ПРОБЛЕМИ КОНКУРЕНТНОГ ПРИСТУПА РЕСУРСИМА И НАЧИНИ ЊИХОВОГ РЕШАВАЊА

У овом документу биће представљена три сценарија до којих може доћи ако се проблем конкурентног приступа ресурсима не ријеши адкеватно. Сваки сценарио биће описан и биће описан начин решавања проблема који дати сценарио уводи.

1. Издавање еРецепата

Издавање лекова је једна од основих функционалности система и један од начина издавања лијека је управо преко еРецепта. Корисник учита QR код у којем су наведени лекови који треба да буду издати. Затим, приказују се апотеке у којима је сваки од датих лекова доступан и пацијент бира апотеку у којем ће резервисати сваки од наведених лекова. У овом сценарију може доћи до неправилног ажурирања количине наведених лекова. Уочено је неколико начина који могу да доведу до овога:

- Може се десити у случају да више корисника истовремено покуша да креира еРецепте који имају заједничке лекове.
- Може се десити у случају да је пацијент алергичан на један од лекова и стања лекова на које није алергичан се ажурирају а његово не. Приликом издавања, или се издају сви, или ниједан
- Може се десити у случају да неки од лекова није на стању, а остали јесу. Они који јесу на стању се ажурирају, а он се. То се у супротности са истим захтјевом наведеним у претходној тачки.

