Конфликтне ситуације студента 2

Уочени проблеми

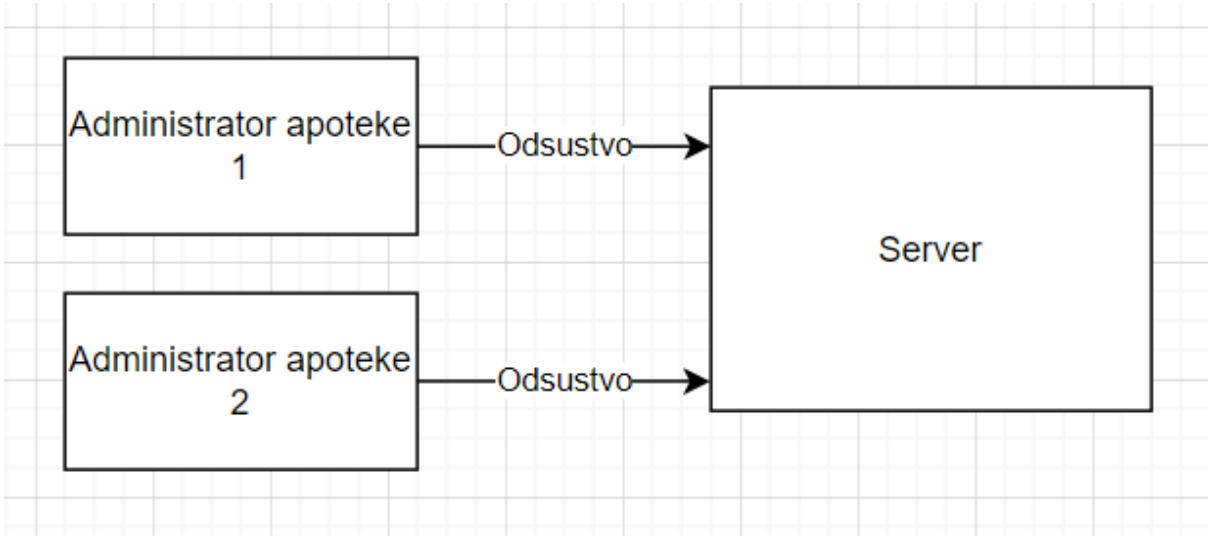
1. Одбијање/одобрење одсуства за фармацеуте као администратор апотеке

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

1. Одбијање/одобрење одсуства за фармацеуте као администратор апотеке

Опис проблема

Ако два администратора апотеке покушају у исто време да одбији или одобре исто одсуство фармацеута, настаће ситуација у којој ће фармацеут добити два мајла са обавештењем о том одсуство и још ако су одговори разлити може доћи до неспоразума. Обојици администратора апотеке ће се потврдити успешно послата информација о одсуству, иако само један треба да прође.



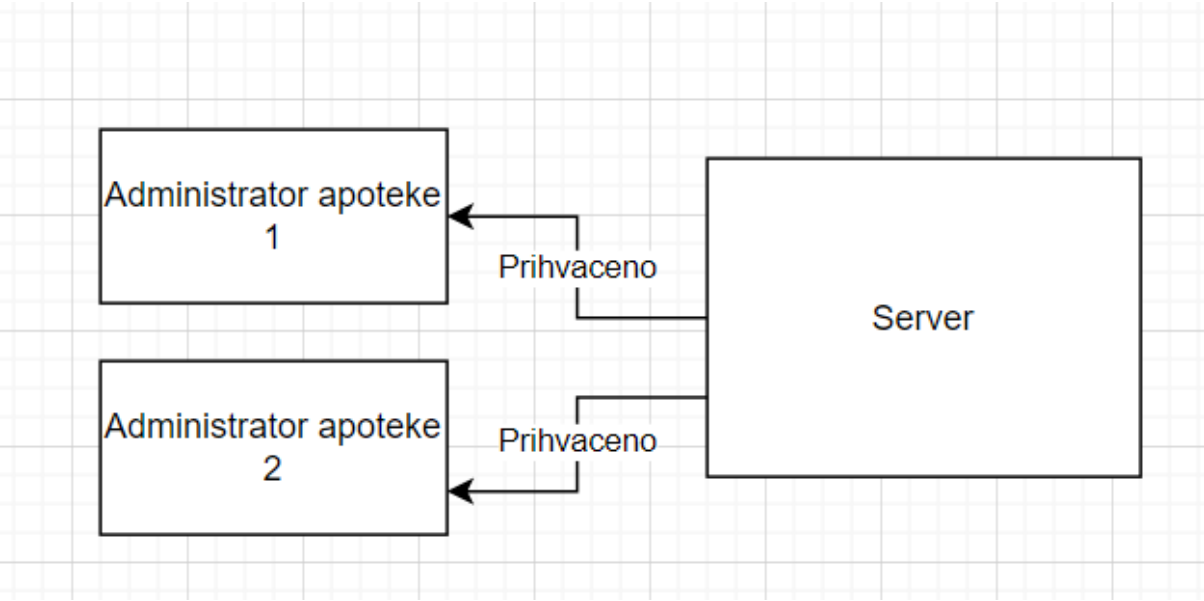
Конкретна приступна тачка која се гађа при овоме је */zdravstveniradnik/updateOdsustvo*, метода:

