Решавање конфликтних ситуација – Студент 2

Ажурирање количине лека приликом прихватања понуде за наруџбеницу

Када админ апотеке прихвати неку понуду за своју поруџбеницу непоходно је коректно ажурирати колочину тог лека у апотеци. Уколико се не води рачуна о конкурентном приступу могло би да дође до ситуација да се количина лека неправилно ажурира. На пример, један корисник наручи 20 комада аспирина, а у истом тренутку админ апотеке прихвата понуду и тако додаје 200 комада на расположиво стање. Могућ резултат би, на пример, био да админ апотеке не прочита да је скинуто 20 комада аспирина са стања и само увећа количину за 200, тако да би укупна количина лека на стању сачувана у бази података била за 20 већа него његова реална количина.

Овај проблем је на нивоу апликације решен тако што се приликом добављања објекта класе која представља стање неког лека у некој апотеци (*MedicineStock*) ради песимистичко закључавање за читање. Овим ће све други нити које покушавају да приступе истом објекту (истом реду у табели) бити блокиране док га нит која га је прва заизела не ослободи. У коду је поменуто закључавање урађено на начин описан на Илустрација 1. Метода се налази у *IMedicineStockRepository*. Како сматрамо да је важно да се количина лека у апликацији увек исправно ажурира, определили смо се да на свим местима користимо наведену методу, те да увек радимо песимистичко закључавање приликом ажурирања стања лека.

