

Konkurentnost - Student 1 - Dina Petrov - SW52-2018

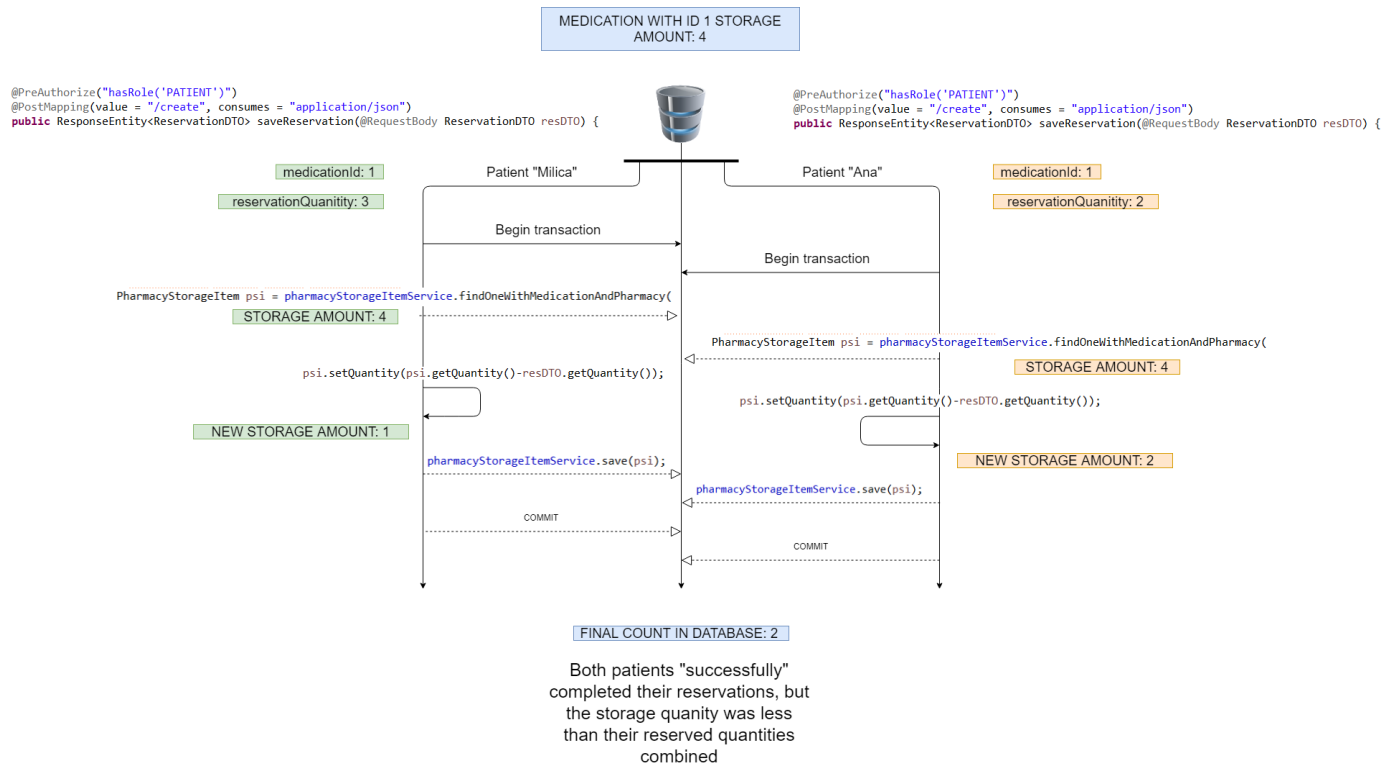
Rešavane konfliktne situacije:

1. Istovremeno rezervisanje leka koji je u međuvremenu postao nedostupan
2. Ispravno ažuriranje količine leka na stanju
 - a. Pri rezervaciji
 - b. Pri otkazivanju rezervacije
 - c. Pri izdavanju leka preko eRecepta
 - d. Pri prihvatanja ponude narudžbenice
3. DODATNA SITUACIJA: Istovremeno ažuriranje ličnih informacija o pacijentu i dodela/reset penala sistemskog sata

