Transakcije

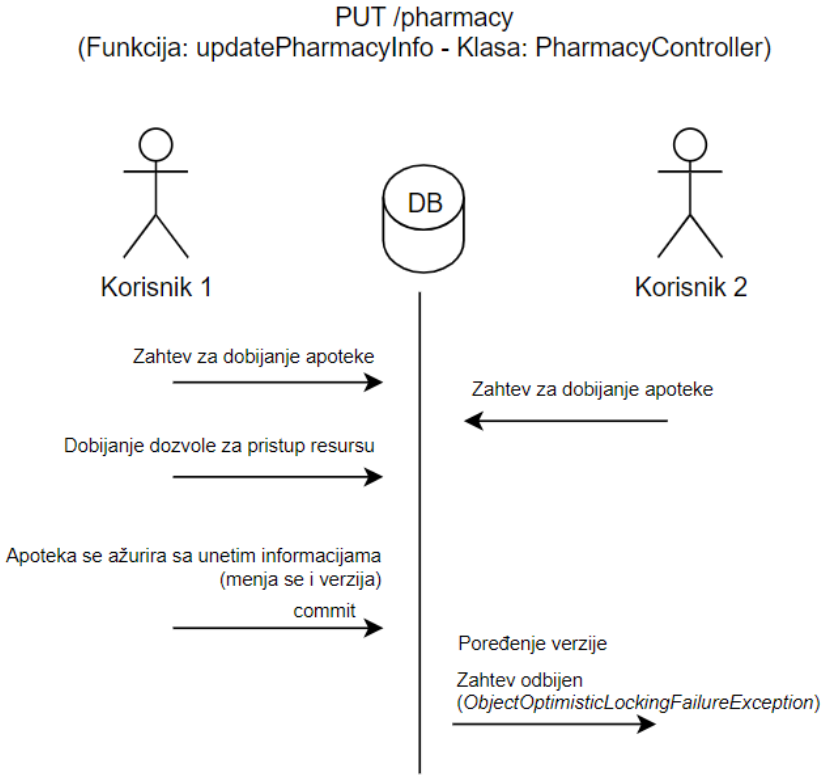
Student 2 – Vasilije Bursać RA148/2017

Situacija 1: ***Izmena profila apoteke***

“Izmena profila apoteke od strane više administratora apoteke istovremeno”

Administratori apoteke imaju mogućnost izmene informacija, tj. profila apoteke.
Ukoliko dva administratora pokušaju istovremeno da izmene informacije apoteke, postoji mogućnost da izmene koje je jedan uneo, prepišu izmene onog drugog.

Grafički prikaz:



Rešenje:

- Korišćeno optimističko zaključavanje, jer je verovatnoća za ovakav konflikt veoma mala, a ne želimo da drugi korisnici čekaju na dobijanje resursa ukoliko je gotovo sigurno da će on buti zaključan. Takođe, optimističko zaključavanje daje bolje performanse i štedljivije je po resurse jer nema zaključavanja pristupa i čekanja.
  U klasi *Pharmacy* dodat je atribut *Long version* koji je anotiran sa *@Version* što omogućava Hibernate-u da pri svakoj izmeni torke inkrementira njegovu verziju u slučaju da je ona bila ista kao i na početku konverzacije. U slučaju da se verzije ne poklapaju dobija se *ObjectOptimisticLockingFailureException* izuzetak i transakcija se prekida.

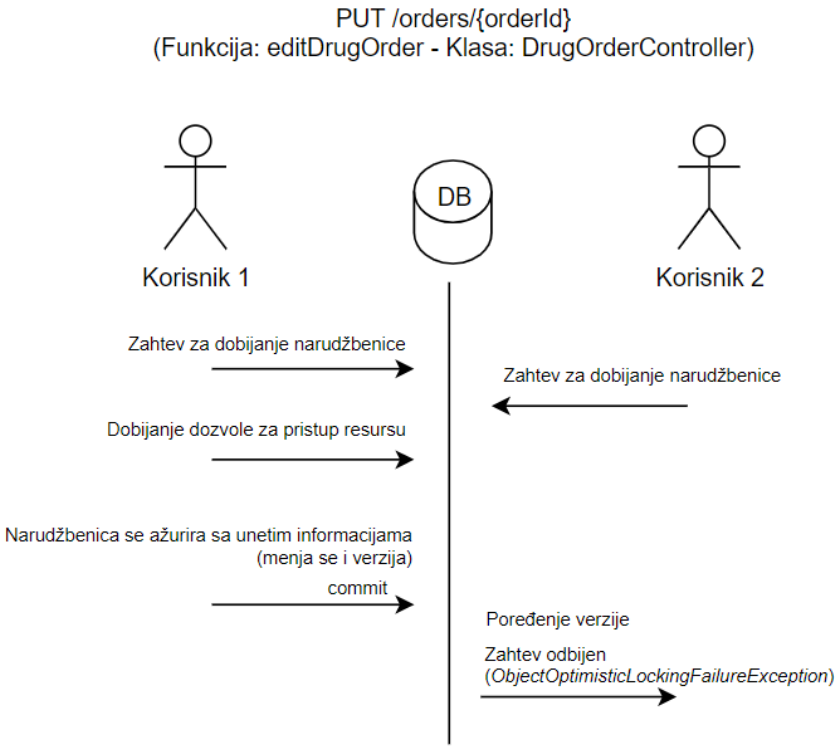
```
@Entity
public class Pharmacy {
    @Id
    @SequenceGenerator(name = "mySeqGen2", sequenceName = "mySeq2",initialValue = 5,a
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "mySeqGen2")
    private long id;

    @Version
    @Column(nullable = false, columnDefinition = "int default 1")
    private Long version;
```

