Решавање конфликтних ситуација – Студент 3

Ажурирање пацијентових поена приликом издавања резервације

Када фармацеут жели да изда резервацију пацијенту потребно је коректно ажурирати поене пацијента које добија у зависности које лекове је резервисао. Уколико се не брине о конкурентном приступу може доћи до неконзистентности података. Могућа ситуација је да два или више фармацеута желе да издају резервацију у истом тренутку и ту се може јавити проблем да се деси неправилно ажурирање поена пацијента, могуће је да добије више поена него што је предвиђено.

Проблем је решен тако што се приликом читања објекта класе која представља резервацију лекова (*MedicineReservation*) ради песимистичко закључавање. Овим закључавањем се постиже да уколико више различитих нити покуша да приступи објекту у исто време, само једна нит ће добити објекат док друге чекају да га ослободи нит која га је заузела. Метода репозиторијума преко које је извршено закључавање је приказана на Илустрација 1 .

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select mr from MedicineReservation mr where mr.id=:id and mr.active=true and mr.reservationStatus=0")
Optional<MedicineReservation> getMedicineReservationToIssue(@Param("id") Long id);
```

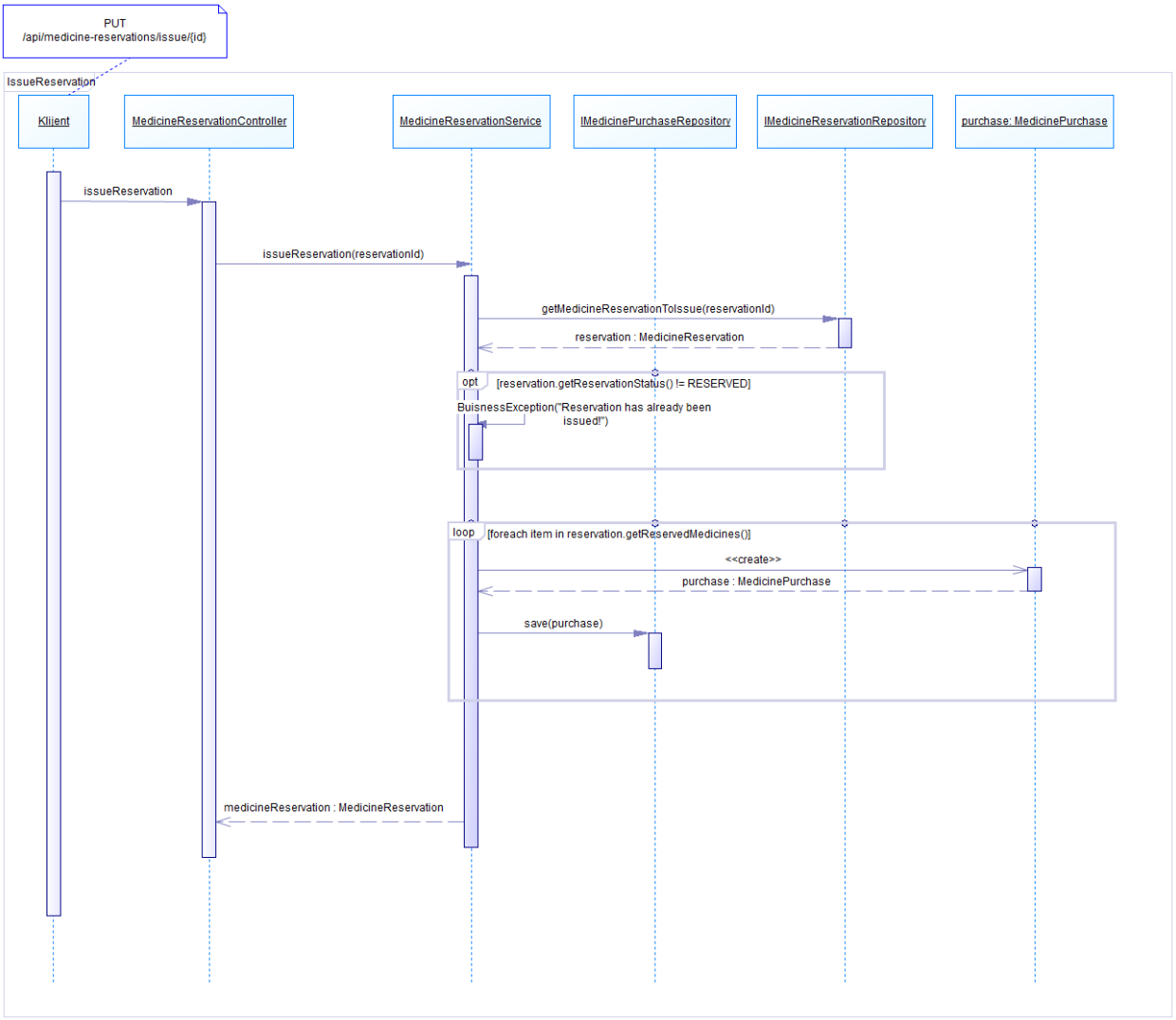
Илустрација 1 Песимистичко закључавање резервације

У сервисној методи (Илустрација 2) након добављања објекта резервације се проверава да ли је резервација већ издата и ако јесте онда се изазива изузетак и фармацеут добија одговарајуће обавештење. Резервација ће само бити издата ако је нека друга нит није већ приступила том објекту и издала резервацију. Тиме се омогућава да само један фармацеут може да изда резервацију, док ће се осталима пријавити проблем. Уколико резервација није издата, онда се прелази на додељивање поена пацијенту за сваки лек који је резервисао, док се на крају мења статус резервације на ажуриран чиме се спречавају друге нити које евентуално чекају на објекат, да направе проблеме око ажурирања података.

