Конфликтне ситуације студента 3

Уочени проблеми

1. Резервација лекова приликом подносења известаја о саветовању/прегледу од стране фармацеута/дерматолога
2. Издавање лекова од стране фармацеута

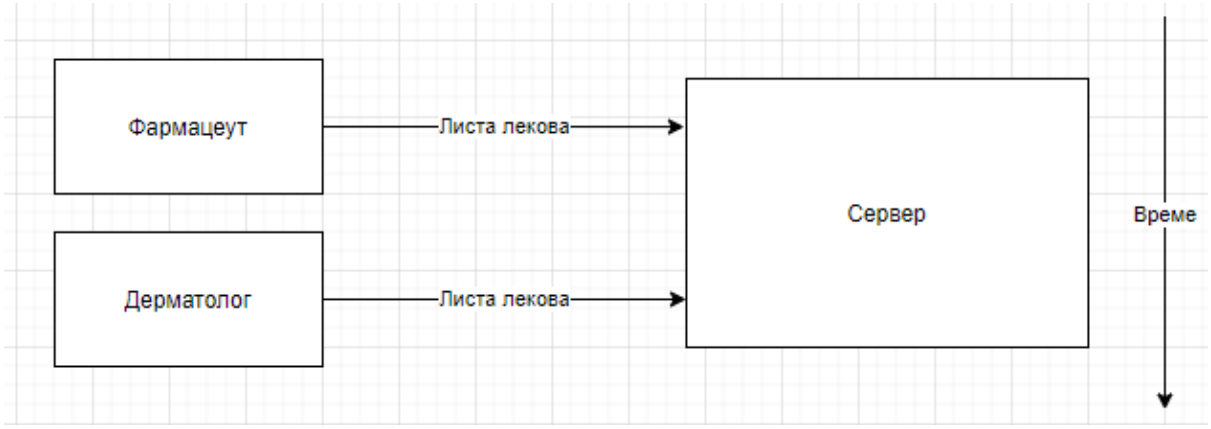
Ваљано понашање апликације при решавању ових проблема је устврђено одговарајућим тестовима који се могу пронаћи у тестној класи *DermatologFarmaceutTest*.

Резервација лекова приликом подношења извештаја о саветовању/прегледу од стране фармацеута/дерматолога

Опис проблема

Ако два фармацеута или дерматолога покрену чување извештаја прегледа, односно саветовања, у исто време, могуће је да се, уколико су препоручили исте лекове из исте апотеке, количина лекова не ажурира ваљано, него да се стање у апотеци ажурира само једном, од стране једног од процеса.

Тиме ће се обојици фармацеута/дерматолога потврдити да су успешно завршили своје извештаје, али се стање у апотеци није ажурирало исправно.



Конкретна приступна тачка која се гађа при овоме је */pregledi/updatePregled*, метода:

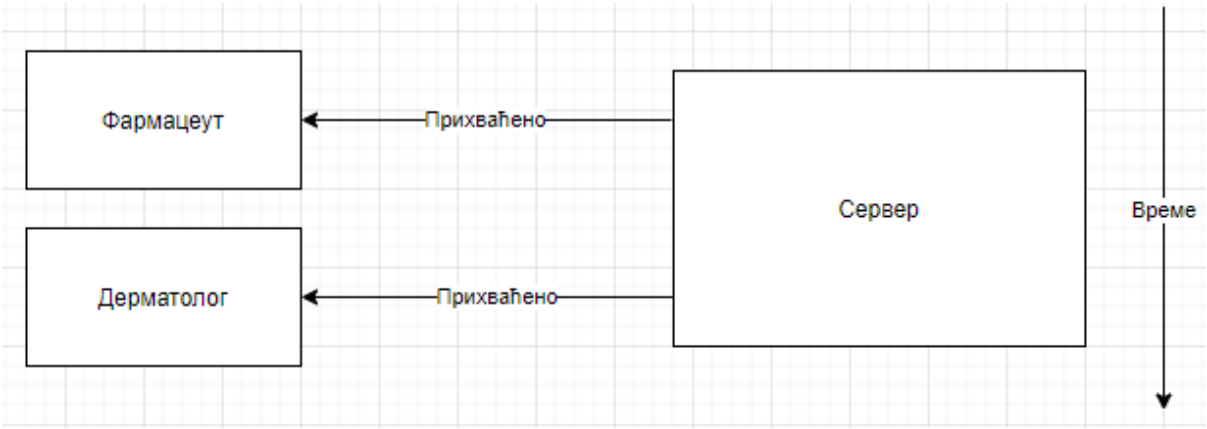
public ResponseEntity<String> updatePregled(@RequestBody PregledDTO pregledDTO) {…} класе *PregledController*.