Rešenja:

1. Istovremeno rezervisanje leka koji je u međuvremenu postao nedostupan

Scenario problema: Dva pacijenta su prijavljena na sistem u isto vreme. Oba pacijenta žele da rezervišu "Panadol" u apoteci "Janković" gde ga ima 4 na stanju, što pacijenti i vide na sajtu. Jedan pacijent želi 3 komada, a drugi 2. Ako pokušaju da ga rezervišu u isto vreme, dolazi do konflikta i stanje leka ulazi u nekonzistentno stanje - oba pacijenta uspevaju da rezervišu zajedno 5 komada, iako na stanju ima samo 4.

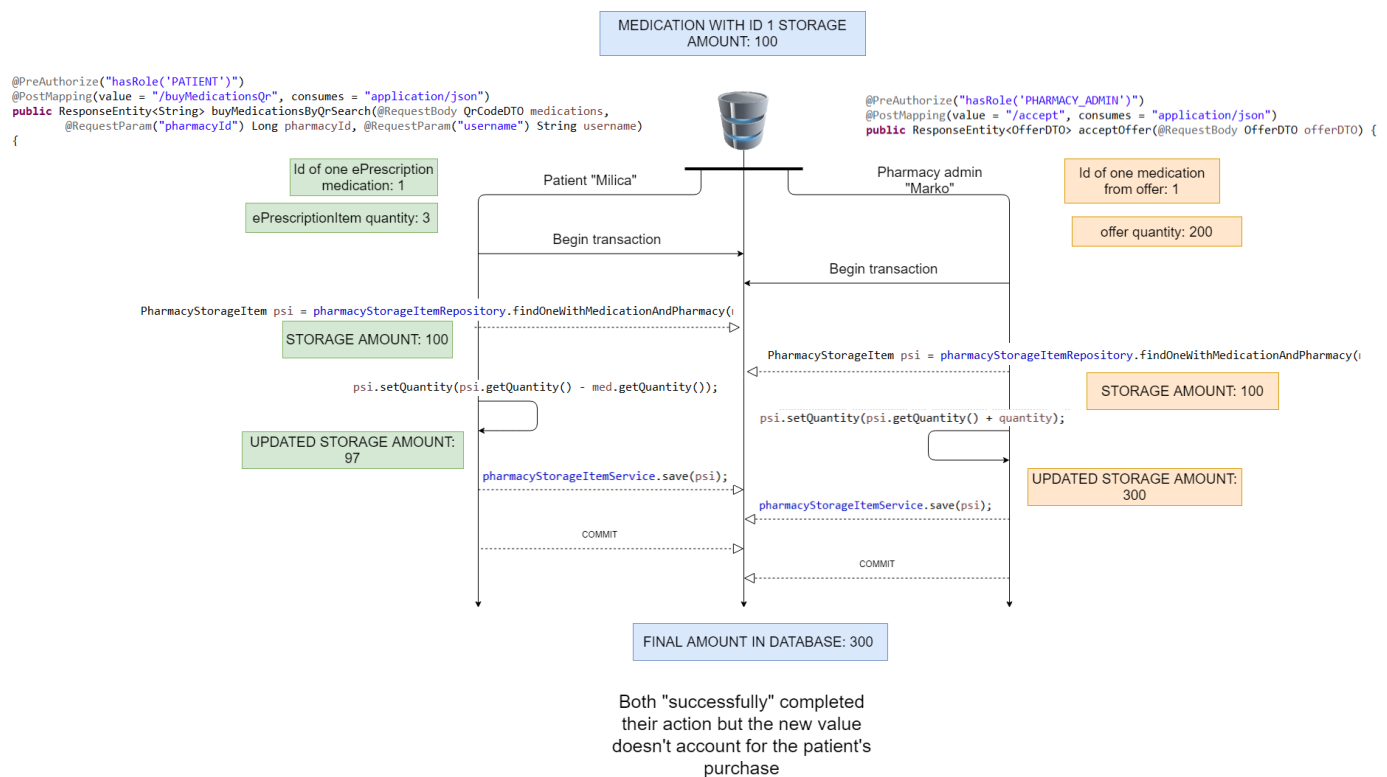


Rešenje problema: Objedinjeno sa rešenjem problema tačke 2.