```
@Override
public MedicineReservation issueReservation(Long id) {
    MedicineReservation medicineReservation = medicineReservationRepository.getMedicineReservationToIssue(id)
            .orElseThrow(() -> new BusinessException("Reservation has already been issued!"));

    if (medicineReservation.getReservationStatus() != ReservationStatus.RESERVED) {
        throw new BusinessException("Reservation has already been issued!");
    }

    Patient patient = medicineReservation.getPatient();
    medicineReservation.getReservedMedicines().forEach(item ->{
        MedicinePurchase medicinePurchase = new MedicinePurchase(item.getQuantity(),
                item.getPrice(), medicineReservation.getPharmacy(), LocalDate.now(), item.getMedicine());
        medicinePurchaseRepository.save(medicinePurchase);
        patient.addPoints(item.getMedicine().getPoints() * item.getQuantity());
    });

    patientRepository.save(patient);
    medicineReservation.setReservationStatus(ReservationStatus.PICKED);

    return medicineReservation;
}
```

Илустрација 2 Сервисна метода за издавање резервације

Дијаграм секвенце на којем је приказан посматрани ток од клијент до сервиса је приказан на Илустрација 3 . На дијаграму су приказане само најбитније интеракције, док су мање битне изостављене.



Илустрација 3 Дијаграм секвенце издавања резервације

Резервисање слободних термина код дерматолога

Приликом заказивања термина код дерматолога, одабира се један од унапред креираних слободних термина. Термине код дерматолога може да одабира пацијент, а може и дерматолог приликом прегледа када заказује наредни преглед за прегледаног пацијента. Могућа ситуација је да у исто време покушају и дерматолог и пацијент да заузму исти слободни термин и тада се јавља проблем ко ће заузети тај термин

Овај проблем је решен увођењем поља *version* у класу *Appointment*, односно урађено је оптимистичко закључавање. То значи да када се добави објекат доступног прегледа и покуша да се уради нека измена над њим, провери се прво верзија, односно да тај објекат у међувремену није

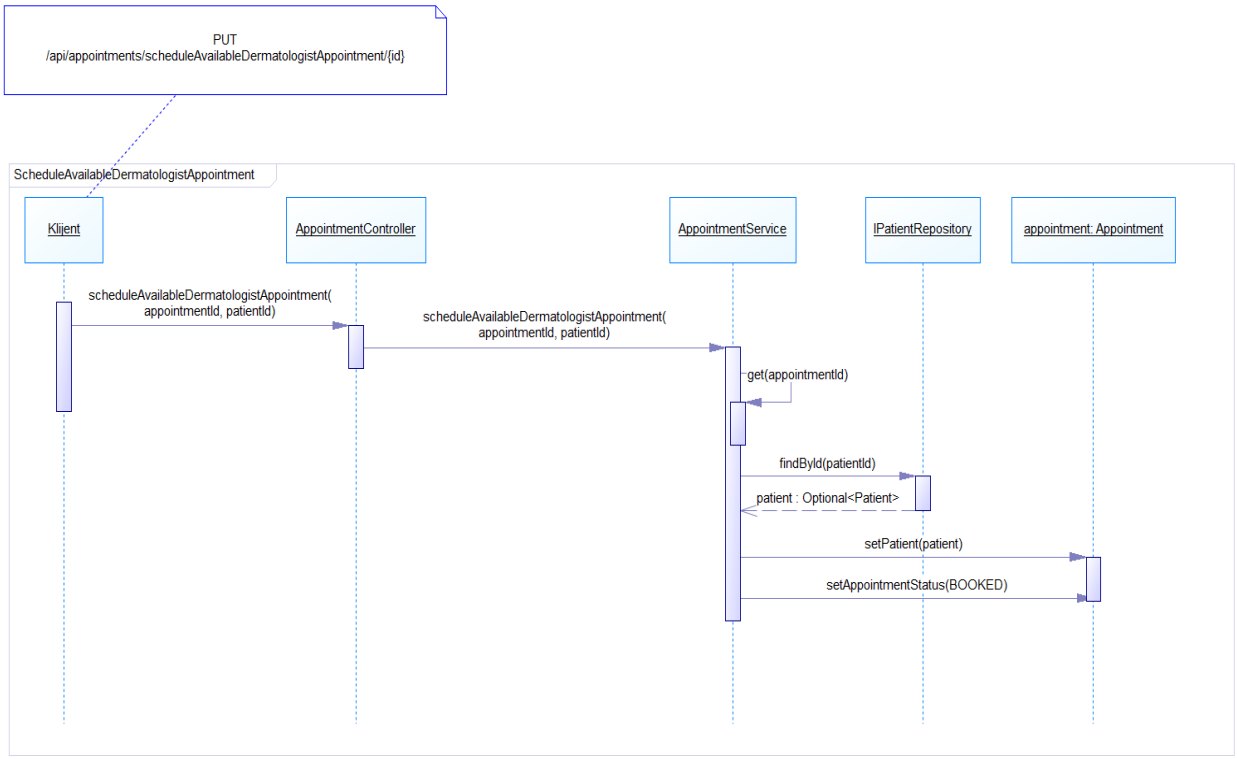
променила нека друга нит и ако је верзија промењена изазива се изузетак и прекида рад. Тиме се постиже да само једна нит може да резервише доступан преглед, док све остале неће моћи и биће пријављена одговарајућа грешка.

Сервисна метода (Илустрација 4) добавља доступан преглед и поставља му статус да је заузет, чиме се онемогућава било која друга нит да у исто време заузме тај преглед.

