Решавање конфликтних ситуација – Студент 1

У коду су решене неке од проблематичних ситуација које могу настати приликом поступка:

- Резервације лекова
- Отказивања резервације лекова
    - Ажурирање стања лека у апотеци
- Аутоматског поништавања истеклих резервација
    - Ажурирање стања лека у апотеци
    - Додела казнених поена кориснику због истеклих резервација
- Ажурирање броја казнених поена на почетку месеца
- Оцењивање апотека
- Оцењивање запослених

У овом документу су детаљно описане 3 проблематичне ситуације:

- Резервација лекова
- Отказивање резервације лекова
- Оцењивање запослених

Више корисника резервише лек који је у међувремену постао недоступан

Корисник приликом резервације специфицира количину и датум преузимања поруџбине, па је потребно коректно ажурирати количину тог лека у апотеци, тако да се стање лека умањи за поручену количину.

Када постоји више корисника који у исто време наручују исти лек у истој апотеци, може се десити да један од њих пошаље захтев за резервацију све преостале количине лекова на стању, а да је у исто време на сервер стигао захтев за резервацију лека од другог корисника. У том случају може се десити да оба захтева могу да очитају количину лека на стању пре него што су га ажурирали, и да оба захтева буду успешна, иако укупна сума наручених лекова превазилази доступну количину лекова на стању.

На пример, ако постоји 100 лекова на стању у апотеци, и стигну 2 захтева за по 100 лекова, онда се може десити да оба захтева очитају да је у апотеци доступно 100 лекова, и обе резервације могу бити успешно направљене. Тада би први захтев поставио стање лека у апотеци на 0, а други захтев би пошто је такође очитао да име 100 лекова на стању, такође на крају резервације поставио стање на 0, и обе резервације би биле успешне. Дакле наручено је 200 лекова иако их је било само 100 на стању, а то је неприхватљиво.

Да би се овај проблем решио, користи се песимистично закључавање објекта класе која моделује количину лека на стању у апотеци (*MedicineStock*). Овим се постиже да једна нит може да приступи количини лека на стању у апотеци и закључава га све док се поступак резервације не заврши, док друге нити чекају да тај ресурс постане поново откључан. Пошто сматрамо да је важно да се количина лека увек исправно ажурира, на свим местима где радимо са лековима користимо ову методу са песимистичким закључавањем (Илустрација 1).

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

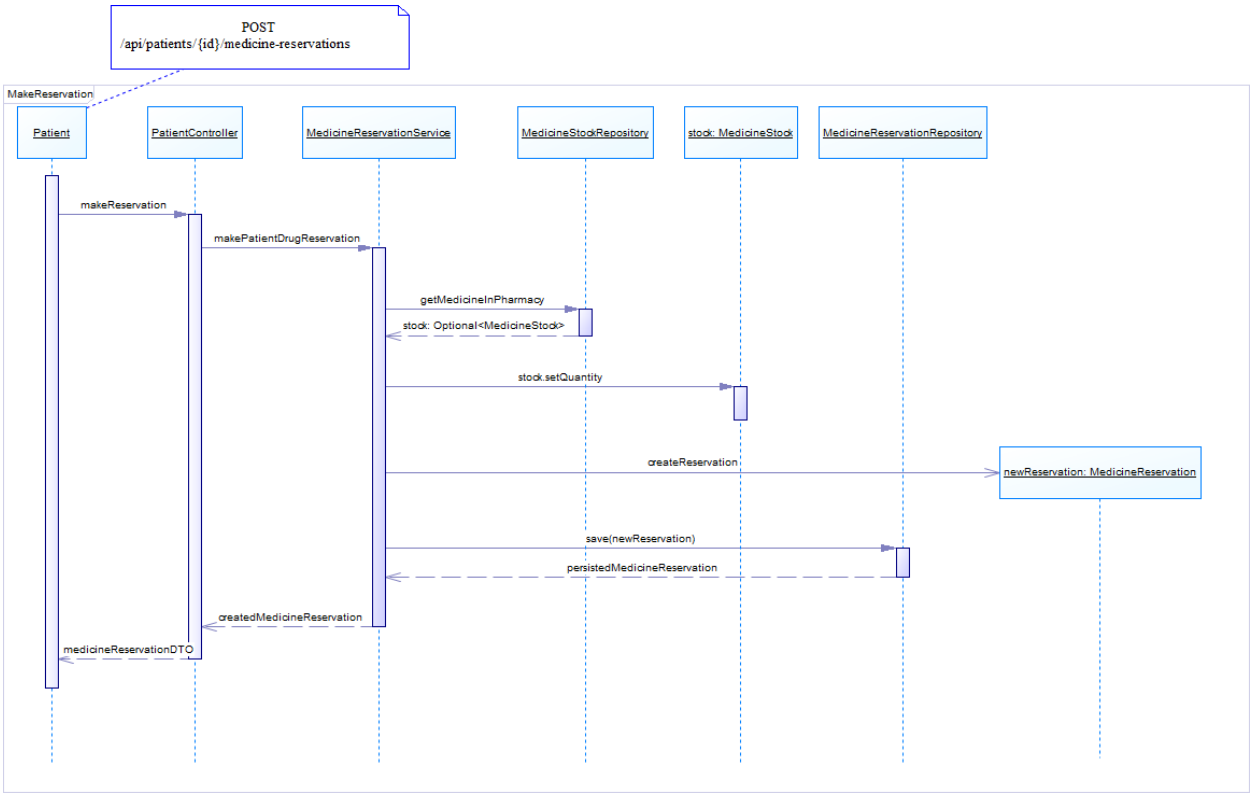
Илустрација 1 Стање лека у апотеци које се песимистично закључава приликом добављања. Метода се налази у IMedicineStockRepository

