Konkurentni pristup bazi

Nemanja Pualić RA162/2017

Konfliktna situacija 1.

Prilikom provere dostupnosti leka u toku izvršavanja pregleda, dva različita dermatologa na dva različita pregleda u istoj apoteci mogu u istom trenutku da pokušaju da prepišu lek korisniku. Prilikom prepisivanja leka u našem sistemu se ta akcija vodi kao da je lek takođe i izdat korisniku. Prilikom izdavanja leka potrebno je u bazi smanjiti amount entiteta AvailableDrug, prilikom update-a količine dostupnog leka od strane dva dermatologa istovremeno dolazi do konfliktne situacije. Dati problem rešavamo korišćenjem optimističkog zaključavanja dodavanjem atributa verison sa anotacijom @Version u klasi AvailableDrug entiteta.

```
@Entity
public class AvailableDrug {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Version
    @Column(nullable = false)
    private Long version;
```

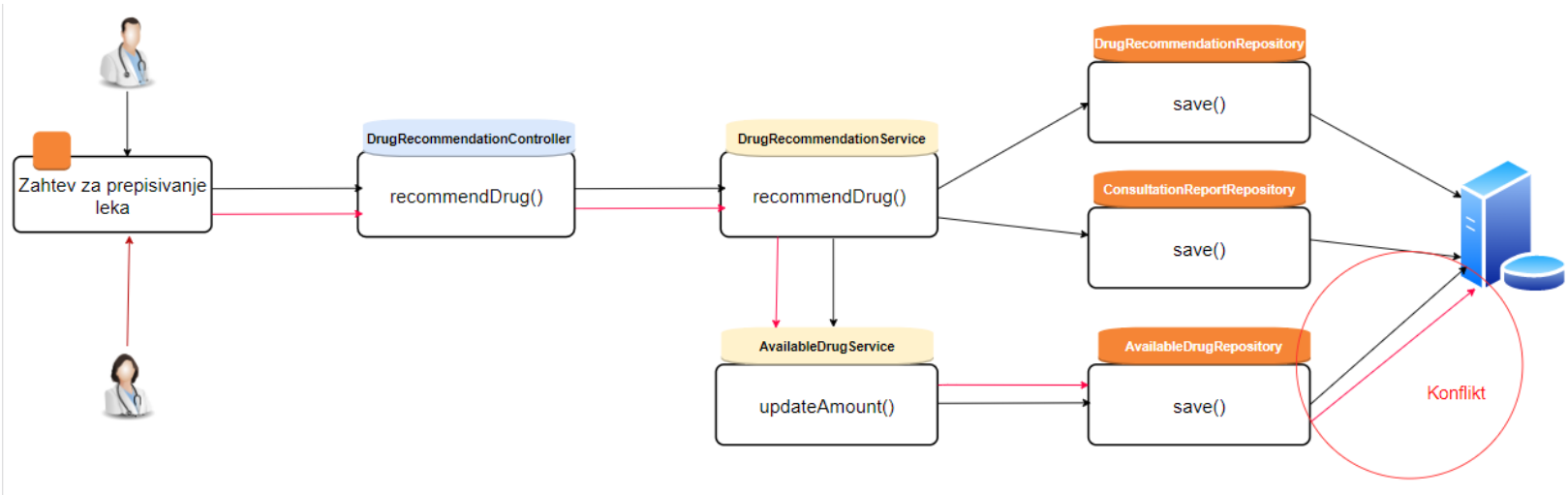
Metoda *updateAmount()* klase *AvailableDrugService()* kojom se izvršava ažuriranje stanja leka označena je kao transakciona, kao i metoda *recomendDrug()* u klasi *DrugRecommendationService().*

```
@Override
@Transactional
public AvailableDrug updateAmount(Long pharmacy
    AvailableDrug availableDrug = availableDrug
            .orElse( other: null);
    if (availableDrug == null)
        throw new PSBadRequestException("There
```

```
@Override
@Transactional
public DrugRecommendationDTO recommendDrug(DrugRec
    Consultation consultation = consultationReposi

    ConsultationReport consultationReport = new Co
```

Ovaj pristup nam omogućava da se prilikom transkacije atribut version poveca za jedan. Sledeća transakcija kada pokuša da sačuva svoje promene, prvo će se uporediti vrednost version u bazi sa vrednošću toga atributa u transakciji i doći će do izuzetka. Transakcija se neće izvršiti. Izabrani nivo izolacije je Read Commited.



Konfliktna situacija 2.

Prilikom zakazivanja predefinisanog termina dermatolog ili farmaceut može da dođe u konfliktnu situaciju sa pacijentom koji takođe zakazuje pregled, ovde potencijalno dolazi do neželjene situacije da jedan dermatolog ili farmaceut istovremeno bude na više pregleda. Dati problem takođe rešavamo korišćenjem optimističkog zaključavanja dodavanjem atributa verison sa anotacijom @Version u klasi entiteta Consultation.