Оба процеса ће узети исти објекат из базе, ажурирати количину и сачувати објекат. Сервер извршава следећи ток акција:

1. Нит 1: Учитавам објекте лекова из базе.
2. Нит 1: Ажурирам количину лекова над објектом.
3. Нит 2: Учитавам објекте лекова из базе.
4. Нит 2: Ажурирам количину лекова над објектом.
5. Нит 1: Чувам објекте у бази
6. Нит 2: Чувам објекте у бази.

При овом току акција, само нит 2 је успела да ажурира стање у бази, тако што је уписала своје измене преко нити 1. Сервер у том случају исписује и фармацеуту и дерматолгу

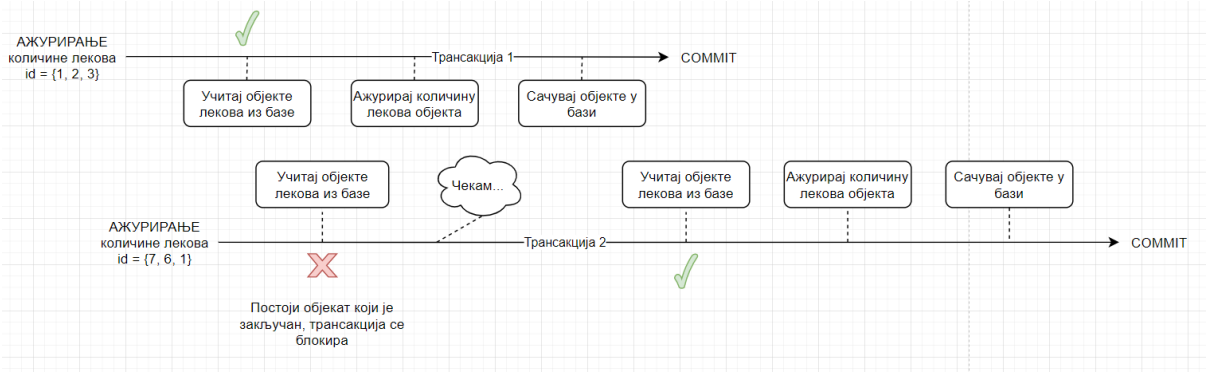
обавештење да су успешно резервисали лекове за своје пацијенте, иако можда неки од њих није био на стању након првог процеса.



Решење

Да би се проблем решио ефикасно, уводи се песимистично закључавање објеката. На овај начин, процес који први приступи објектима лекова ће исте закључати, тако да ће следећи процес морати да чека њихово отпуштање од стране првог процеса.

Трансакције се понашају на следећи начин:



Део кода из методе контролера у којему се позива трансакциони сервис:

```
//konkurentno odredi novu kolicinu
try {
    apotekaLekService.updateKolicinaLekovaKonkurentno(
pregled.getPreporuceniLekovi(), pregled.getApoteka().getId(), false);
} catch (LekNijeNaStanjuException e) {
    return new ResponseEntity<String>("Lek nije na stanju zbog konflikta!",
HttpStatus.BAD_REQUEST);
}
```

Као што можемо приметити, трансакцији се шаље листа лекова које је фармацеут/дерматолог означио као препоручене у извештају прегледа, односно саветовања, ID апотеке у коме се саветовање или преглед извршава, и ознака за тестирање, да ли ће трансакција чекати једну секунду.

Трансакциони сервис:

```
@Transactional(propagation = Propagation.REQUIRES_NEW, readOnly = false)
   public void updateKolicinaLekovaKonkurentno(Set<PregledLek>
preporuceniLekovi, Long idApoteke, boolean wait) throws
LekNijeNaStanjuException {
```

```
   Map<Long, Integer> kolicineLekova = new HashMap<>();

   for (PregledLek pregledLek : preporuceniLekovi) {
      kolicineLekova.put(pregledLek.getLek().getId(),
pregledLek.getKolicina());
   }

   List<ApotekaLek> apotekaLekovi =
apotekaLekRepository.findApotekaLekoviByIdWithLock(kolicineLekova.keySet(),
idApoteke);
   if (wait)
      try {
         Thread.sleep(1000);
      } catch (InterruptedException e) {
         e.printStackTrace();
      }

   for (ApotekaLek apotekaLek : apotekaLekovi) {
      if (kolicineLekova.get(apotekaLek.getLek().getId()) >
apotekaLek.getKolicina())
         throw  new LekNijeNaStanjuException();
      apotekaLek.setKolicina(apotekaLek.getKolicina() -
kolicineLekova.get(apotekaLek.getLek().getId()));
   }
   apotekaLekRepository.saveAll(apotekaLekovi);
}
```

Трансакциони сервис најпре врши упит у коме добавља лекове које треба резервисати, након чега им ажурира количину и чува објекте.