Приликом резервације лека позива се сервисна метода *makePatientDrugReservation* из класе *MedicineReservationService*. Та метода има анотацију *Transactional* са параметром *rollbackFor* где је наведено да се *rollback* ради за *ResponseStatusException.* (Илустрација 2).

```java
@Override
@Transactional(rollbackFor = ResponseStatusException.class)
public MedicineReservation makePatientDrugReservation(Long patientId,
                                                      Long pharmacyId,
                                                      Long reservedDrugId,
                                                      Integer quantity,
                                                      LocalDateTime reservedAt,
                                                      LocalDateTime reservationDeadline) {
```

Илустрација 2 Сервисна метода која се позива приликом резервације лека

Дијаграм секвенце приказује је ток наручивања лека од стране пацијента (Илустрација 3). На дијаграму нису приказане провере типа да ли пацијент, апотека и лек уопште постоје, да ли су датуми важећи и друге ситне провере као ни детаљи прављења резервације, већ су само приказане најбитније интеракције због којих могу настати проблеми приликом конкурентног приступа.



Илустрација 3 Дијаграм секвенце за резервацију лека од стране пацијента

Ажурирање количине лека на стању након отказивања резервације лека

Приликом отказивања резервације лека, потребно је увећати количину лека на стању у апотеци за количину која је била резервисана, да би ти лекови поново постали доступни за резервацију.

Овде због повреде Бернштајнових услова може доћи до трке до података, и тада стање лека у апотеци може бити погрешно ажурирано. То се на пример може десити када један корисник откаже резервацију лека у апотеци, а други корисник у исто време откаже резервацију истог лека у истој апотеци. Тада може као и у претходном описаном примеру доћи до погрешног ажурирања стања лека.

Све операције које ажурирају стање лека у исто време када се врши и отказивање лека у апотеци могу да резултују трком до података. То могу бити на пример друга отказивања, поручивања лека, прихватање понуде за наруџбеницу.

Због тога је урађено песимистично закључавање објекта класе који моделује количину лека на стању у апотеци (*MedicineStock*). Овим се постиже да једна нит може да приступи количини лека на стању у апотеци и закључава га све док се поступак резервације не заврши, док друге нити чекају да тај ресурс постане поново откључан. Пошто сматрамо да је важно да се количина лека увек исправно ажурира, на свим местима где радимо са лековима користимо ову методу са песимистичким закључавањем (Илустрација 4).