```
@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 Песимистичко закључавање лека у апотеци

Сервисна метода у којој се позива закључана метода репозиторијума (Илустрација 2) је проглашена за трансакциону и то са подрузамеваном вредношћу за параметар који представља пропагацију, тако да се ова метода извршава у сколу једне веће трансакције. У потпису ове методе је специфицирано да се ради ролбек уколико дође до изузетка тупа *BusinessException*, што може бити случај ако неки лек који се налази у поруџбини није регистрован у тренутној апотеци. Ако у обзир узмемо ограничења имплементирана у самој апликацији, овака ситуација је могућа једино ручном изменом редова у бази података.

```
@Override
@Transactional(rollbackFor = BusinessException.class)
public void updatePharmacyStock(Long pharmacyId, Long orderId) {
```

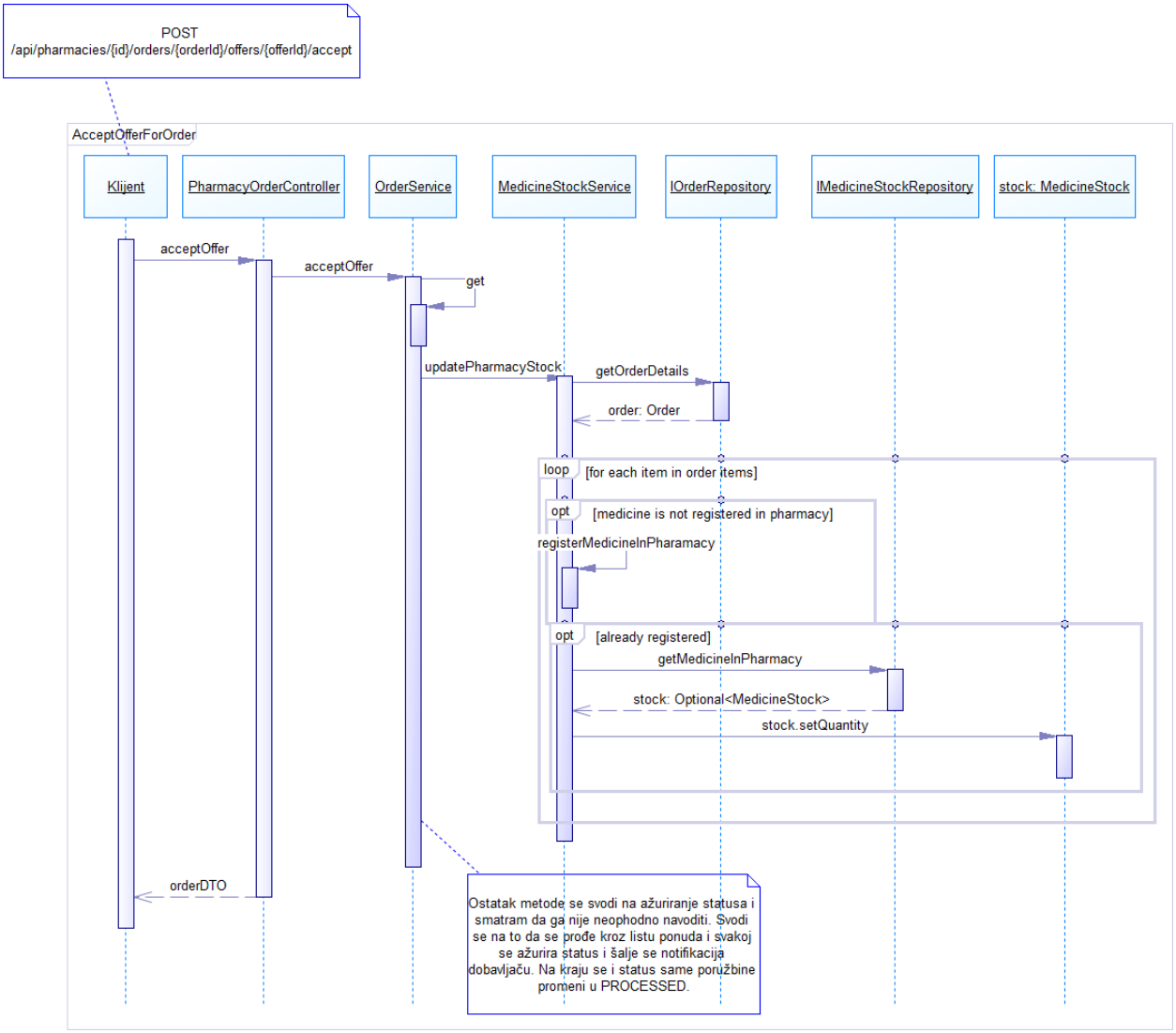
Илустрација 2 Потпис сервисне методе у којој се врши ажурирање количине лека

Као што је већ речено, метода *updatePharmacyStock* класе *MedicineStockService*, се извршава у оквиру трансакције чије границе превазилазе границе саме методе. Границе те трансакције чине границе методе *acceptOffer* класе *OrderService* (Илустрација 4). Ова метода не мора означена као трансакциона из разлога што је базна сервисна класа коју она наслеђује означена као трансакциона, а та анотација је попагирајућа на потомке.