```
@Override
public void scheduleAvailableDermatologistAppointment(Long appointmentId, Long patientId) {
    Appointment appointment = this.get(appointmentId);
    Patient patient = this.patientRepository.findById(patientId).orElseThrow(
            () -> new BusinessException("Patient with id  " + patientId + " does not exist"));

    appointment.setPatient(patient);
    appointment.setAppointmentStatus(AppointmentStatus.BOOKED);
}
```

Илустрација 4 Сервисна метода за заказивање доступног прегледа

Дијаграм секвенце на којем је приказан посматрани ток је приказан на Илустрација 5 .



Илустрација 5 Дијаграм секвенце заказивања слободног прегледа код дерматолога

Ажурирање лека након завршеног прегледа

Након што фармацеут или дерматолог обележи да је преглед завршен, уколико је било преписаних лекова тада се за сваки од лекова скида преписана количина из апотеке где је преглед одржан. Уколико је у исто време у истој апотеци одржано више прегледа и на тим прегледима су преписани неки исти лекови, тада је могућа да се деси следећа ситуација. На првом прегледу је преписано 10

аспирина и на другом прегледу 20 аспирина, а укупна количина аспирина је 25. Тада се дешава ситуација да нема довољно аспирина за оба прегледа и није могуће коректно завршити прегледе.

Овај проблем је решен тако што је добављање објекта класе која представља стање лека у апотеци (*MedicineStock*) закључано песимистичко. Односно приликом ажурирања стања лека само једна нит може у једном тренутку да ажурира исти објекат и тиме се постиже да када друга нит добије тај лек прво провери доступну количинуи ако је она мања од тражене изазива се изузетак и прекида се програм. Добављање објекта песимистички је приказано на Илустрација 6 .

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select ms from MedicineStock ms where ms.active=true and ms.pharmacy.id=:pharmacyId and ms.id=:stockId")
Optional<MedicineStock> getMedicineStockInPharmacy(@Param("pharmacyId") Long pharmacyId,
                                                   @Param("stockId") Long stockId);
```

Илустрација 6 Добављање објекта MedicineStock

Сервисна метода (Илустрација 7) у којој се позива закључана метода за добављање је проглашена за трансакциону и специфицирано је у потпису да у случају *BuisnessException* изузетка уради ролбек. Ролбек је рађен из разлога зато што је могуће да прва три лека буде на стању у апотеци и они се исправно ажурирају док четврти лек не буде на стању и изазове изузетак, тада се поништавају измене над претходна три лека.

```
@Override
@Transactional(rollbackFor = BusinessException.class)
public void concludeAppointment(String reportText, Long pharmacyId, Long patientId, Long appointmentId, List<MedicineStockConcludeDTO> medicineStockConcludeDTOS)
```

