Transakcije – Student4 – Vladislav Maksimović

1. Izdavanje leka lekova preko eRecepta

*Korisniku je potrebno omogućiti upload QR koda. QR kod sadrži **listu lekova** koje je potrebno izdati pacijentu. Nakon uspešnog upload-a i procesiranja QR koda, korisniku se prikazuje lista svih apoteka koje na stanju imaju sve lekove koji su potrebni. Korisnik bira apoteku u kojoj želi da kupi sve lekove. Nakon odabira apoteke, korisniku se šalje mail o potvrdi izdavanja leka preko eRecepta i ažurira se stanje svih izdatih lekova u odabranoj apoteci.*

Prilikom izdavanja eRecepta se izdaju ili svi ili ni jedan lek i stanje leka u apoteci se ažurira.

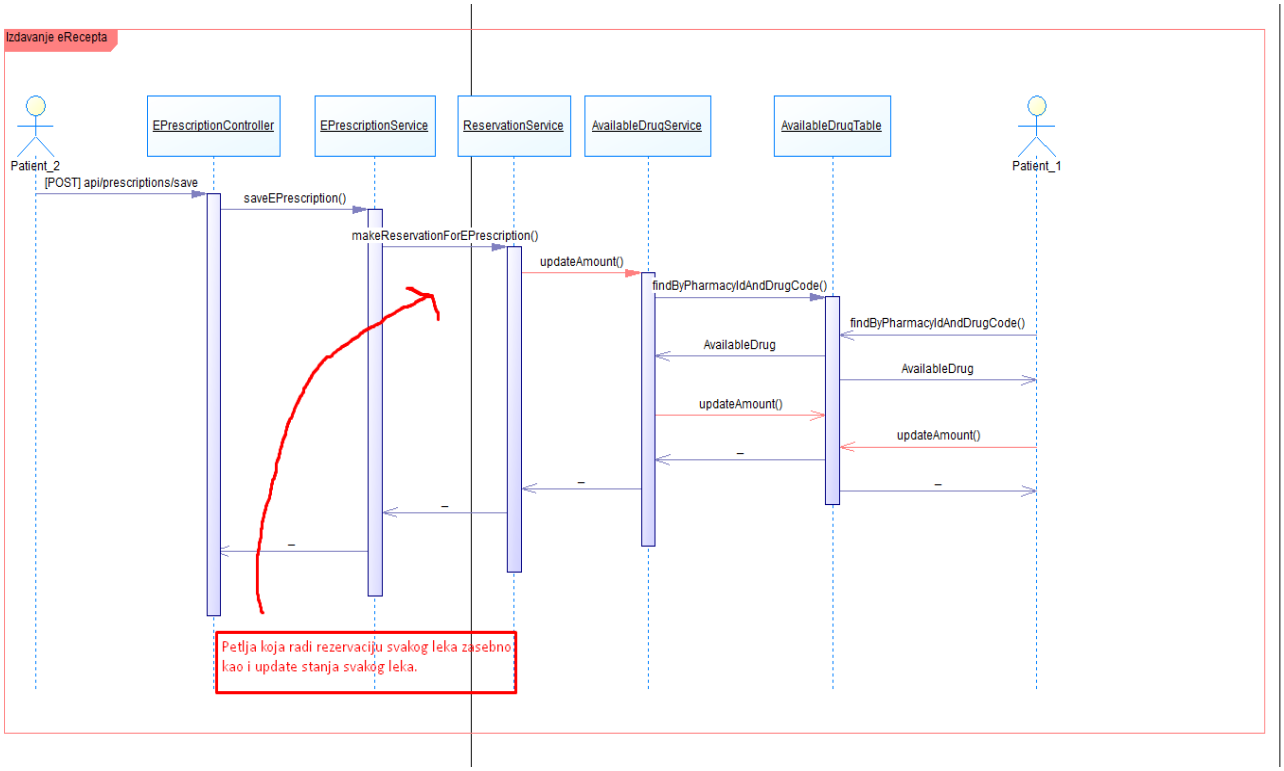
Opis konfliktne situacija koja je rešavana.

Problem koji ovde posmtaramo (slika 1.) je trenutak kada treba da se kreira rezervacija(u našem sistemu rezervacija predstavlja sinonim za rezervisanje leka) kao i ažurira stanje leka za svaki lek koji je prepisan preko eRecepta.

Ono što je konfliktna situacija je to da ako recimo od 10 prepisanih lekova pacijent X uspe da rezerviše i ažurira stanje 9 lekova, a u tom trenutku ***stigne zahtev pacijenta Y*** koji pokušava da rezerviše lekove iz svog eRecepta. Tada može da nastane konfliktna situacija ako taj pacijent Y uzme poslednje zalihe 10-og leka pacijenta X.

(Dijagram se jednako preslikava i na desnu stranu, odnosno pozivaju se iste stvari, samo trenutci u kojima se to dešava su drugačiji, a radi preglednosti i lakšeg razumevanja, redukovao sam desnu stranu tako da samo predstavljam zadnji korak koji čini pacijent 2(Y).

