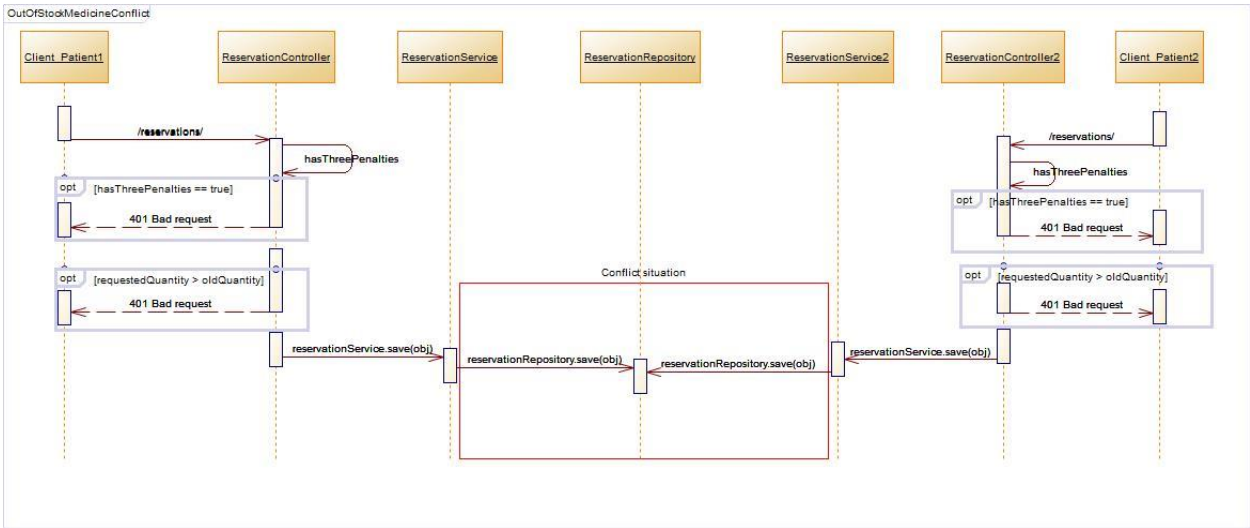
Opis konfliktnih situacija – Student 1

1. Više istovremenih korisnika aplikacije ne može da rezerviše lek koji je u međuvremenu postao nedostupan

Prilikom rezervacije leka se ažurira i količina leka koja se nalazi na stanju u apoteci. Situacija koja može dovesti do konflikta je sledeća: Korisnik1 želi da poruči Lek1 kojeg na stanju ima 3 komada. On kreira svoju rezervaciju, birajući sva 3 komada, tako ostavljajući količinu leka u apoteci na 0, međutim, za to vreme dolazi korisnik2 koji takođe želi da rezerviše Lek1. Korisnik2 pokušava da rezerviše 2 komada leka, i pošto je u međuvremenu korisnik1 već kreirao svoju rezervaciju, leka1 na stanju više nema, ali se ova situacija ne prijavljuje korisniku2. Detaljnije je ova situacija prikazana na sledećem dijagramu - Slika 1.