```
@Override
public void acceptOffer(Long orderId, Long offerId) {
```

Илустрација 3 Поптипс методе која представља границе трансакције

Дијаграм секвенце на коме је предстаљен комплетан овај ток је приказан на Илустрација 3.



Илустрација 4 Дијаграм секвенце за ажурирање колилине лека приликом прихватања понуде за наруџбеницу

Креирање слбодних термина за фармацеута и дерматологе

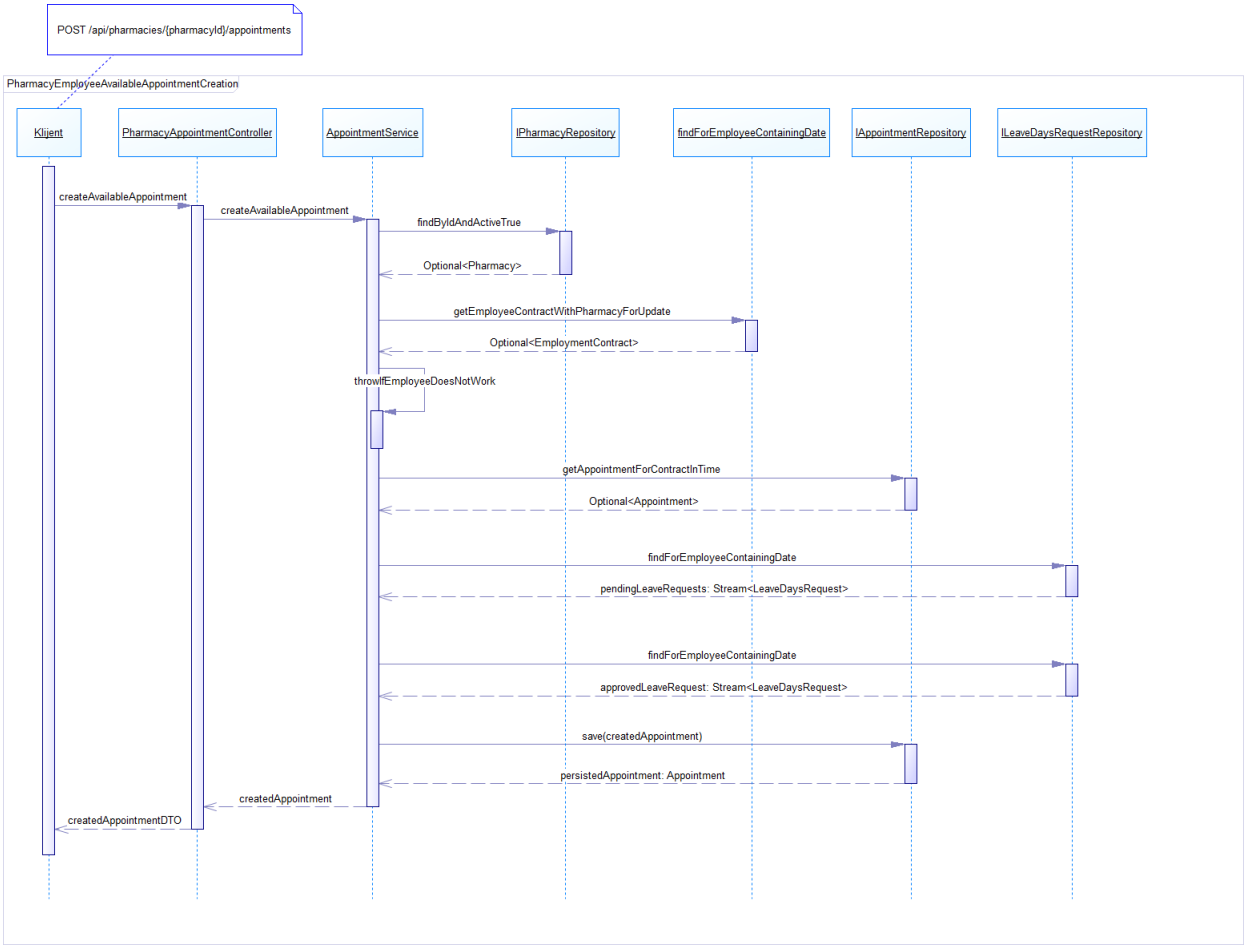
Како једном апотеком може да управља више администратора, приликом дефинисања слободних термина за запослене у апотеци (фармацеуте и дерматологе) не дође до ситуације да два админа апотеке дефинишу два различита термина за једног запосленог у исто време. Ако не бисмо водили рачуна о конкурентном приступу, до овакве ситуације би могло да дође уколико 2 трансакције нису свесне једна друге, односно трансакција А не би била свесна да је трансакција Б додала нови преглед у базу, те би приликом извршавања свог упита закључила да запослени у то време нема дефинисаних термина.

Овај проблем је решен тако што се приликом добављања објекта класе *EmploymentContract*, која представља уговор о запослењу запосленог у апотеци и која је неопходна за креирање објекта класе *Appointment*, ради песимистичко закључавање за читање. Овим се све друге нити које добављају исто запосленог у сврху креирања слободног термина блокирају. Анотирана метода *IEmploymentContractRepository* интерфејса је дата на Илустрација 5. Ова метода се позива у оквиру трансакције – сервисне методе *createAvailableAppointment* класе *AppointmentService.* Ова метода опет није експлицитно означена као трансакциона, јер је ту анотација наследила од базне сервисне класе.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select ec from EmploymentContract ec where ec.active=true and ec.pharmacyEmployee.id=:employeeId and ec.pharmacy.id=:pharmacyId")
Optional<EmploymentContract> getEmployeeContractWithPharmacyForUpdate(@Param("employeeId") Long employeeId,
                                                                       @Param("pharmacyId") Long pharmacyId);
```