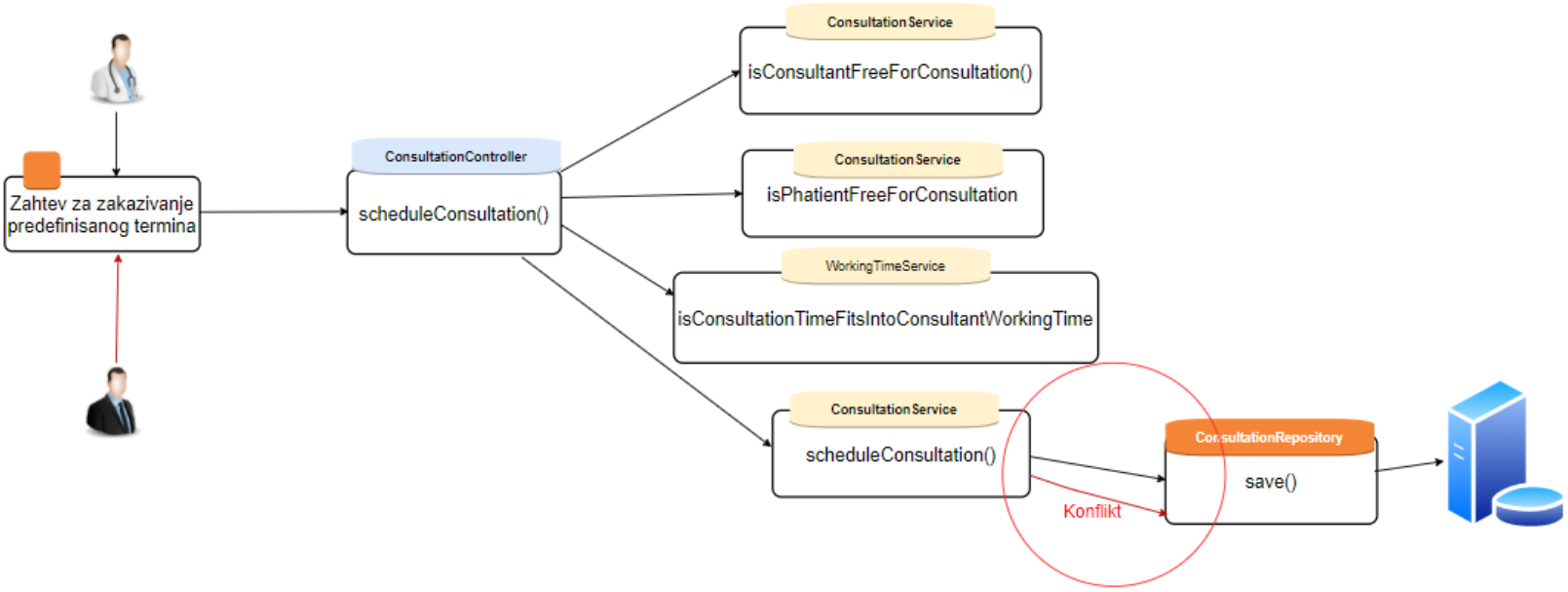
```
@Entity
public class Consultation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Version
    @Column(nullable = false)
    private Long version;
```

Izabrani nivo izolacije je Read Commited. Metoda *scheduleConsultation()* klase *ConsultationService()* onnačena je kao transakciona.

```
@Override
@Transactional
public Consultation scheduleConsultation(ScheduleExaminationDTO consultation) {

    DateRange consultationDataRange = new DateRange();
```



Konfliktna situacija 3.

Prilikom izdavanja rezervisanog leka postoji mogućnost konflikne situacije i zloupotrebe sistema, naime postoji mogućnost da dva korisnika istovremeno dođu sa istim jedinstvenim kodom rezervacije u dve razlicite filijale iste apoteke, tj. na dva različita šaltera i da pre nego što prvi farmaceut izda lek prvom korisniku drugi farmaceut proveri da li je rezervacija sa istim jedinstvenim kodom validna sistem mu vrati informaciju da je validna, u tom trenutku će dva korisnika podići dva leka na istu rezervaciju.
Ovu konfliktnu situaciju rešavamo rešavamo korišćenjem optimističkog zaključavanja dodavanjem atributa verison sa anotacijom @Version u klasi entiteta Reservation

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

Metoda *dispensiingMedicine()* klase *ReservationService()* onznačena je kao transakciona. Izabrani nivo izolacije je Read Commited.

```java
@Override
@Transactional
public ReservationDTO dispensingMedicine(ReservationDTO reservationDTO) {
    Reservation reservation = reservationRepository.getOne(reservationDTO.getReservationID());
```