Crtež tokova zahteva



Slika 1. Prva rešavana konfliktna situacija

Opis rešenja

Dati problem rešavamo korišćenjem optimističkog zaključavanja dodavanjem atributa verison sa anotacijom @Version u klasi EPrescription, Reservation.

```
@Entity
public class EPrescription {
    @Version
    @Column(nullable = false)
    private Long version;
```

```
@Entity
public class Reservation {
    @Version
    @Column(nullable = false)
    private Long version;
```

```
@Override
@Transactional
public EPrescription savePrescription(EPrescriptionCreateDTO prescriptionDTO) {
```

```java
@Override
@Transactional
public void makeReservationForEPrescription(EPrescription ePrescription, Long pharmacyIdWhereIsMadeReservation) {
```

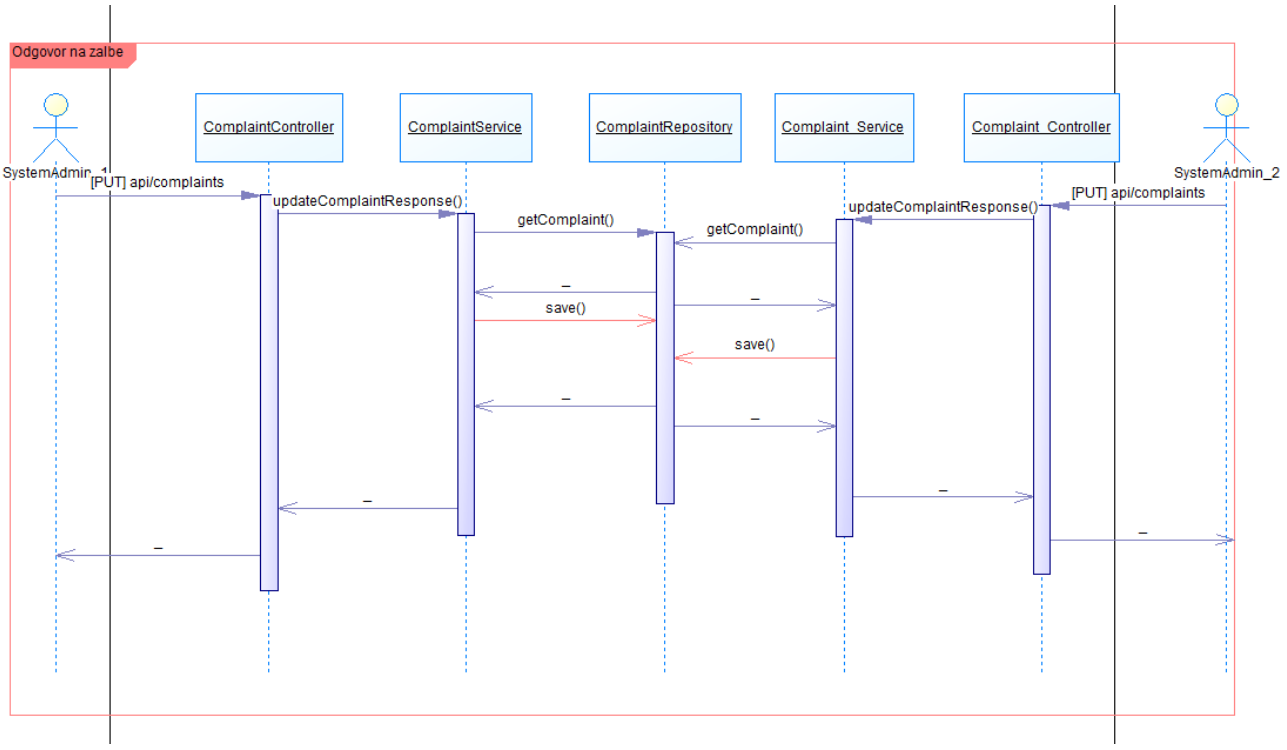
```java
@Override
@Transactional
public AvailableDrug updateAmount(Long pharmacyId, String drugCode, Integer amount) {
```

```java
@Entity
public class AvailableDrug {
    @Version
    @Column(nullable = false)
    private Long version;
```

2. Postupak odgovara na žalbu

Administrator sistema vidi sve žalbe na koje može da odgovori. Odgovor se unosi u slobodnoj formi i šalje se korisniku na mail.

