Конфликтне ситуације студента 1

Уочени проблеми

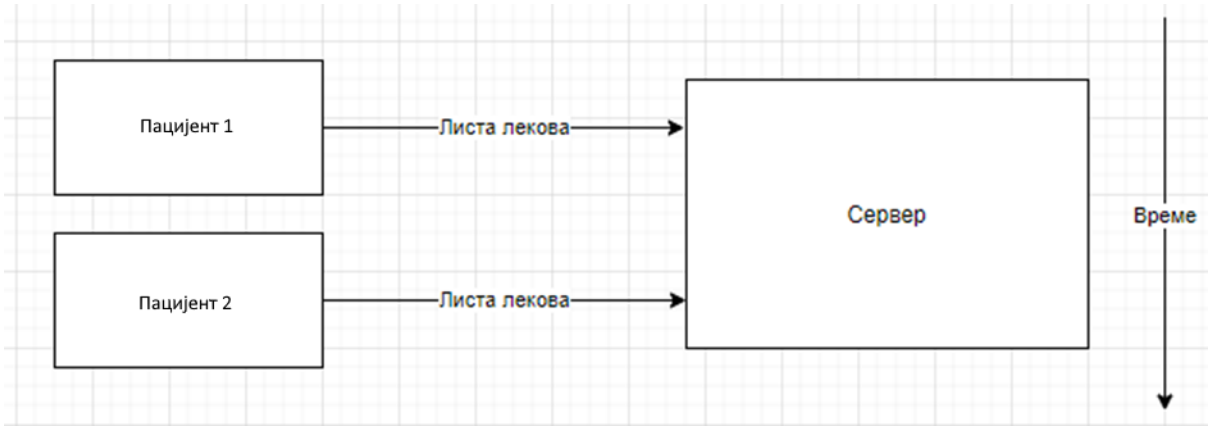
1. Резервација лекова од стране пацијента
2. Отказивање резервисаних лекова од стране пацијента
3. Резервисање дефинисаних прегледа код дерматолога од стране пацијента
4. Креирање термина саветовања код слободног фармацеута од стране пацијента
5. Регистрација пацијента на систем

Резервација лекова од стране пацијента

Опис проблема

Ако два различита пацијента резервишу листу лекова у исто време, и уколико су препоручили исте лекове из исте апотеке, количина лекова се не ажурира ваљано, него да се стање у апотеци ажурира само једном, од стране једног од процеса.

Тиме ће се пацијентима потврдити да су успешно завршили своје извештаје, али се стање у апотеци није ажурирало исправно.



Уколико резервишемо један лек из прегледа свих лекова приступна тачка која се гађа је */rezervacije/rezervacija-leka* у RezervacijaController-у, метода:

```
@GetMapping(value="/rezervacija-leka")
public ResponseEntity<Void> receiveData(
            @RequestParam("sifra") Long sifra,
            @RequestParam("kolicina") int kolicina,
            @RequestParam("istekRezervacije") String datum,
            @RequestParam("cookie") String cookie)
            throws ParseException, InterruptedException {…}
            {
```

Или уколико резервишемо више различитих лекова из једне апотеке гађа се /rezervacije/pacijent , метода:

```
@PostMapping(value="/pacijent")
public ResponseEntity<List<RezervacijaDTO>> dodajRezervaciju (
            @RequestBody List<RezervacijaDTO> rezervacije)
            throws InterruptedException {…}
```

