationRepository.save(reservation);

    }

    return reservation1;
}
```

## 3. Ocenjivanje apoteke, leka, dermatologa i farmaceuta

### Funkcionalnost

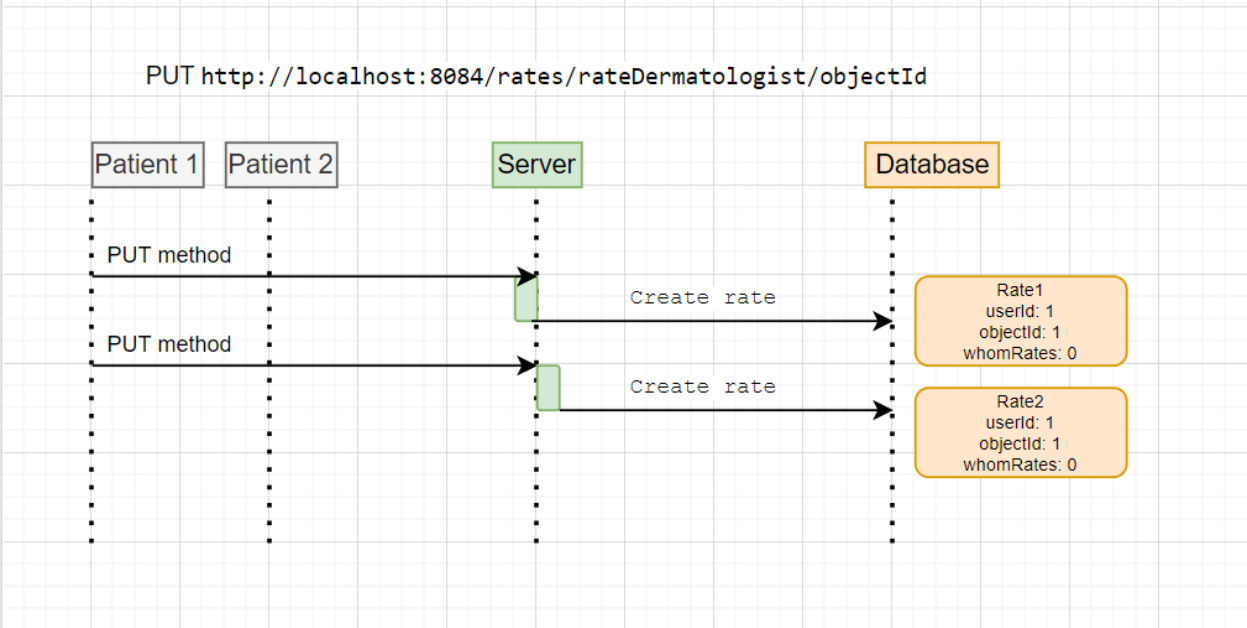
Na stranici profila, pacijent može izabrati koga želi da oceni. Ponuđene su mu 4 opcije: dermatolog, farmaceut, lek i apoteka. Kada izabere nešto od ponuđenog, prikazaće mu se dve liste. Jedna u kojoj će biti objekti koje do sada nije ocenio, a ispod toga druga koja će nuditi

opciju promene ocene već ocenjenog objekta. Za objekte koji nisu ranije ocenjeni prikazuje se prosečna ocena, a za objekte koji jesu, prikazuje se prethodno dodeljena ocena.

### Opis konfliktne situacije

Kada bi pacijent pokušao da oceni nešto od ponuđenog istovremeno sa dve pristupne tačke, na primer jednu sa jednog otvorenog prozora, a drugu sa drugog, moramo mu prikazati grešku da je ocena već napravljena, da ne bi dva puta upisali u bazu dve ocene. Pacijent samo jednom može da oceni nešto, a kasnije ocenu može menjati.

Prikazan je slučaj ocenjivanja dermatologa




### Rešenje konfliktne situacije

Za rešavanje ove konfliktne situacije korišćen je optimistički pristup zaključavanju, koji je već opisan za prethodne konfliktne slučajeve.

U našem slučaju, proverili smo pre kreiranja ocene, da li je takva ocena već napravljena.

```java
@Override
public Object rate(User user, Long objectId, Float rate, WhomRates whom) {
    //Kreiram rate
    Rate rate1 = new Rate();
    rate1.setIdOfRatedObject(objectId);
    rate1.setRate(rate);
    rate1.setUser(user);
    rate1.setWhomRates(whom);

    //provera da li je vec kreirana takva ocena
    if(!rateRepository.findByUserIdAndIdOfRatedObjectAndWhomRates(user.getId(),objectId, whom)
            .equals(Optional.empty())){
        throw new OptimisticLockException();
    }

    Object o = addRate(rate1);
```

```java
    //poziv metode koja ce ponovo da izracuna prosecnu ocenu
    //i da je sacuva

    if(whom == WhomRates.DERMATOLOGIST){
        Optional<Dermatologist> dermatologist = dermatologistService.findById(objectId);
        float avg = calculateAvgRate(objectId,WhomRates.DERMATOLOGIST);
        dermatologist.get().setStars(avg);
        dermatologistService.save(dermatologist.get());
    }else if(whom == WhomRates.PHARMACIST){
        Optional<Pharmacist> pharmacist = pharmacistService.findById(objectId);
        float avg = calculateAvgRate(objectId,WhomRates.PHARMACIST);
        pharmacist.get().setStars(avg);
        pharmacistService.save(pharmacist.get());
    }else if(whom == WhomRates.MED){
        Optional<Med> med = medService.findById(objectId);
        float avg = calculateAvgRate(objectId,WhomRates.MED);
        med.get().setStars(avg);
        medService.save(med.get());
    }else{
        Optional<Pharmacy> pharmacy = pharmacyService.findById(objectId);
        float avg = calculateAvgRate(objectId,WhomRates.PHARMACY);
        pharmacy.get().setAvgRank(avg);
        pharmacyService.save(pharmacy.get());
    }
    return o;
```

```java
public Float calculateAvgRate (Long objId, WhomRates whomRates){

    //imam id dermatologa
    //nadjem ga u bazi
    //prodjem kroz sve njegove rate-ove
    //sve saberem i podelim sa brojem ocena
    List<Rate> rates = rateRepository.findAllByIdOfRatedObjectAndWhomRates(objId,whomRates);
    int howManyRates = 0;
    float sum = 0;

    for (int i=0; i<rates.size(); i++){

        sum += rates.get(i).getRate();
        howManyRates += 1;
        System.out.println(sum);
        System.out.println(howManyRates);
    }

    return sum/howManyRates;
}
```