Crtež tokova zahteva



Opis rešenja

Dati problem rešavamo korišćenjem optimističkog zaključavanja dodavanjem atributa verison sa anotacijom @Version u klasi Complaint.

```java
@Entity
@Check(constraints = "(consultant_id is not null) or (pharmacy_id is not null)")
public class Complaint {
    @Version
    @Column(nullable = false)
    private Long version;
```

```java
@Override
@Transactional
public Complaint updateComplaintResponse(ComplaintDTO complaintDTO) {
```

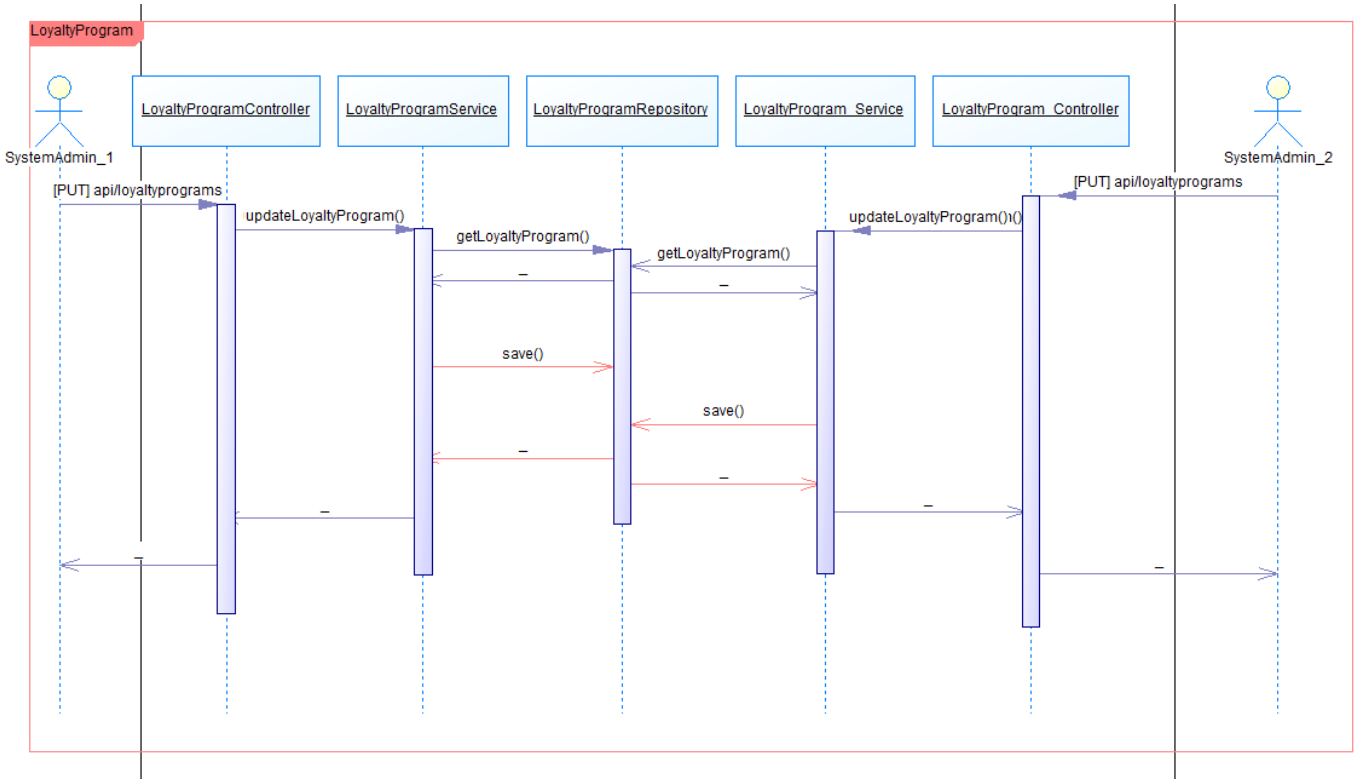
3. Postupak ažuriranja loyalty programa

Administrator sistema može da definiše loyalty program za sve korisnike koji važi na nivou čitavog informacionog sistema. Takođe, definiše i broj poena koje korisnik ostvaruje nakon svakog pregleda i savetovanja Sem broja poena, administrator sistema definiše skalu na osnovu koje se određuje kategorija korisnika (npr. Regular, Silver, Gold). Na osnovu kategorije, korisnik ostvaruje dodatni popust koji se primenjuje na svaku rezervaciju leka, kao i na svaki pregled i savetovanje.

Opis konfliktne situacija koja je rešavana.

Naš sitem poznaje jedan loyalty program. Taj loyalty program dalje može da se ažurira. A u okviru njega postoji CRUD za kategorije. Ono što predstavlja konfliktnu situaciju je trenutak kada dva različita administratora sistema pokušaju ***da izmene broj poena, trajanje i aktivnost loyalty programa*** .

Crtež tokova zahteva



Slika 3. Treća konfliktna situacija koja je rešavana

Opis rešenja

U sistemu smo ovu konfliktnu situaciju rešili primenom optimističkog zaključavanja. To smo uradili tako što smo uveli verzionisanje u tabelu koja sadrži atribute koji mogu izazvati konfliktne situacije.

```
@Entity
@Check(constraints = "(active_until is not null and is_active) or (active_until is null and not is_active)")
public class LoyaltyProgram {
    @Version
    @Column(nullable = false)
    private Long version;
```

```
@Override
@Transactional
public LoyaltyProgram updateLoyaltyProgram(LoyaltyProgramDTO loyaltyProgramDTO) {
```