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

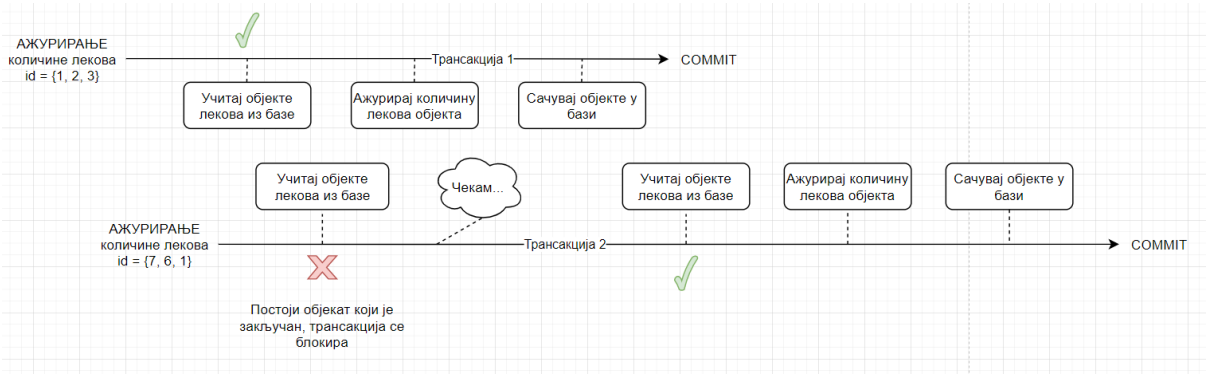
1. Нит 1: Учитавам објекте лекова из базе.
2. Нит 1: Ажурирам количину лекова над објектом.
3. Нит 2: Учитавам објекте лекова из базе.
4. Нит 2: Ажурирам количину лекова над објектом.
5. Нит 1: Чувам објекте у бази
6. Нит 2: Чувам објекте у бази.

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

Решење

Да би решили проблем уводимо 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:

```
try {
      apotekaLekService.updateKolicinaSlobodnihLekovaKonkurentno(lekIDs,kolicine);
} catch (LekNijeNaStanjuException e) {
            return new ResponseEntity<>(HttpStatus.CONFLICT);
}
```

Метода прослеђује листу лекова и њихових количина у трансакциони сервис, уз то уколико дође до конфликта методу смо окружили try-catch blokom da to uhvati.

Трансакциони сервис:

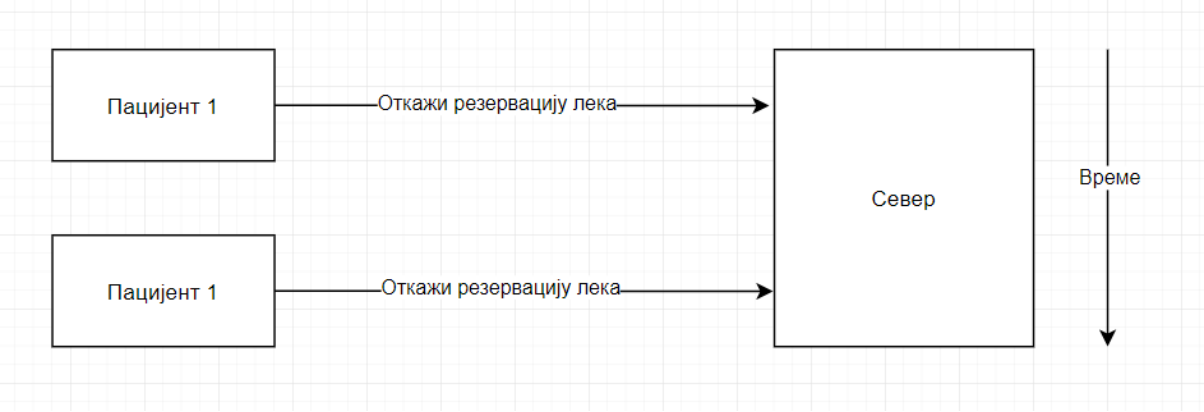
```
@Transactional(propagation = Propagation.REQUIRES_NEW, readOnly = false)
public void updateKolicinaSlobodnihLekovaKonkurentno(ArrayList<Long> lekIDs, ArrayList<Integer> kolicine) throws LekNijeNaStanjuException, InterruptedException{
    Map<Long, Integer> kolicineLekova = new HashMap<>();

    int i = 0;
    for (Long id : lekIDs) {
        kolicineLekova.put(id, kolicine.get(i));
        i++;
    }

    List<ApotekaLek> apotekaLekovi = apotekaLekRepository.findAllApotekaLekByIDWithLock(kolicineLekova.keySet());

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

    Thread.sleep(2000);

    apotekaLekRepository.saveAll(apotekaLekovi);
}
```

Трансакциони сервис најпре спаја лекове и њихове количине у мапу, потом закључавамо све АпотекаЛекове које пацијент жели тренутно резервисати из базе са *@Lock(LockModeType.PESSIMISTIC_WRITE)* анотацијом. 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. Пре самог чувања објекта имамо Thread.sleep() како би тестирали наше решење.