Situacija 2: ***Izmena narudžbenice apoteke***

“Izmena narudžbenice apoteke od strane više administratora apoteke istovremeno”

Kao i u prošlom primeru, postoji mogućnost da više administratora apoteke radi izmenu narudžbenice istovremeno, što može dovesti stanje entiteta u nekonzistentno i neželjeno stanje.

Grafički prikaz:



Rešenje:

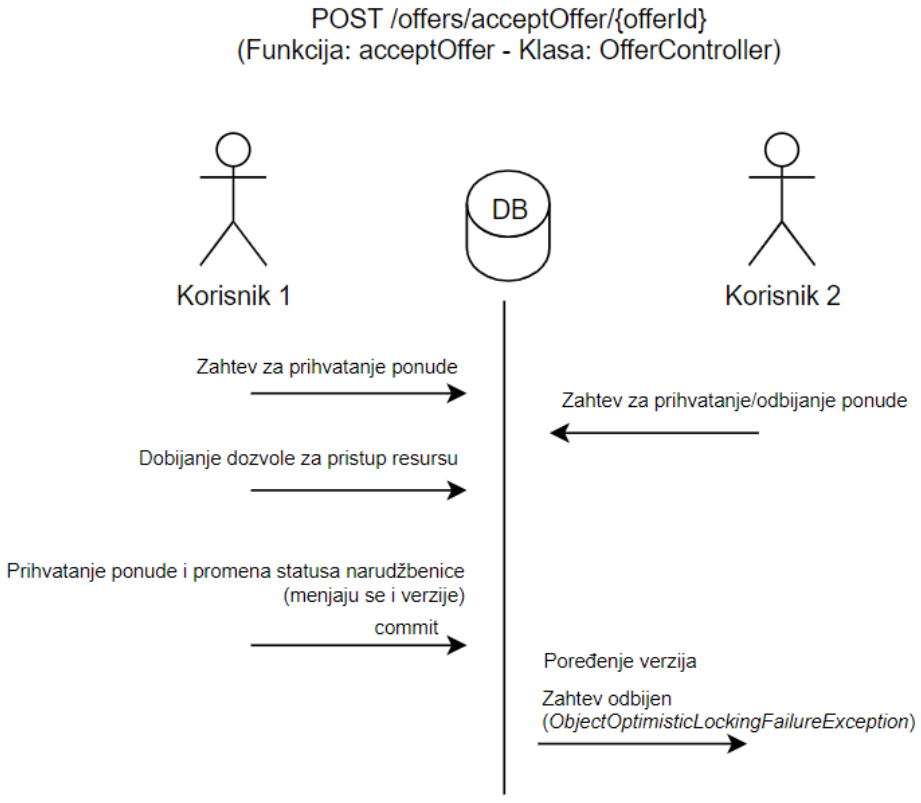
- Kao rešenje, iskorišćen je princip optimističkog zaključavanja. Budući da radimo nad celim objektom i da ceo objekat šaljemo repozitorijumu, provera verzije će se automatski obaviti, te će transakcija biti izvršena ili ćemo dobiti odgovarajući izuzetak. Polje *version* je dodato u klasu *DrugsOrder*.

```java
@Entity
@Table(name="orders")
public class DrugsOrder {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Version
    @Column(nullable = false, columnDefinition = "int default 1")
    private Long version;
```

Situacija 3: ***Prihvatanje/odbijanje ponuda za narudžbenicu apoteke***
"Prihvatanje/odbijanje ponuda za narudžbenicu od strane više administratora apoteke istovremeno"

Do konfliktne situacije dolazi ukoliko više administratora apoteke prihvati različite ponude za jednu narudžbenicu u isto vreme, ali i kada podaci na frontu jednog od admina nisu u skladu sa stanjem u sistemu, pa se može desiti da on pokuša da prihvati neku ponudu koja je već prethodno odbijena.