Начин решавања

Кориштен је приступ песимистичгог закључавања за писање (*LockModeType.PESSIMISTIC_WRITE*) над функцијом *getMedicineInPharmacy* у репозиторијуму *IMedicineStockRepository (*Илустрација 2) . Такође, сама метода гдје се врши креирање рецепта,метода *createRecipe* у *PatientService (*Илустрација 1*)* , је анотирана анотацијом *Transactional* са атрибутима *isolation=Isolation.READ_COMMITED* и *rollbackFor=ResponseStatusException.class.*

```
@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 2 Метода за добављање стања лијека у датој апотеци у IMedicineStockRepository

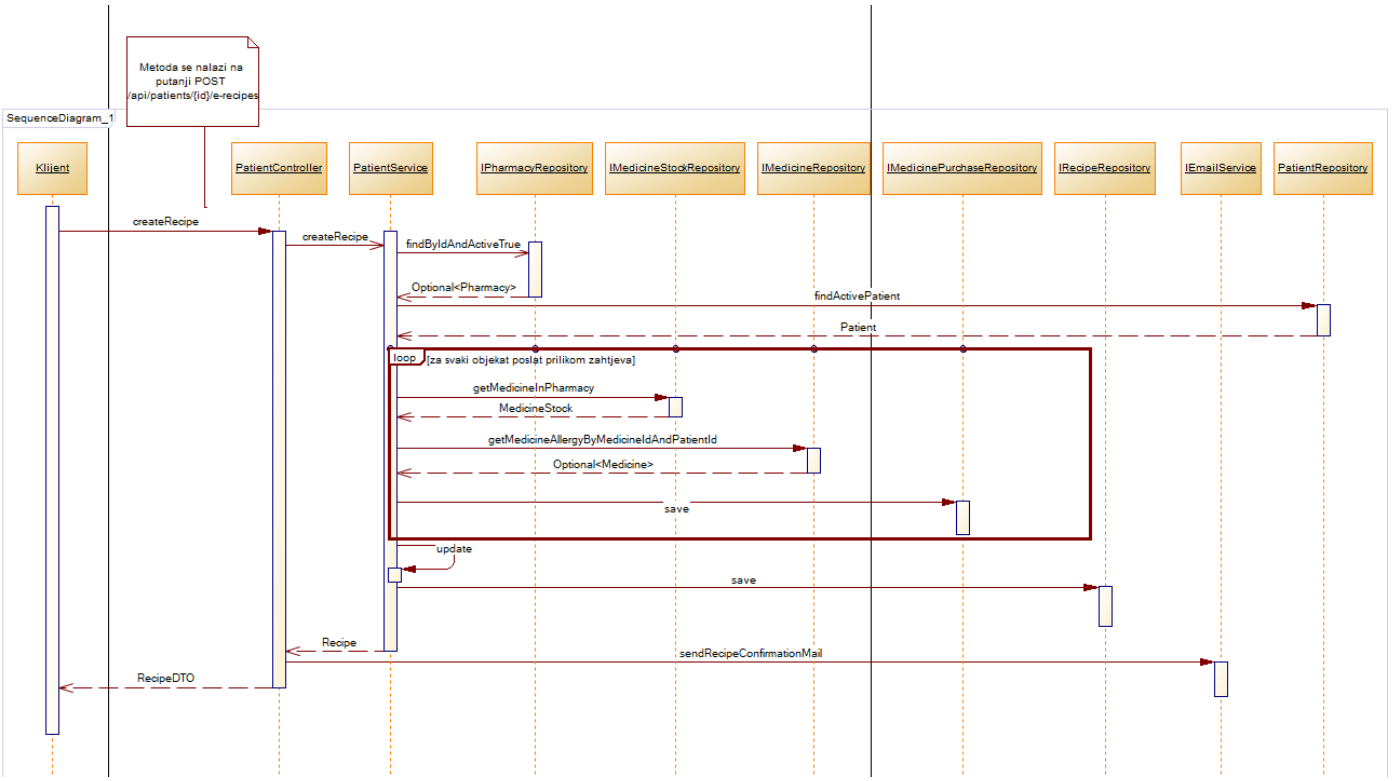
```
@Override
@Transactional(isolation = Isolation.READ_COMMITTED, rollbackFor = ResponseStatusException.class)
public Recipe createRecipe(Long patientId, Long pharmacyId, List<MedicineQRCodeReservationItemDTO> medicines)
```

Илустрација 1 Метода за креирање рецепта у PatientService

Песимистичко закључавање је изабрано из разлога јер се такво закључавање очекује на другим мјестима у апликацији. Количина лијека на стању представља виталну информацију за систем и његово правилно ажурирање је од изузетне важности. Процес резервисања би за клијента требао да буде идентичан сваки пут, а у случају да је кориштено оптимистичко закључавање и да је посјећеност сајта велика, ако би дошло до грешке, решавање би пало на клијента (порука о грешци са сервера би могла да буде у облику 'Освежите страницу.'), што би брзо довело до фрустрација. Песимистичко закључавање је можда спорије, али његовим коришћењем доприносимо униформном искуству корисника.

Што се тиче методе *createRecipe,*она је анотирана са *Transactional* да би се експлицитно дефинисао опсег трансакције, као и да би се специфицирали услови за *rollback* (појава *ResponseStatusException* изузетка), као и ниво изолације. Коришћен је *READ_COMMITED* ниво изолације, јер *Hibernate* заправо може да ријеши и *Unrepeatable read* проблем коришћењем овог нивоа изолације, због коришћења кеша.