Упит **findApotekaLekoviByIdWithLock** је анотиран са:

@Lock(LockModeType.PESSIMISTIC_WRITE)

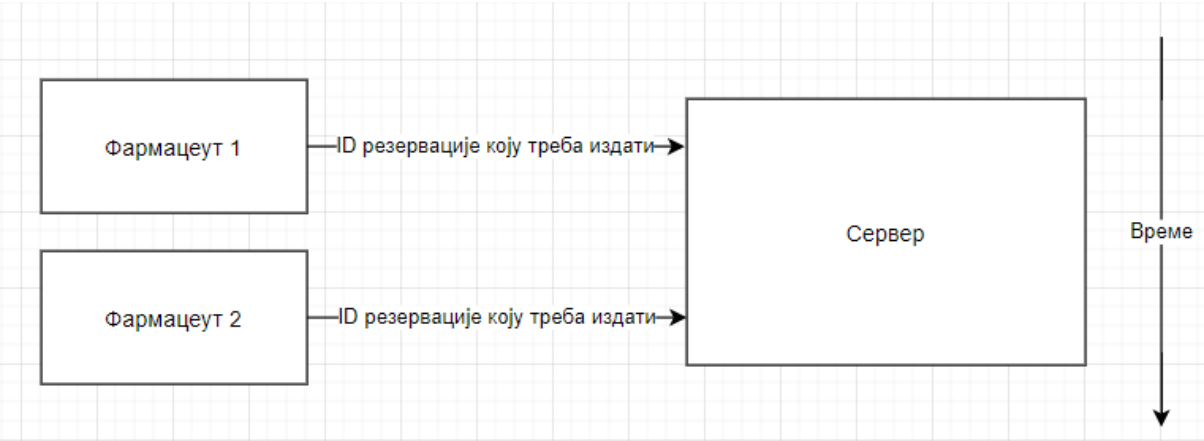
То значи да ће се објекти враћени датим упитом закључати, док ће друга трансакција која их потражује ће чекати на њихово откључавање, без обзира да ли их отвара само за читање или за писање.

Издавање лекова од стране фармацеута

Опис проблема

Уколико два фармацеута покрену акцију за издавање лекова исте резервације у исто време, могуће је да дође до ситуације где се обе резервације потврде као издате, и оба фармацеута ће у том случају продати лекове, тако да ће се лекови уделити два пута за једну резервацију. Ово врло штетно понашање апликације ће омогућити да лекови нестану из магацина без претходно да прођу кроз систем апликације.

Обојици фармацеута ће сервер потврдити да су издали лекове, док би заправо један од њих требао да добије поруку да су лекови већ издати.



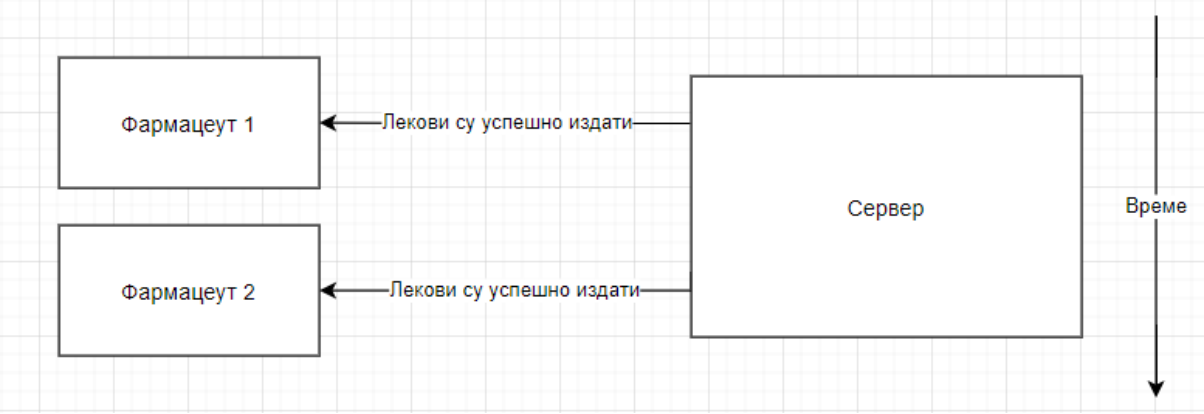
Конкретна приступна тачка која се гађа при овоме је */rezervacije/izdajLekove*, метода:

public ResponseEntity<String> izdajLekove(@RequestParam String cookie, @RequestParam Long idRezervacije) {…} класе *RezervacijaController*.