2. Ispravno ažuriranje količine leka na stanju

Scenario problema: Mnoge funkcionalnosti sistema rade sa brojčanim stanjem lekova po apotekama. Neke od njih su: rezervacija leka (stanje se smanjuje za rezervisanu količinu), otkazivanje rezervacije leka (stanje se povećava za rezervisanu količinu), izdavanje leka preko eRecepta (stanje se smanjuje) i prihvatanje ponuda narudžbenica (stanje se povećava).

Moguća situacija: Pacijent preko QR čitača dobija lekove sa eRecepta iz “Benu” apoteke, pri čemu se količina leka iz eRecepta smanjuje. U isto vreme, administrator prihvata ponudu narudžbenice čime se takođe stanje tog leka u “Benu” apoteci povećava. Kao rezultat imamo prividno uspešno izvršene obe akcije, ali sa neispravnom količinom u bazi.



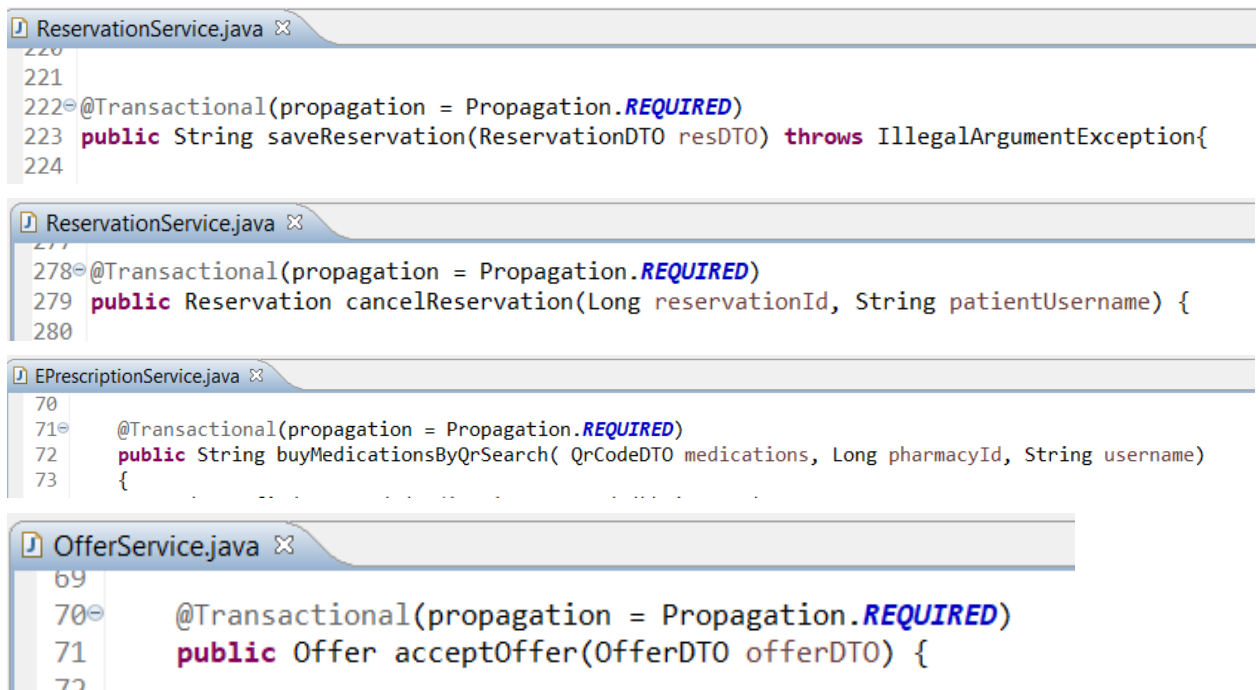
Rešenje problema: Kako je u ovoj konfliktnoj situaciji, a i u prethodnoj tački 1. u centru moguće kolizije upravo raspoloživa količina leka, rešenje pokriva obe tačke. Pošto je od najveće važnosti održavati i prikazivati ispravnu količinu leka na stanju, predloženo rešenje je pesimističko zaključavanje. Ovaj pristup je odbran upravo zato što su najčešće akcije u sistemu rad sa lekovima i njihovim količinama, te se pretpostavlja da bi potencijalno najviše kolizija i dolazilo u tim situacijama. Dovoljno je da istovremeno dva korisnika vrše rezervaciju leka (što jeste jedna od najbitnijih funkcionalnosti sistema

apoteka) da potencijalno dođe do kolizija, a samim tim i nekonzistentnog stanja količine leka u bazi.