Отказивање резервисаних лекова од стране пацијента

Опис проблема

Врло сличан претходном проблему јесте проблем да уколико два пацијента откажу резервисане лекове у исто време, и при томе је у питању исти лек, доће до конфликта где ће се за оба пацијента приказати да је лек отказак, док ће у суштини количина лекова на стању бити ажурирана само једном.



Конкретна приступна тачка која се гађа при овоме је */rezervacije/otkazi-rezervaciju*, метода:

```
@GetMapping(value="/otkazi-rezervaciju")
public ResponseEntity<Void> otkaziRezervaciju(@RequestParam String id_rezervacije)
throws InterruptedException {…}
```

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

1. Нит 1: Учитавам објекате лекова из базе.
2. Нит 1: Ажурирам количину објекта лека.
3. Нит 2: Учитавам објекате лекова из базе.
4. Нит 2: Ажурирам количину објекта лека.
5. Нит 1: Чувам објекат у бази.
6. Нит 2: Чувам објекат у бази.

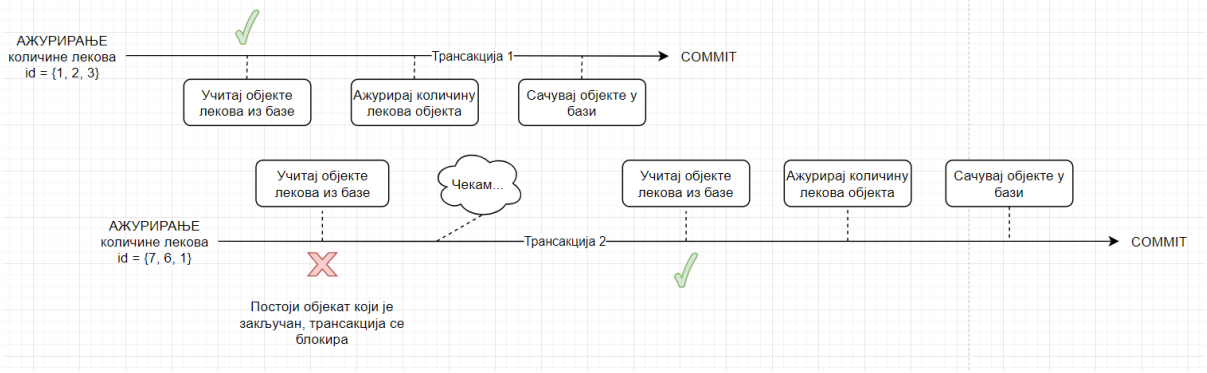
При овом току акција, само нит 2 је успела да ажурира стање у бази, тако што је уписала своје измене преко нити 1. Сервер у том случају исписује обавештење пацијентима да су успешно отказали резервисане лекове, иако можда неки од њих није био на стању након првог процеса.

Решење

Исто као код првог проблема, да би се проблем решио ефикасно, уводи се песимистично закључавање објекта. На овај начин, процес који први приступи отказивању ће

исти објекат и закључати, тако да ће следећи процес морати да чека њено отпуштање од стране првог процеса.

Трансакције се понашају на следећи начин:



Као што видимо овај проблем је врло сличан првом конкурентном проблему.