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select ms from MedicineStock ms where ms.active=true and ms.pharmacy.id=:id and lower(ms.medicine.code)=:code")
Optional<MedicineStock> getByMedicineCodeForPharmacy(@Param("code") String medicineCode, @Param("id") Long pharmacyId);
```

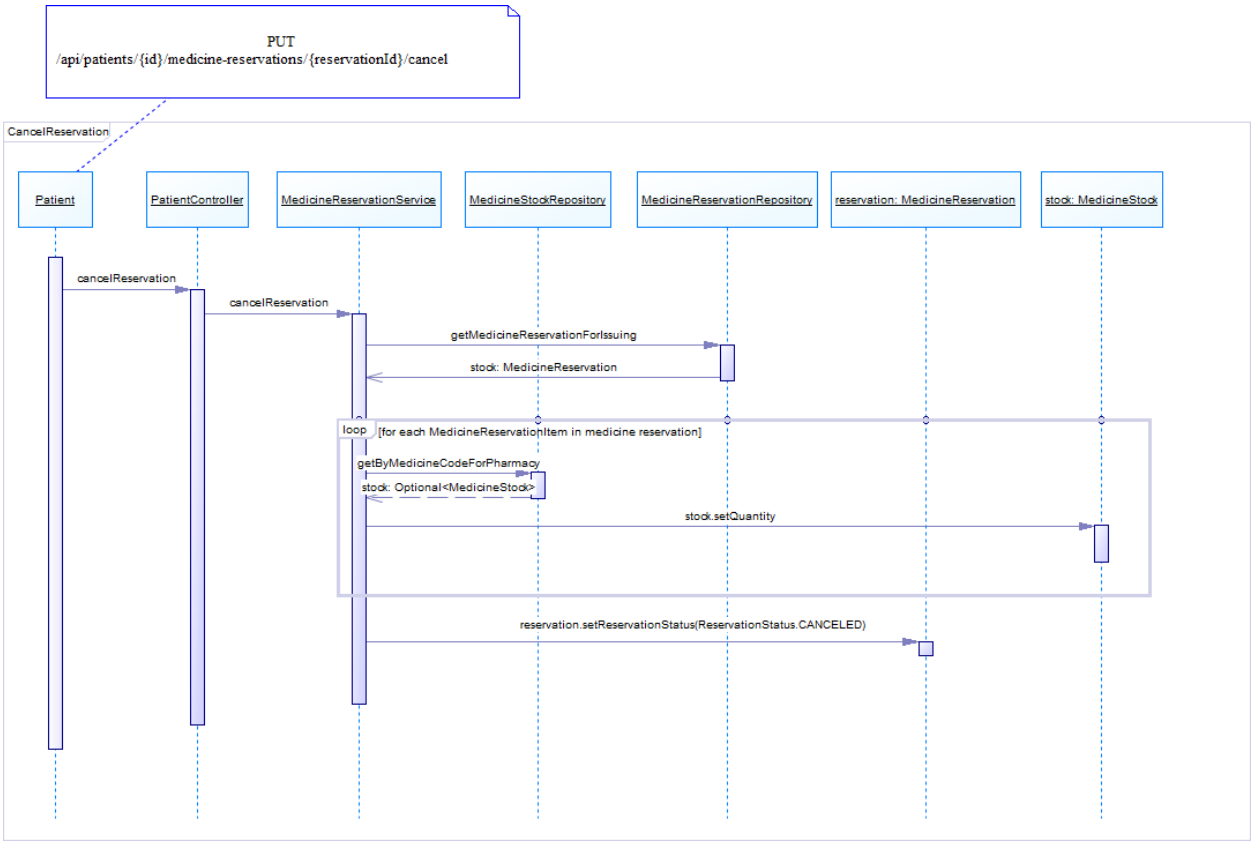
Илустрација 4 Стање лека у апотеци које се песимистично закључава приликом добављања. Метода се налази у IMedicineStockRepository

Приликом отказивања лека позива се сервисна метода *cancelReservation* из класе *MedicineReservationService*. Ова метода (Илустрација 5) има анотацију *Transactional* са параметром *rollbackFor* где је наведено да се *rollback* ради за *ResponseStatusException*.