Илустрација 5 Закључана метода за добављање запосленог у апотеци

Дијаграм секвенце прављења процеса прављења слободних термина дат је на . Како ми у апликацији не правимо разлику између термина код дерматолога и саветовања код фармацеута, све иде преко истог *endpoint-a*.



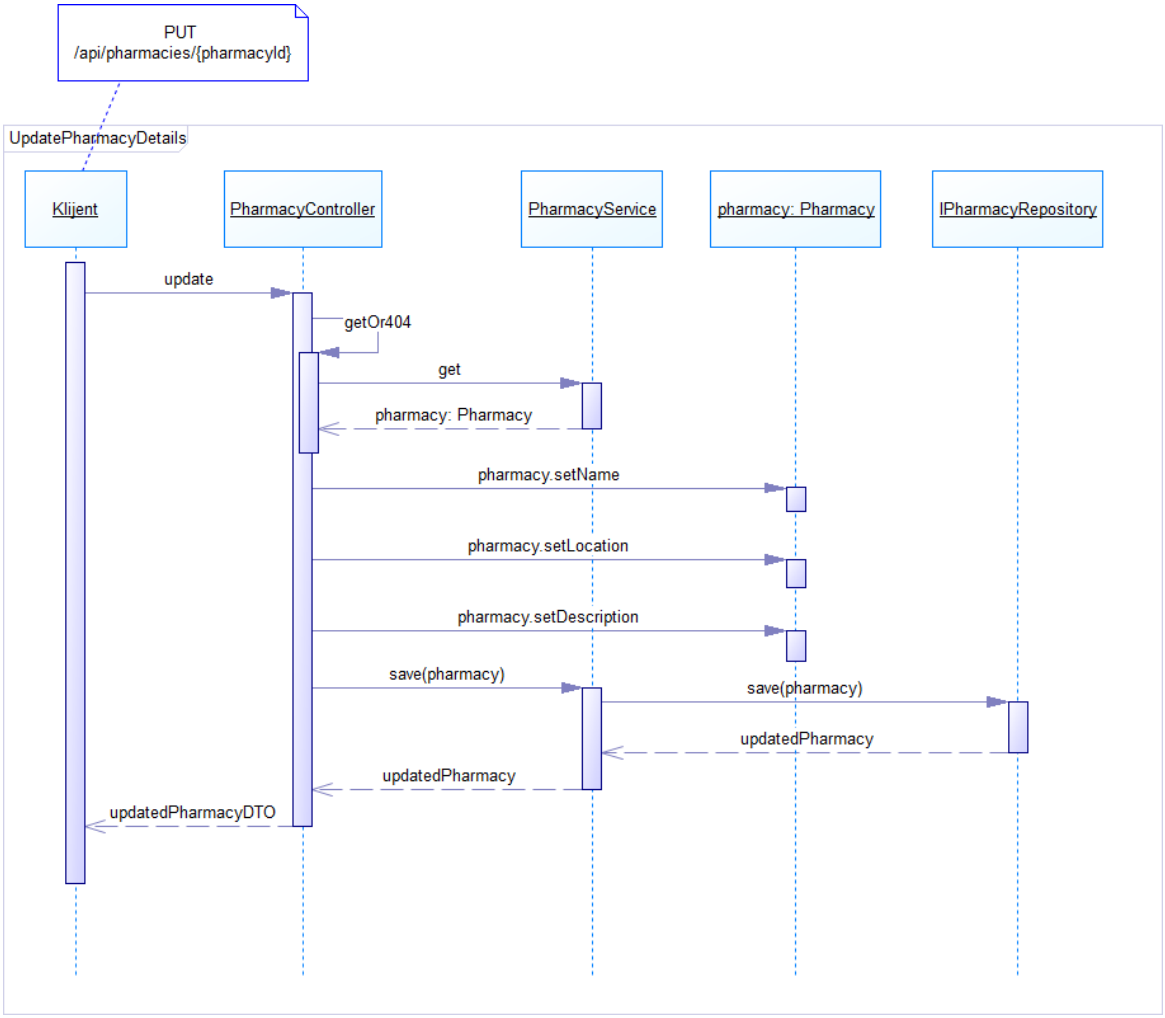
Илустрација 6 Дијаграм секвенце процеса дефинисања слободних термина код запослених у апореци

Ажурирање детаља о апотеци

Како једном апотеком управља више администратора, могућа је да два админа у исто време ажурирају детаље о апотеци. У том случају, измене админа чији је захтев први обрађен би биле прегажене изменама које други админ направи, с тим да други админ не би био свестан ни да су измене првог админа и постојале.

Овај проблем је решен применом оптимистичког закључавања. У класу *Pharmacy* је додато поље за верзију која се ажурира приликом сваког писања. Уколико сада админ А и админ Б покушавају да сачувају измене у исто време и рецимо да админ А први сачува измене, тада ће се ажурирати верзија реда и када админ Б покуша да сачува своје измене десиће се грешка. У случају грешке сам написао глобани *Exception handler* за тип изузетка *OptimisticLockingFailureException*, који ће вратити поруку да је у међувремену дошло до измене података. Овај проблем није толико критичан за решавање па сматрам да је овакав вид поруке прихватљив.