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

```
// zakljucavamo kolicinu
try {
        apotekaLekService.updateKolicinaSlobodnihLekovaKonkurentno(lekIDs, kolicine);
} catch (LekNijeNaStanjuException e) {
    return new ResponseEntity<>(HttpStatus.CONFLICT);
}
```

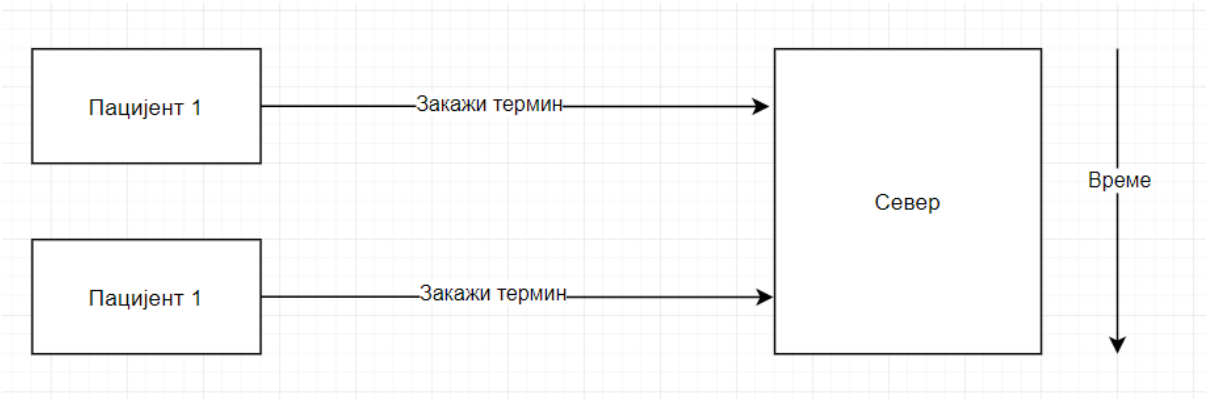
Као што можемо видети користимо исту конкурентну методу као и из првог случаја. Према томе нема потребе приказивати трансакциони сервис.

Резервисање дефинисаних прегледа код дерматолога од стране пацијента

Опис проблема

Ако два различита пацијента закажу исти термин код дерматолога, тај термин се неће ажурирати како треба, већ ће се термин заказати само једном од два процеса.

Тиме ће се пацијентима потврдити да су успешно завршили заказивање термина, али се стање тог термина неће ажурирати исправно.



Уколико закажемо термин приступна тачка која се гађа је */pregledi/zakaziPregled* у PregledController-у, метода:

```
@GetMapping(value="/zakaziPregled")
public ResponseEntity<Void> otkaziRezervaciju(
            @RequestParam("id_pregleda") Long id_pregleda,
            @RequestParam("cookie") String cookie)
            throws InterruptedException {…}
```

Сервер извршава следећи ток акција:

1. Нит 1: Учитавам објекат прегледа из базе.
2. Нит 2: Учитавам објекат прегледа из базе.
3. Нит 1: Проверавам да ли је стање прегледа заказано
4. Нит 1: Ажурирам стање прегледа у заказан.
5. Нит 2: Проверавам да ли је стање прегледа заказано
6. Нит 2: Ажурирам стање прегледа у заказан.
7. Нит 1: Чувам објекат у бази.
8. 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 обавештење пацијентима да су успешно заказали термин, иако један од њих није био на стању након првог процеса.

Решење

Ове такође уводимо песимистично закључавање објеката. На овај начин, процес који први приступи објекту термина ће исти закључати, тако да ће следећи процес морати да чека његово отпуштање од стране првог процеса.

Трансакције се понашају на следећи начин:

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

```
// zakljucavamo pregled
try {
        pregled = pregledService.zakaziPregledKonkurentno(id_pregleda, p);
} catch (PregledRezervisanException e) {
    return new ResponseEntity<>(HttpStatus.CONFLICT);
}
```

Метода прослеђује индентификатор прегледа који покушавамо да закажемо као и инстанцу пацијента која заказује тај преглед, уз то уколико дође до конфликта методу смо окружили try-catch blokom da to uhvati.