Ток извршавања је приказан на слици Илустрација 3.



Илустрација 3 Дијаграм секвенвце за процес прављења ерецепта

2. ПИСАЊЕ ОДГОВОРА НА ЖАЛБЕ

Овај сценарио је мање вјероватан да се деси од претходног, пошто се не очекује превелики број жалби, а поред тога очекује се и донекле усклађен рад администратора система. Међутим, може доћи до конфликтне ситуације када више админа покуша истовремено да одговори на исту жалбу. Теоријски, може доћи до ситуације да се свим администима прикаже порука да су успјешно одговорили на жалбу, а да је сачуван само један одговор.

Начин решавања:

Кориштен је приступ песимистичког закључавања за писање над функцијом *findByIdAndActiveTrueForUpdate* у репозиторијуму *IComplaintRepository (*Илустрација 4*).*

Овдје се можда могао користити и приступ са оптимистичким закључавањем и коришћењем верзија, али овај је изабран пошто је робуснији, и не очекује се толики број одговарања. Поред тога, важно је да клијент добије одговарајући одговор.

Сервисна метода гдје се врши креирања одговора, метода *writeReply* унутар *ComplaintService* је анотирана са *Transactional,* са атрибутима *isolation=Isolation.READ_COMMITED* и *rollbackFor=ResponseStatusException.class* (Илустрација 5). Резоновање је идентично као у претходном сценарију.

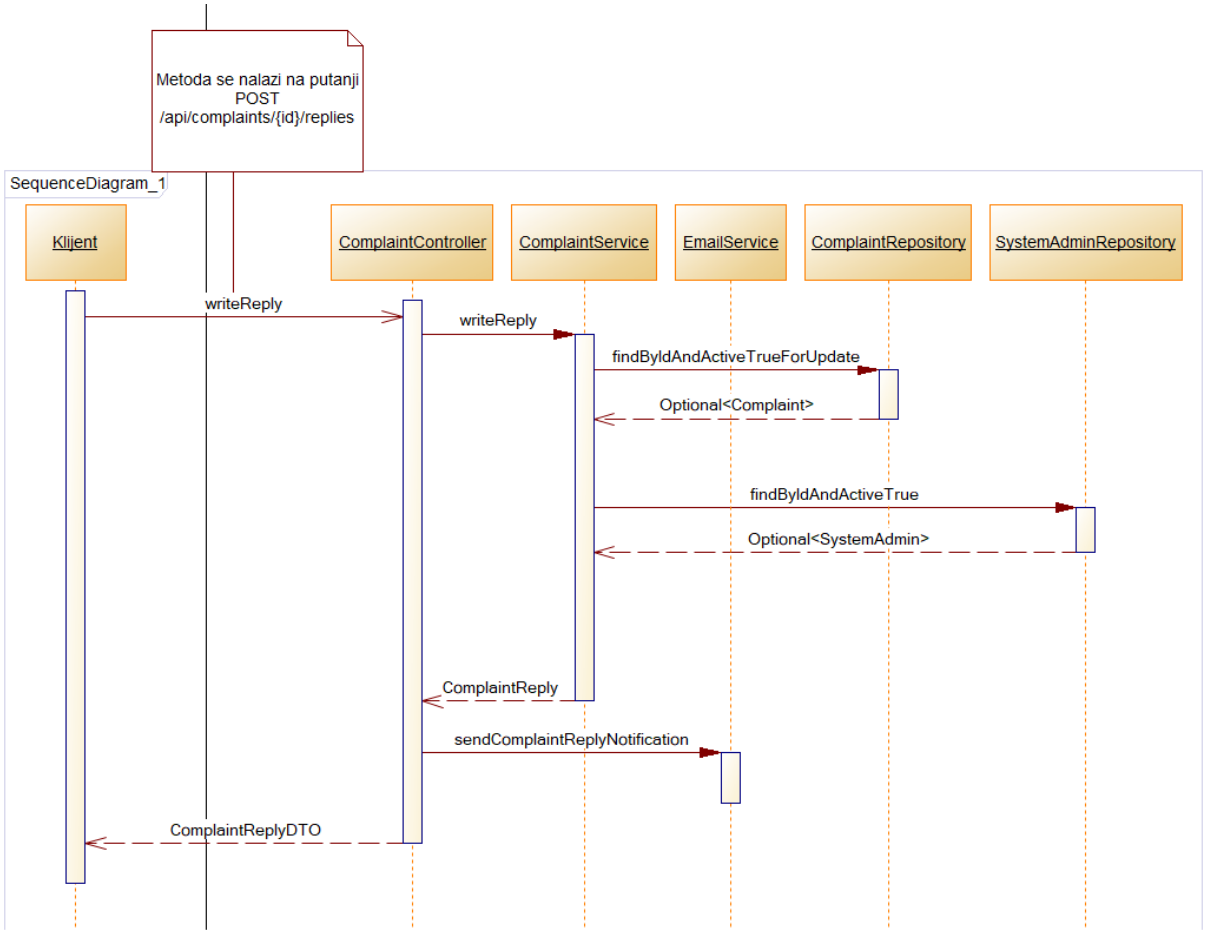
```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select c from Complaint c where c.active = true and c.id = :id")
Optional<Complaint> findByIdAndActiveTrueForUpdate(@Param("id") Long id);
```

Илустрација 4 Метода за добављање жељене жалбе у IComplaintRepository