Дијаграм секвенце процеса ажурирања детаља о апотеци дат је на Илустрација 7.



Илустрација 7 Дијаграм секвенце процеса ажурирања детаља о апотеци

Одговарање на захтев за одсуство

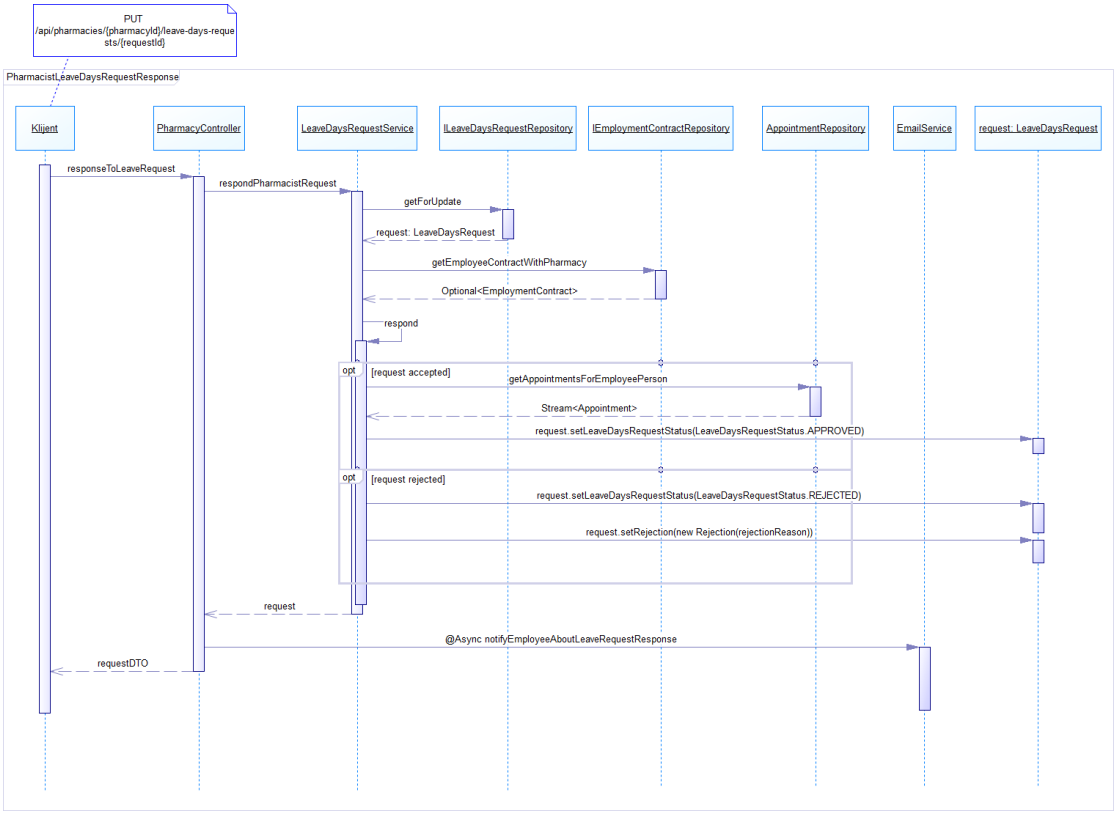
Фармацеути и дерматолози могу да поднесу захтев за одуство и на ове захтеве одговарају администратори апотека и администратори система, респективно. Могуће је да дође до ситуације да на један захтев одговори више администратора. Ово може бити проблем уколико администратори имају различит став о захтеву за одсуство – на пример, један администратор прихвати захтев, а други га одбије.