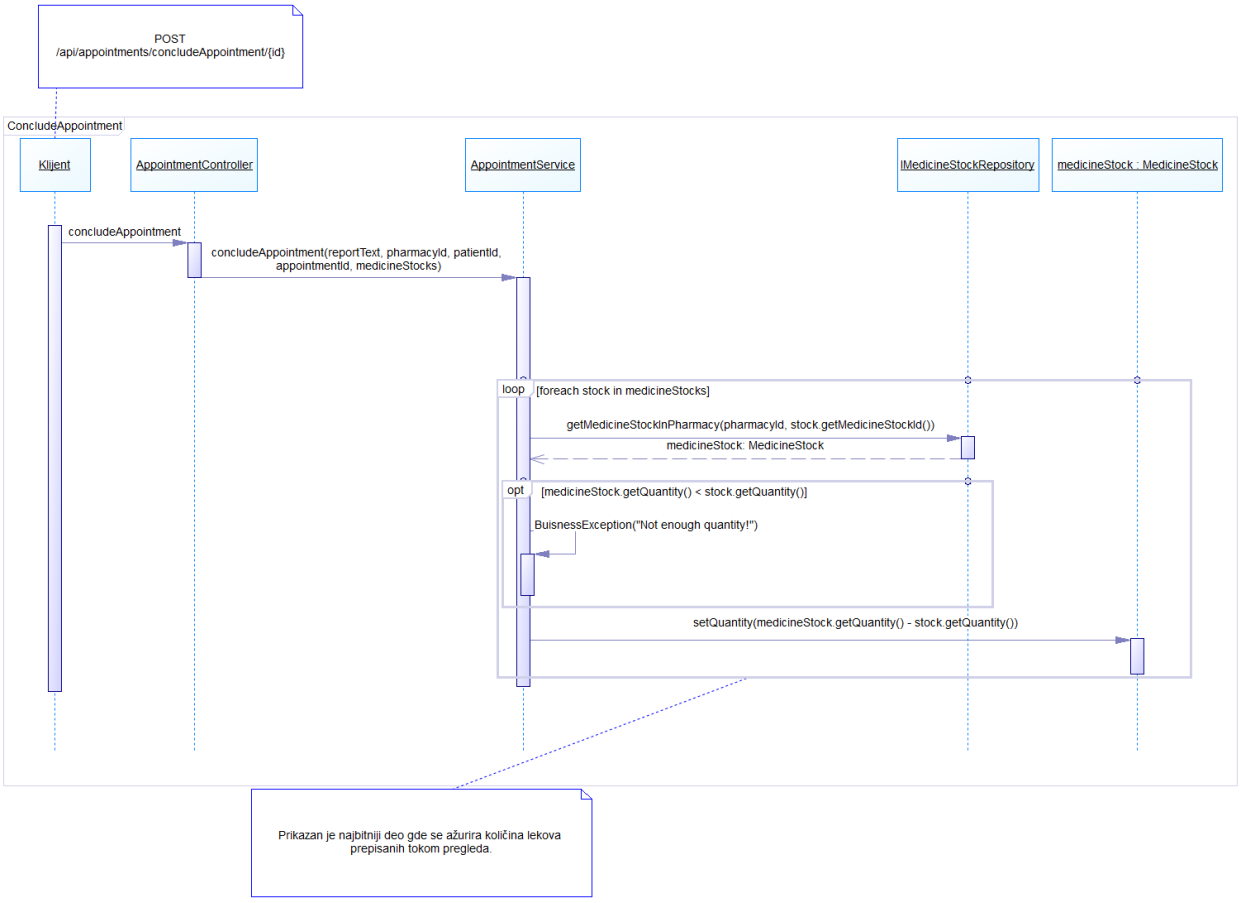
Илустрација 7 Сервисна метода за завршавање прегледа

Најбитнији део сервисне методе где се врши пролазак кроз све преписане лекове и добављање одговарајућег објекта који представљ стање лека у апотеци је приказан на Илустрација 8 .

```
for(MedicineStockConcludeDTO medicineStockConcludeDTO : medicineStockConcludeDTOS) {
    MedicineStock medicineStock = medicineStockRepository.getMedicineStockInPharmacy(pharmacyId, medicineStockConcludeDTO.getMedicineStockId())
            .orElseThrow(() -> new BusinessException("Medicine stock with id " + medicineStockConcludeDTO.getMedicineStockId() + " does not exist"));

    if (medicineStock.getQuantity() < medicineStockConcludeDTO.getQuantity()) {
        throw new BusinessException("Not enough quantity of medicine " + medicineStock.getMedicine().getName() + "!");
    }
```

Илустрација 8 Део сервисне методе где се врши добављање из закључане методе

Дијаграм секвенце на којем је приказан посматрани ток је приказан на Илустрација 9. Нису приказане све интеракције, зато што има превише њих које нису битне, а не могу стати на дијаграм, већ је приказано ажурирање стања лекова.



Илустрација 9 Дијаграм секвенце ажурирања стања лекова