Prilikom bilo kakvo rada sa količinama nekog leka u skladištu neke apoteke, korišćena je *findOneWithMedicationAndPharmacy* metoda klase *PharmacyStorageItemRepository*. Dodavanjem anotacije **@Lock(LockModeType.PESSIMISTIC_WRITE)** nad ovom metodom, zaključavamo objekat klase *PharmacyStorageItem* za čitanje i pisanje. Dodatno je i podešena vrednost timeout-a na 0, kako bismo dobili *PessimisticLockingFailureException* u slučaju da ova torka nije dostupna, te se ispisuje adekvatna greška.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
@Query("select s from PharmacyStorageItem s where s.medication.id =?1 and s.pharmacy.id =?2")
public PharmacyStorageItem findOneWithMedicationAndPharmacy(Long medicationId, Long pharmacyId);
```

Takođe je dodata **@Transactional** anotacija nad onim metodama servisa koje rade sa stanjem količina lekova.



```
ReservationService.java
220
221
222 @Transactional(propagation = Propagation.REQUIRED)
223 public String saveReservation(ReservationDTO resDTO) throws IllegalArgumentException{
224

ReservationService.java
277
278 @Transactional(propagation = Propagation.REQUIRED)
279 public Reservation cancelReservation(Long reservationId, String patientUsername) {
280

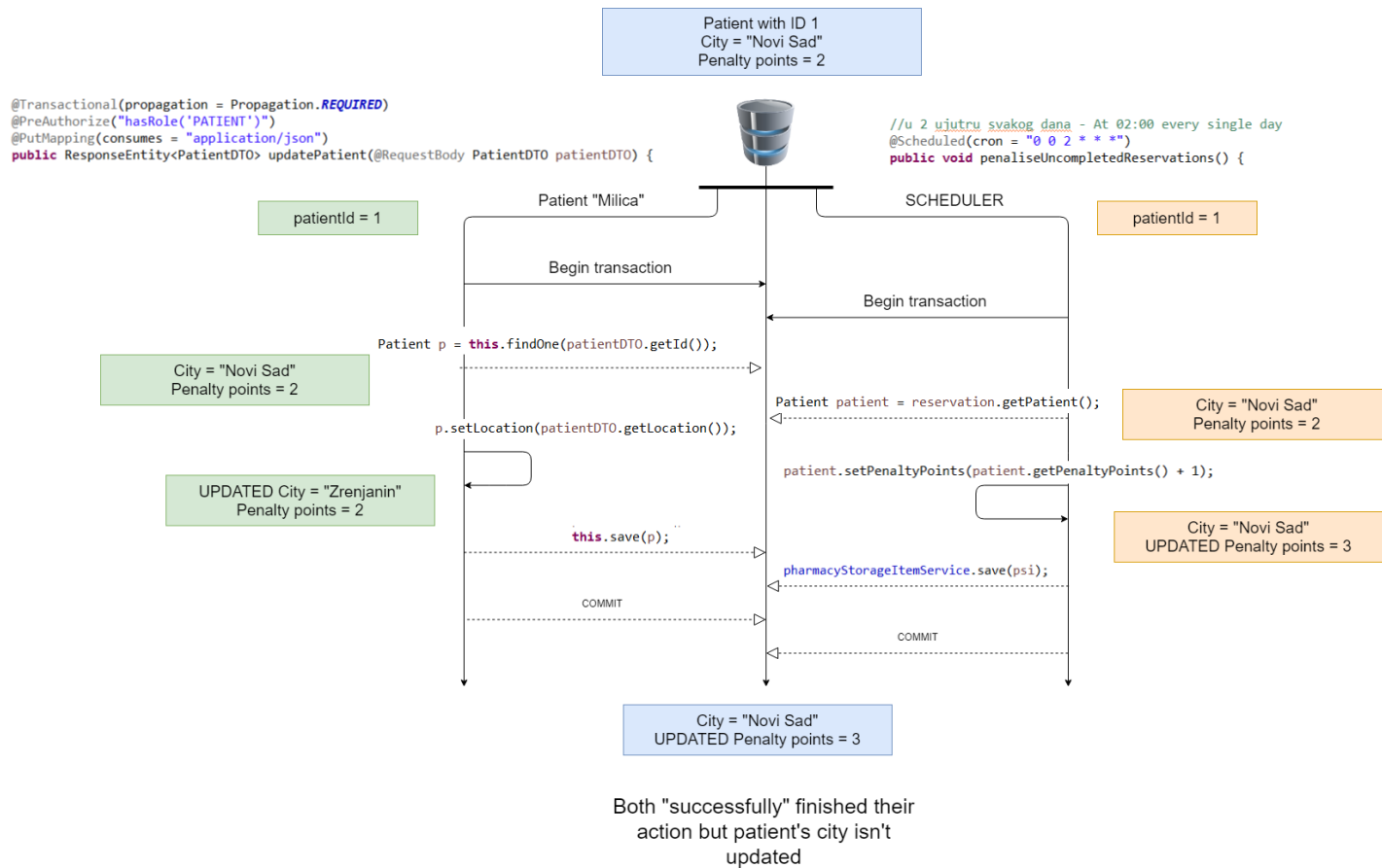
EPrescriptionService.java
70
71 @Transactional(propagation = Propagation.REQUIRED)
72 public String buyMedicationsByQrSearch( QrCodeDTO medications, Long pharmacyId, String username)
73 {