Слично као што је ситуација у првом проблему, оба процеса ће узети исти објекат из базе, ажурирати статус резервације и означити да је преузета, и сачувати објекат. Сервер извршава следећи ток акција:

1. Нит 1: Учитавам објекат резервацију.
2. Нит 1: Ажурирам статус објекта резервације.
3. Нит 2: Учитавам објекат резервацију.
4. Нит 2: Ажурирам статус објекта резервације.
5. Нит 1: Чувам објекат у бази
6. Нит 2: Чувам објекат у бази.

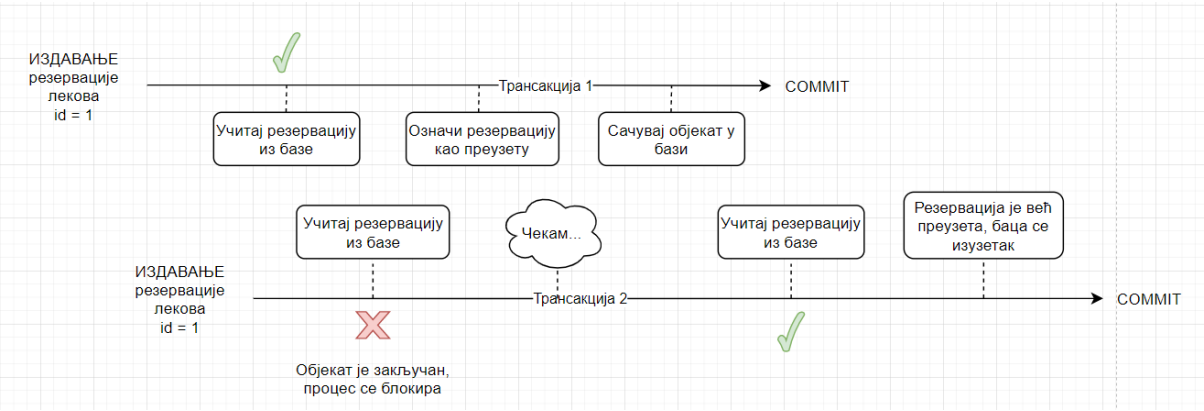
При овом току акција, само нит 2 је успела да ажурира стање у бази, тако што је уписала своје измене преко нити 1. Сервер у том случају исписује обојици фармацеута обавештење да су успешно издали лекове, иако то не би требало да буде могуће, већ треба једном да се испише порука да је издао, а другом да је резервација већ издата.



Решење

Исто као код првог проблема, да би се проблем решио ефикасно, уводи се песимистично закључавање објекта. На овај начин, процес који први приступи резервацији ће је закључати, тако да ће следећи процес морати да чека њено отпуштање од стране првог процеса.

Трансакције се понашају на следећи начин:



Део кода из методе контролера у којему се позива трансакциони сервис:

```
try {
   r = rezervacijaService.obradiRezervacijuKonkurentno(idRezervacije,
         f.getApoteka().getId(), LocalDateTime.now().plusDays(1), false);
} catch (RezervacijaNeispravnaException e) {
   return new ResponseEntity<>("Rezervacija neispravna",
HttpStatus.BAD_REQUEST);
}
```

Као што можемо приметити, трансакцији се шаље ID резервације, ID апотеке у коме су резервисани лекови, сутрашњи дан (за проверу истека резервације) и ознака за тестирање, да ли ће трансакција чекати једну секунду.

Трансакциони сервис:

```
@Transactional(propagation = Propagation.REQUIRED)
public Rezervacija obradiRezervacijuKonkurentno(Long idRezervacije, Long
idApoteke, LocalDateTime tommorow, boolean wait) throws
RezervacijaNeispravnaException {

    Rezervacija r =
rezervacijaRepository.findRezervacijaByIdAndApotekaIdBeforeRok(
idRezervacije, idApoteke, tommorow);

    if(r==null)
        throw new RezervacijaNeispravnaException();
    r.setPreuzeto(true);

    if (wait)
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

    rezervacijaRepository.save(r);
    return r;
}
```

Трансакциони сервис најпре врши упит у коме добавља резервацију коју треба издати, након чега проверава да ли је она већ преузета, па је преузима уколико није, а баца изузетак ако јесте.

Упит **findRezervacijaByIdAndApotekaIdBeforeRok** је анотиран са:

@Lock(LockModeType.PESSIMISTIC_WRITE)

То значи да ће се објекти враћени датим упитом закључати, док ће друга трансакција која их потражује ће чекати на њихово откључавање, без обзира да ли их отвара само за читање или за писање.