Slika 1

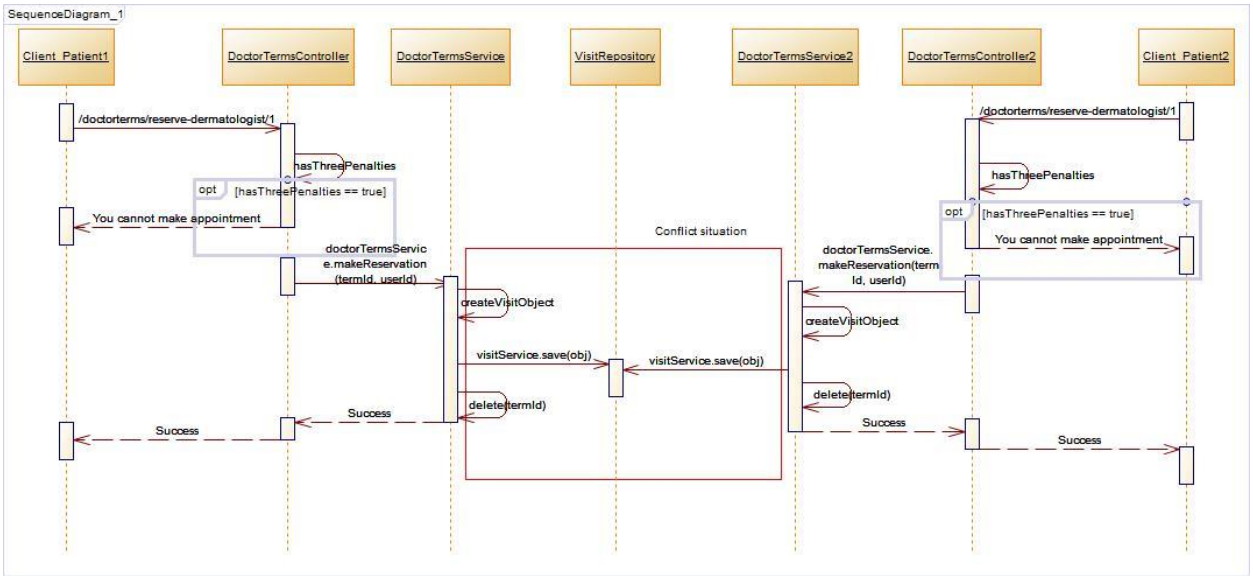
Da bi se razrešila ova konfliktna situacija, korišćen je pesimistički pristup, čime se zaključava resurs rezervacija. Logika koja služi za kreiranje rezervacije je izdvojena u posebnu servisnu metodu koja je zatim označena kao transakciona. Ovim se dobija garancija da će se ova metoda izvršiti od početka do kraja i da drugi klijenti neće moći da menjaju stanje tabele, dok se trenutna metoda ne izvrši.

```
reservation = reservationService.createReservation(reservation);
if (reservation == null) {
    return new ResponseEntity<ReservationDTO>(HttpStatus.BAD_REQUEST);
}
```

```
@Transactional
@Override
public Reservation createReservation(Reservation reservation) {
    reservation.setStatus(Status.RESERVED);
    reservation = save(reservation);
```

2. Pregledi koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika

Prilikom rezervisanja pregleda kod doktora, pacijenti imaju mogućnost da odaberu koji termin im najviše odgovara i da ga zatim klikom na dugme rezervišu. Međutim, tu može doći do konfliktne situacije gde dva različita pacijenta rezervišu u isto vreme isti termin kod istog doktora. Na Slika 2 se može videti dijagram prethodno objašnjene konfliktne situacije.