OfferService.java
69
70 @Transactional(propagation = Propagation.REQUIRED)
71 public Offer acceptOffer(OfferDTO offerDTO) {
72
```

3. DODATNA SITUACIJA: Istovremeno ažuriranje ličnih informacija o pacijentu i dodela/reset penala sistemskog sata

Scenario problema: Pacijent je prijavljen na sistem u 2 ujutru i želi da izvrši izmenu svojih ličnih podataka. S druge strane, sistemski sat vrši proveru i potencijalnu dodelu penala svakih 24 sata, u 2 ujutru, a svakog prvog u mesecu u isto vreme vrši brisanje penala. Tom prilikom, ako se pokaže da je istekla rezervacija pacijenta a da on lek nije

preuzeo ili izvršio otkazivanje rezervacije, sistem mu dodeljuje penal. Potencijalnim istovremenim ažuriranjem ličnih informacija u vreme ove automatske izmene penala, može doći do nekonzistentnih podataka pacijenta.



Rešenje problema: Verovatnoća odigravanja ove situacije nije velika, ali je moguća. Otuda je odabran pristup optimističkog zaključavanja. Verzionisan je pacijent, odnosno klasa User koju klasa Patient nasleđuje, tako da će se pri pokušaju ažuriranja pacijenta gde nije najnovija verzija desiti *ObjectOptimisticLockingFailureException*. Ovo se radi dodavanjem **@Version** anotacije nad novim atributom *version*.

```
User.java x
83
84 @Version
85 @Column(columnDefinition = "integer DEFAULT 0", nullable = false)
86 private Long version;
87
```

Zatim su odgovarajuće servisne metode anotirane **@Transactional** anotacijom.

```
PatientService.java ✕
131
132 @Transactional(propagation = Propagation.REQUIRED)
133 public Patient updatePatient(PatientDTO patientDTO) {
134
```

```
PatientService.java ✕
109
110 @Transactional(propagation = Propagation.REQUIRED)
111 @EventListener(ApplicationReadyEvent.class)
112 public void resetPenals() {
```

```
ReservationService.java ✕
119
120 @Transactional(propagation = Propagation.REQUIRED)
121 @EventListener(ApplicationReadyEvent.class)
122 public void penaliseUncompletedReservations() {
123
```