Grafički prikaz:



Rešenje:

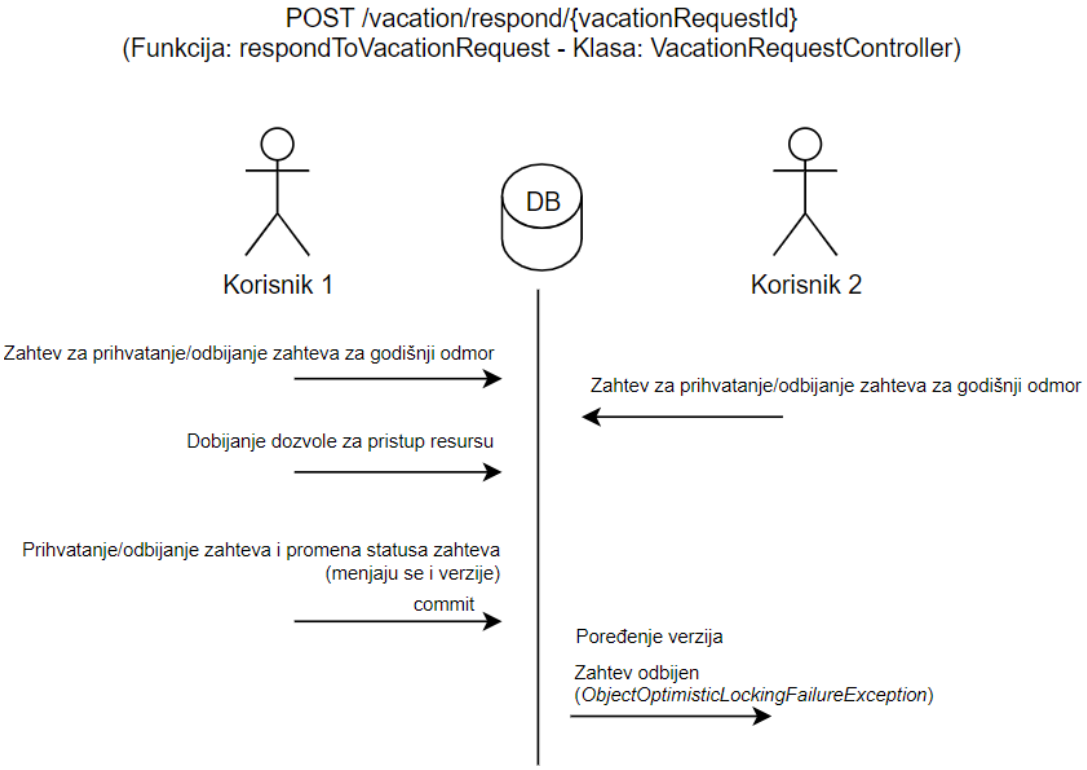
- Koristimo metod optimističkog zaključavanja. Kada korisnik hoće da prihvati ili odbije neku ponudu, vrši se provera verzije date torke u tabeli kako bi se proverilo da li korisnik ima staru verziju koju je već dobio ili se prilikom prihvatanja/odbijanja desilo da je neki drugi korisnik već prihvatio/odbio tu ponudu, povećao verziju i promenio status narudžbenice i ponuda. Ukoliko ima istu verziju torke, koju je dobio, korisnik obrađuje ponude i narudžbenicu, ukoliko ne, transakcija se prekida i zahtev se mora ponoviti.
  Pored toga, pri svakom pozivu funkcije za obradu ponuda i narudžbenice, proverava se status narudžbenice, tj. da li je narudžbenica već obrađena. Ukoliko je narudžbenica već obrađena, zahtev se prekida. Pored gore pomenute klase *DrugsOrder*, polje *version* je dodato i u klasu *Offer*.

```
@Override
public void acceptOffer(long offerId) throws Exception {
    Offer selectedOffer = offerRepository.findOneById(offerId);
    long orderAuthorId = selectedOffer.getOrder().getAdministrator().getId();
    Authentication authentication = SecurityContextHolder.getContext().getAuthenti

    if(selectedOffer.getOrder().getStatus() == OrderStatus.Fulfilled)
        throw new OrderAlreadyFulfilledException();
```

Situacija 4: ***Prihvatanje/odbijanje zahteva za godišnji odmor zaposlenog***
"Prihvatanje/odbijanje zahteva za godišnji odmor od strane više administratora apoteke istovremeno"

Administratori apoteke imaju mogućnost izmene informacija, tj. profila apoteke.
Ukoliko dva administratora pokušaju istovremeno da izmene informacije apoteke, postoji mogućnost da izmene koje je jedan uneo, prepišu izmene onog drugog.

Grafički prikaz:



Rešenje:

- Za rešavanje konflikta koji mogu nastati ukoliko više administratora apoteke prihvati/odbija zahtev u isto vreme, ili ukoliko jedan pokuša da prihvati već odbijeni zahtev ili suprotno, korišćen je pristup optimističnog zaključavanja podataka. Transakciona metoda confirmVacationRequest će rollback-ovati stanje za svako narušavanje verzije podataka. Verzija podataka smeštena je u entitet *VacationRequest*. Situacije kao što su različita odobravanja/odbijanja i promene statusa nam pokazuju da je optimističko zaključavanje, dobar izbor i kada postoji veća mogućnost za preplitanje, ali kada nam je bitno da jedna transakcija uspe.

```java
@Entity
public class VacationRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Version
    @Column(nullable = false, columnDefinition = "int default 1")
    private Long version;
```