Трансакциони сервис:

```
@Transactional(propagation = Propagation.REQUIRES_NEW, readOnly = false)
public Pregled zakaziPregledKonkurentno(Long idPregleda, Pacijent p) throws InterruptedException, PregledRezervisanException {

    Pregled pregled = pregledRepository.findOneByIdWithLock(idPregleda);

    Thread.sleep(2000);

    if(pregled.isPregledZakazan()) {
        throw new PregledRezervisanException();
    }

    pregled.setPacijent(p);
    pregled.setPregledZakazan(true);
    pregledRepository.save(pregled);

    return pregled;
}
```

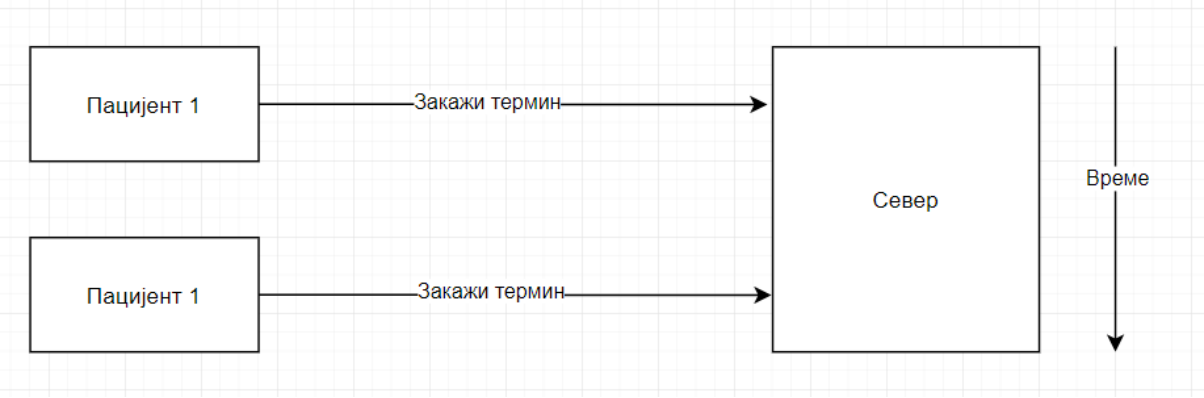
Трансакциони сервис прво закључава преглед у бази са *LockModeType.PESSIMISTIC_WRITE* анотацијом. То значи да ће се објекти враћени датим упитом закључати, док ће друга трансакција која их потражује ће чекати на њихово откључавање, без обзира да ли их отвара само за читање или за писање. Потом имамо Thread.sleep() како би тестирали наше решење. После тога проверавамо да ли је преглед већ заказан, уколико јесте бацамо exception kako bi obavestili pacijenta. Na samom kraju čuvamo novo stanje pregleda kao i pacijenta koji ga je zakazao.

Kreiгање термина саветовања код слободног фармацеута од стране пацијента

Опис проблема

Ако два различита пацијента закажу саветовање код истог фармацеута у исто време, том фармацеуту ће бити додељена оба саветовања која се преклапају.

Тиме ће се пацијентима потврдити да су успешно завршили заказивање термина, али се стање тог термина неће ажурирати исправно.



Уколико закажемо саветовање приступна тачка која се гађа је */pregledi/createSavetovanje* у PregledController-у, метода:

```
@GetMapping(value = "/createSavetovanje")
public ResponseEntity<Void> createSavetovanje(
            @RequestParam("start") String startDate,
            @RequestParam("end") String endDate,
            @RequestParam("cookie") String cookie,
            @RequestParam("idFarmaceuta") Long idFarmaceuta,
            @RequestParam("idApoteke") Long idApoteke)
throws InterruptedException {…}
```

Сервер извршава следећи ток акција:

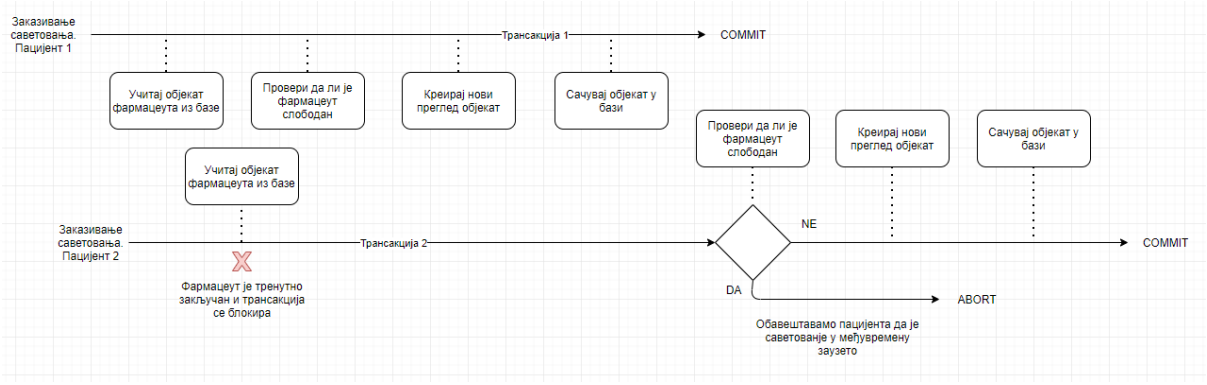
1. Нит 1: Учитавам објекат фармацеута из базе.
2. Нит 2: Учитавам објекат фармацеута из базе.
3. Нит 1: Проверавам да ли је фармацеут заузет.
4. Нит 1: Креирам саветовање.
5. Нит 2: Проверавам да ли је фармацеут заузет.
6. Нит 2: Креирам саветовање.
7. Нит 1: Чувам нови објекат у бази.
8. Нит 2: Чувам нови објекат у бази.

При овом току акција, два различита пацијента креираће два саветовања код истог фармацеута, без обзира да ли је он био заузет у то време.

Решење

Ове такође уводимо песимистично закључавање објеката. На овај начин, процес који први провери да ли је фармацеут заузет ће истог и закључати, тако да ће следећи процес морати да чека његово отпуштање од стране првог процеса.

Трансакције се понашају на следећи начин:



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

```
// pessimistic save
try {
    p = pregledService.savePregledAndCheckIfFarmacistsIsFreeConcurent(pac, idApoteke, idFarmaceuta, start, end);
} catch (FarmaceutZauzetException e) {
    return new ResponseEntity<>(HttpStatus.CONFLICT);
}
```

Методи прослеђујемо пацијента, индентификаторе апотеке и фармацеута као и почетак и крај саветовања.

Трансакциони сервис:

```
@Transactional(propagation = Propagation.REQUIRES_NEW, readOnly = false)
public Pregled savePregledAndCheckIfFarmacistsIsFreeConcurent(Pacijent pac, Long idApoteke, Long farmaceut,LocalDateTime start, LocalDateTime end) throws FarmaceutZauzetException, InterruptedException{

    ZdravstveniRadnik zdravstveniRadnik = (ZdravstveniRadnik) zdravstveniRadnikService.findOneByIdWithLock(farmaceut);
    Apoteka a = apotekaService.findOne(idApoteke);

    if (!findAllInDateRangeByZdravstveniRadnik(start, end, zdravstveniRadnik.getCookieTokenValue()).isEmpty()) {
        throw new FarmaceutZauzetException();
    }

    Pregled p = new Pregled();
    p.setVreme(start);
    p.setKraj(end);
    p.setPregledZakazan(true);
    p.setPregledObavljen(false);
    p.setApoteka(a);
    p.setPacijent(pac);

    if (zdravstveniRadnik instanceof Farmaceut)
        p.setCena(a.getCenaSavetovanja());

    p.setZdravstveniRadnik(zdravstveniRadnik);


    Thread.sleep(2000);

    pregledRepository.save(p);

    return p;
}
```

Трансакциони сервис прво песимистично закључава здравственог радника, потом проверавамо да ли је закључани фармацеут слободан у предвиђеном термину. Уколико јесте креирамо нови термин и чувамо га у бази.