Како се у систему очекује да је број запослених у апотеци већи од броја администратора, ова ситуација не би требало често да се дешава. За решавање ове ситуације сам се определио за песимистичко закључавање зато што се на тај начин одмах види да је захтев већ обрађен. У случају оптимистичког закључавања би се морао извршити још један упит којим се проверава да ли запослени у том периоду има неки преглед да би се тек на крају трансакције видело да је захтев већ обрађен. Закључавање се ради на методи *getForUpdate* репозиторијума *ILeaveDaysRequestRepository* (Илустрација 8). Ова метода се позива у методи за одговор на захтев фармацеута и дерматолога, једино се разлију валидације које се раде након позива те методе.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select req from LeaveDaysRequest req where req.id=:requestId and req.active=true")
Optional<LeaveDaysRequest> getForUpdate(@Param("requestId") Long requestId);
```

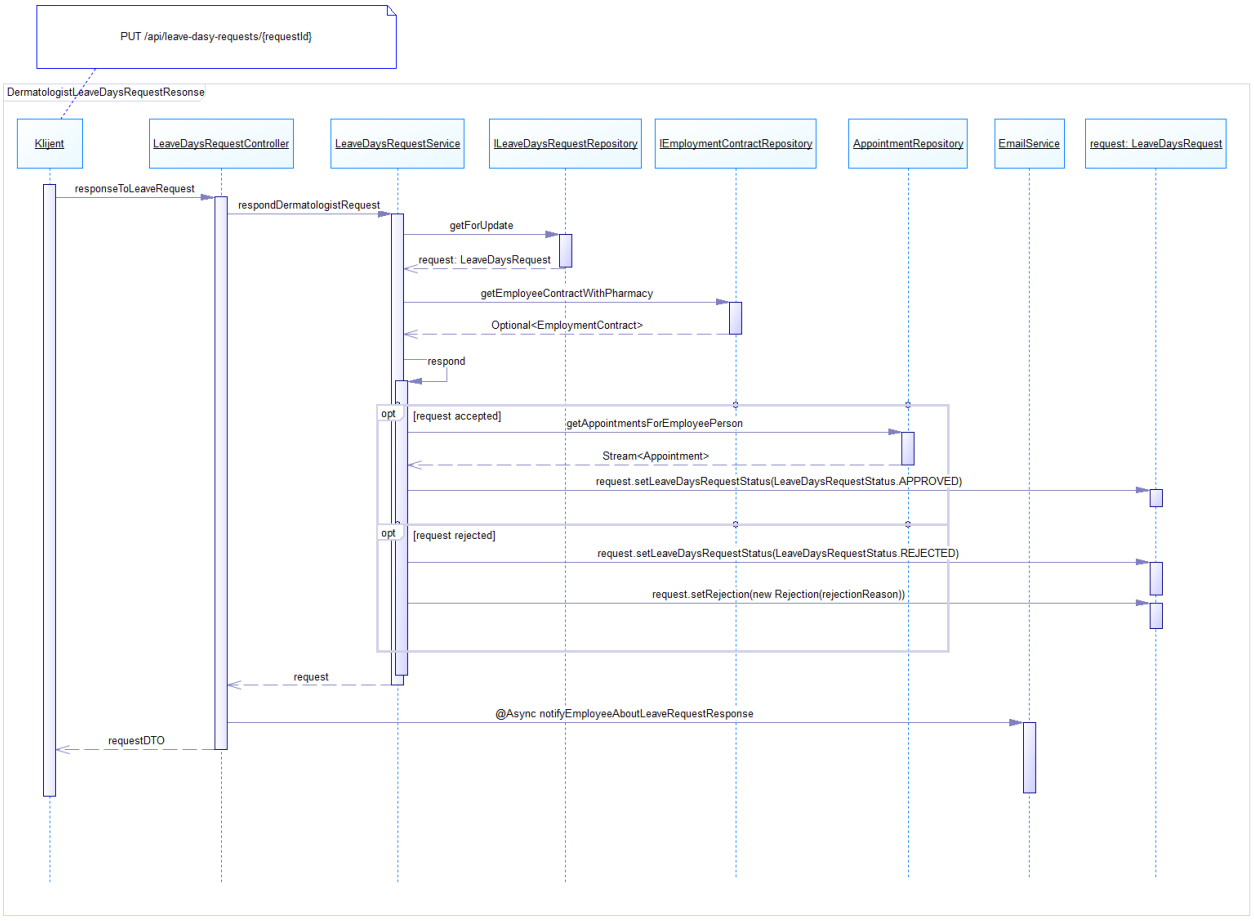
Илустрација 8 Закључана метода за добављање захтева за одсуство

Дијаграм процеса одговарања на захтеве фармацеута дат је на Илустрација 9.



Илустрација 9 Дијаграм секвенце процеса одговарања на захтев за одуство фармацеута

Дијаграм секвенце за процес одговарања на захтев за одсуство дерматолога дат је на Илустрација 10.



Илустрација 10 Дијаграм секвенце за процес одговарања на зтехв за одсуство дерматолога