Slika 2

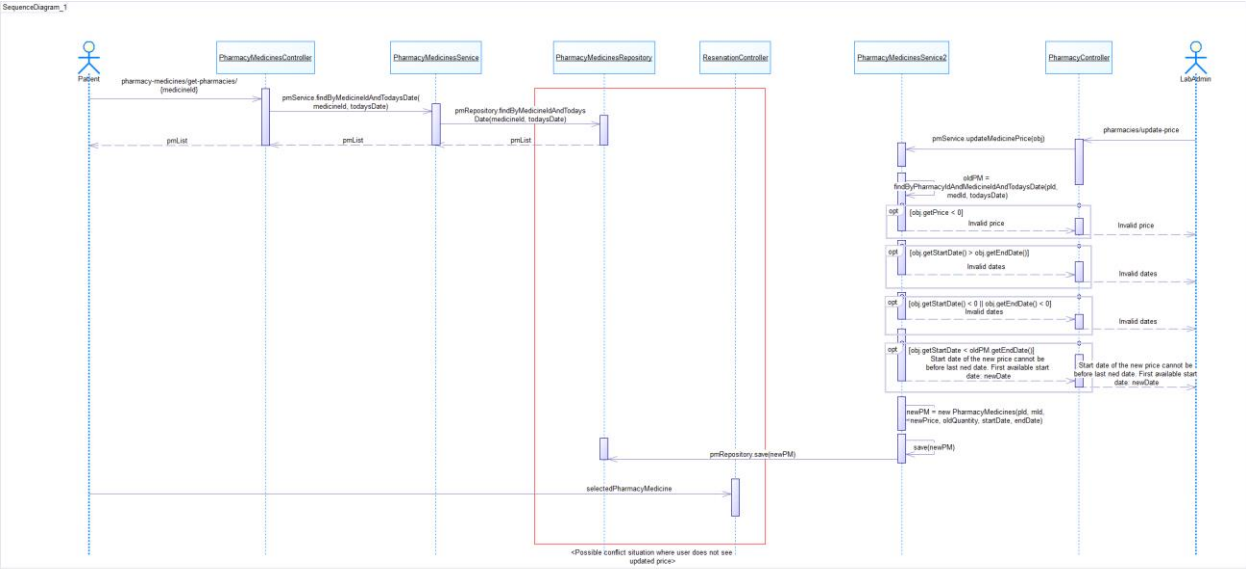
Da bi se ova situacija rešila, potrebno je zaključati resurs koji služi za odabir predefinisanog termina. Kako se ova servisna metoda sa anotira kao transakcionom, sigurni smo da će pri odabiru predefinisanog

```
@Transactional
public Visit makePharmacistAppointmentPatient(Visit newReservation, Long patientId) {
    newReservation.setPatientId(patientId);
    newReservation.setFinish(newReservation.getStart().plusHours(1));

    if(!checkTermTaken(newReservation))
        return null;
    if(checkTermDerm(newReservation, newReservation.getDoctorId()))
        return null;
    if(checkIfInWorkingHours(newReservation))
        return null;

    newReservation.setStatus(Status.RESERVED);

    save(newReservation);
    return newReservation;
}
```

termina od strane klijenta biti eliminisana mogućnost da se izvrše dve rezervacije, istog termina, istovremeno.

3. Izmena cene leka za vreme rezervacije istog (dodatna situacija)

Prilikom rezervacije leka, pacijent bira lek koji želi da rezerviše klikom na dugme iz ponuđene liste lekova. Potencijalna konfliktna situacija do koje može doći, jeste da se za to vreme, dok se klijent nalazi na stranici za rezervaciju leka, izvrši izmena cene istog tog leka od strane administratora apoteke. U ovom slučaju korisnik neće biti obavešten o izmeni i automatski će biti omogućena rezervacija leka iako je cena izmenjena. Dijagram ovog problema se nalazi na Slika 3.



Kako bi se sprečila ova situacija, mora se zaključati tabela u kojoj se dobavlja taj izabrani lek od strane pacijenta, tako da se koristi pesimistički pristup za rešavanje ovog problema. Nakon toga je metoda za rezervaciju leka, kao i za izmenu cene, označena kao transakciona, čime se osigurava njeno izvršavanje do kraja.

```
@Transactional
@Override
public ResponseObject updateMedicinePrice(PharmacyMedicineAddRemoveObject obj) {
    PharmacyMedicines oldPM = findByPharmacyIdAndMedicineIdAndTodaysDate(obj.getPharmacyId(), obj.getMedicineId(),
            new Date().getTime());
    double price = obj.getPrice();
    if (price < 0) {
        return new ResponseObject(400, "Invalid price");
    }

    long startDate = obj.getStartDate();
```

```java
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select pm from PharmacyMedicines pm where pm.pharmacy.id= ?1 and pm.medicine.id = ?2 and ?3 between pm.st
PharmacyMedicines findByPharmacyIdAndMedicineIdAndTodaysDate(Long pharmacyId, Long medicineId, long todaysDate);
```

```java
@Transactional
@Override
public Reservation createReservation(Reservation reservation) {
    reservation.setStatus(Status.RESERVED);
    reservation = save(reservation);

    PharmacyMedicines pm = pmService.updateAfterReservation(reservation, reservation.getQuantity());
    if (pm == null) {
        return null;
    }

    return reservation;
}
```