```
@Override
@Transactional(rollbackFor = ResponseStatusException.class)
public void cancelReservation(Long reservationId) {
```

Илустрација 5 Сервисна метода за отказивање резервације лека

Дијаграм секвенце приказује је ток наручивања лека од стране пацијента (Илустрација 6). На дијаграму су приказане само кључне интеракције.



Илустрација 6 Дијаграм секвенце за отказивање резервације лека и ажурирање стања лека у апотеци

Ажурирање просечне оцене приликом оцењивања запосленог

Ажурирање просечне оцене дерматолога се врши када пацијент оцени запосленог. То се постиже тако што се итерира кроз све оцене које је запослени добио, и рачуна се просек. Та оцена се затим записује као просечна оцена запосленог. Овакав начин ажурирања просечне оцене приликом оцењивања се користи да би се избегла ситуација да се просечна оцена рачуна приликом сваког добављања просечне оцене запосленог. Претпоставка је да се акција оцењивања много ређе јавља од акције добављања и приказивања просечне оцене.

Проблем са овим ажурирањем може настати ако два корисника оцене истог запосленог у исто време. Тада као и у претходно описаним ситуацијама може доћи до трке до података. Сваки од пацијената би приликом оцењивања израчунао просечну оцену и важила би просечна оцена оног пацијента који би је последњи уписао.

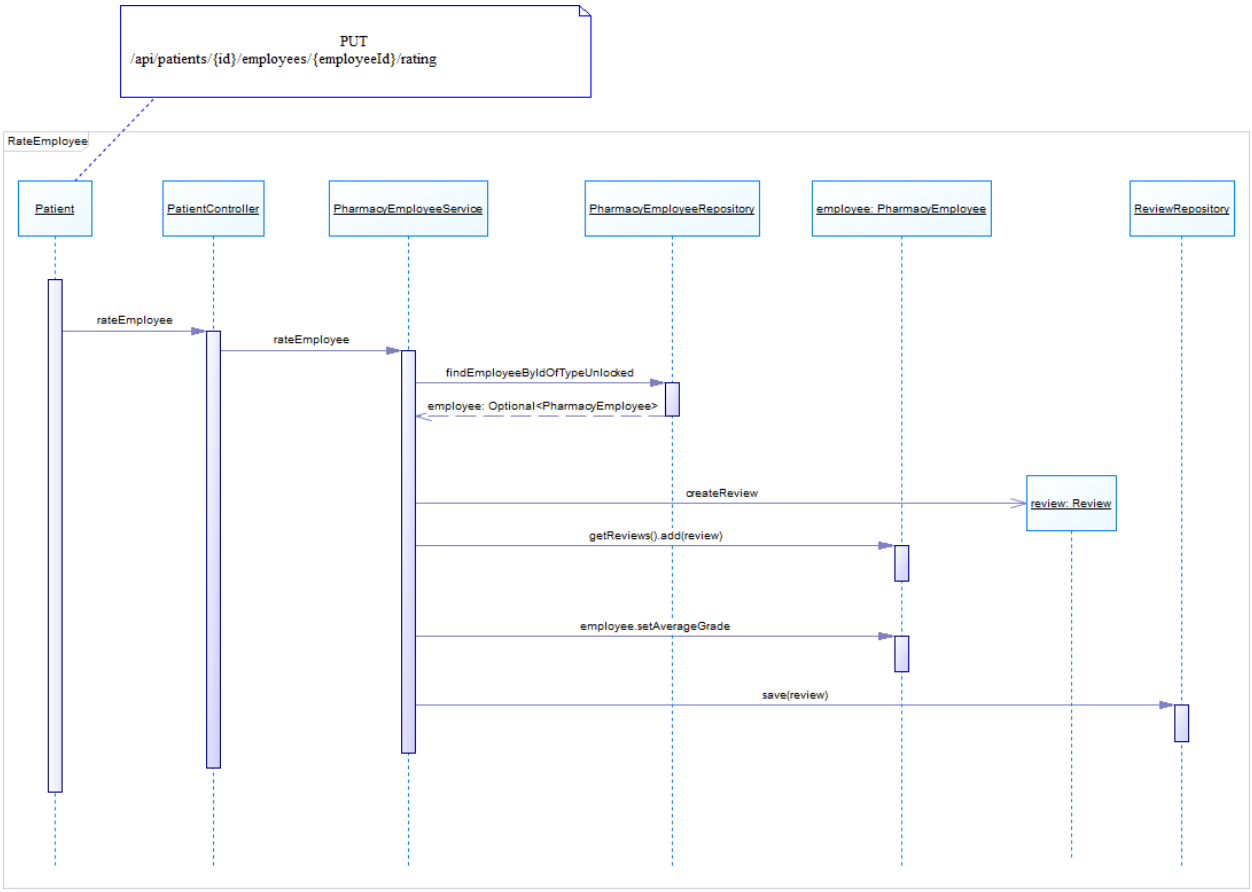
Због тога је урађено оптимистичко закључавање објекта класе која моделује запосленог у апотеци (*PharmacyEmployee*). Глобални *ExceptionHandler* ће у случају грешке вратити адекватну поруку јер проблем није критичан за функционисање система. Оптимистичко закључавање се ради због потенцијално бољих перформанси у односу на песимистичко закључавање због тога што се оцењивање врши ређе па се и просечна оцена ређе ажурира, али се често та израчуната оцена приказује.

Приликом отказивања лека позива се сервисна метода *rateEmployee* из класе *PharmacyEmployeeService*. Ова метода (Илустрација 7) има анотацију *Transactional* са параметром *rollbackFor* где је наведено да се *rollback* ради за *ResponseStatusException*.

```
@Override
@Transactional(rollbackFor = ResponseStatusException.class)
public void rateEmployee(Long patientId, Long employeeId, EmployeeType employeeType, Integer rating) {
```

Илустрација 7 Метода за оцењивање запосленог

Дијаграм секвенце приказује је оцењивања дерматолога од стране пацијента (Илустрација 8). На дијаграму су приказане само кључне интеракције.



Илустрација 8 Поступак оцењивања запосленог