```
@Override
@Transactional(isolation = Isolation.READ_COMMITTED, rollbackFor = ResponseStatusException.class)
public ComplaintReply writeReply(Long complaintId, Long adminId, String content) {
```

Илустрација 5 Метода за креирање одговора у ComplaintService

Ток извршавања је приказан на слици Илустрација 6.



Илустрација 6 Дијаграм секвенце за процес писања одговора на жалбу

3. АЖУРИРАЊЕ БОДОВА ЗА ТЕРМИНЕ

Овај сценарио је најмање вјероватан да се деси, јер његова учесталост не зависи од активности корисника, за разлику од прва два.

До конфликта може доћи када више администратора система истовремено покуша да ажурира број бодова за термине дерматолога и фармацеута.

Начин решавања:

Кориштен је приступ песимистичког закључавања. Овдје је могло бити кориштено и оптимистичко закључавање, с тим што предности тог приступа нису толико видљиве у овом случају, па се приступило решавању проблема по узору на претходна два сценарија.

Песимистичко закључавање је кориштено над методом *findByIdForUpdate* у *ISystemSettingsRepository* (Илустрација 8).

Сервисна метода гдје се врши само ажурирање, метода *updateSystemSettings* у *SystemSettingsService*, је анотирана анотацијом *Transactional*, са атрибутима *isolation=Isolation.READ_COMMITED* и *rollbackFor=ResponseStatusException.class* (Илустрација 7).

Резоновање које је довело до овог приступа је идентично оном из првог сценарија.

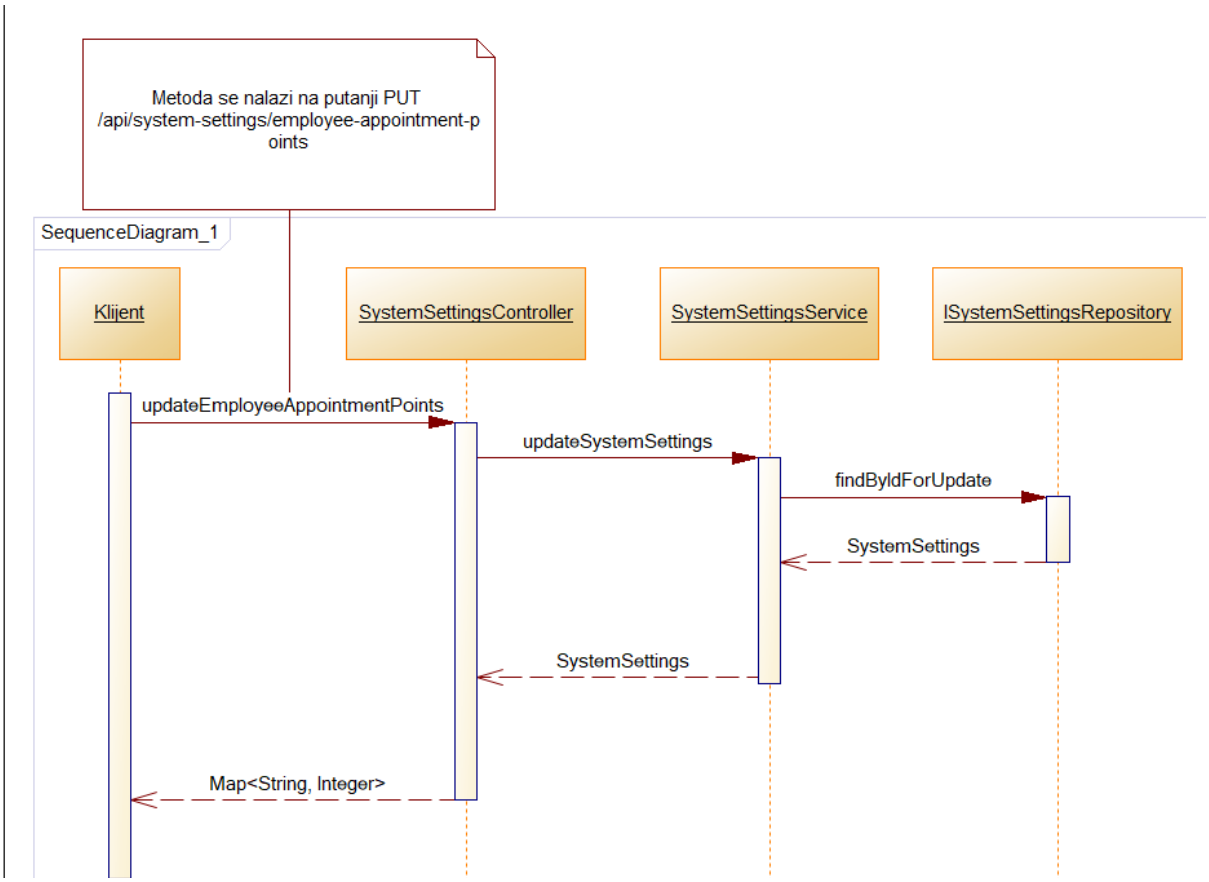
```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select ss from SystemSettings ss where ss.active = true and ss.id = :id")
SystemSettings findByIdForUpdate(@Param("id") Long id);
```

Илустрација 8 Метода за добављање жељене конфигурације

```
@Override
@Transactional(isolation = Isolation.READ_COMMITTED, rollbackFor = ResponseStatusException.class)
public SystemSettings updateSystemSettings(Integer dermatologistPoints, Integer pharmacistPoints) {
```

Илустрација 7 Метода за ажурирање бодова

Ток извршавања је приказан на слици Илустрација 9.



Илустрација 9 Дијаграм секвенце за поступак ажурирања поена