public ResponseEntity<Void> updateOdsustvo(@RequestBody OdsustvoDTO odsustvoDTO){..} класе *ZdravstveniRadnikController*.

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

1. Нит 1: Учитавам објекта одсуство из базе.
2. Нит 1: Ажурирам статус објекта одсуства.
3. Нит 2: Учитавам објекте лекова из базе.
4. Нит 2: Ажурирам статус објекта одсуства.
5. Нит 1: Чувам објекат у бази
6. Нит 2: Чувам објекат у бази.

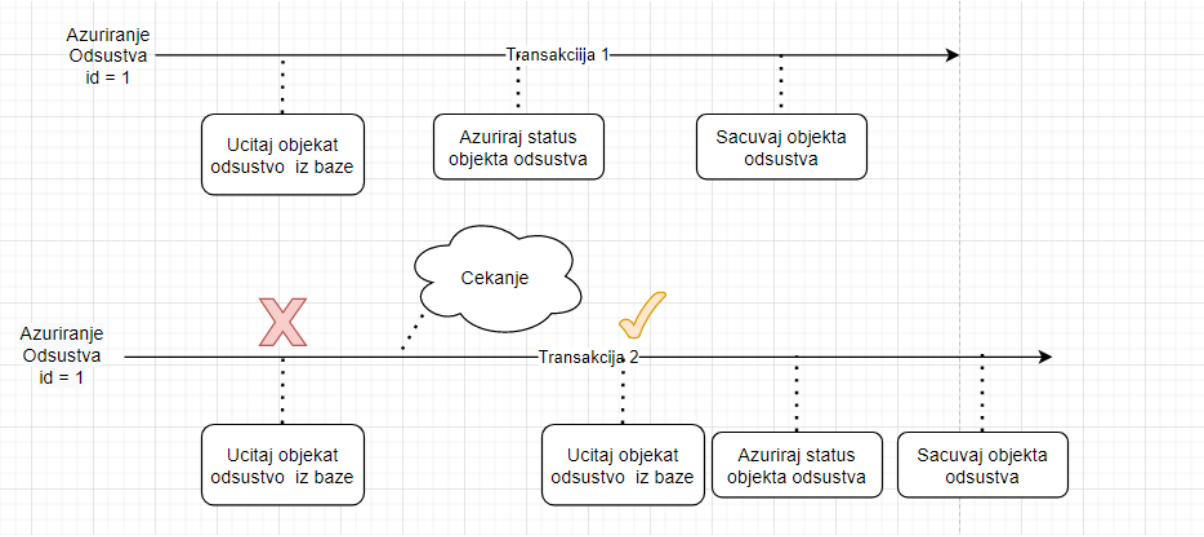
При овом току акција, само нит 2 је успела да ажурира стање у бази, тако што је уписала своје измене преко нити 1. Сервер у том случају исписује и фармацеуту и дерматолгу обавештење да су успешно изменили статус одсуства.



Решење

Да би се проблем решио ефикасно, уводи се песимистично закључавање објеката. На овај начин, процес који први приступи објекту одсуство 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:

```
Odsustvo o;

try{
      o = odsustvoService.saveOdsustvoConc(odsustvoDTO.getId(),
      odsustvoDTO.getStatus());
   }
catch(OdsustvoException e) {
        return new ResponseEntity<Void>(HttpStatus.BAD_REQUEST);
        }
```

Као што можемо приметити, трансакцији се шаље ид одсуства и статус који желимо да поставимо на изабрано одсуство лекова.

Трансакциони сервиси:

```
	@Transactional(readOnly = false)
	public Odsustvo saveOdsustvoConc(Long id,String status) throws OdsustvoException {
		Odsustvo odsustvo = findOneConc(id);
		if(!odsustvo.getStatus().equals("cekanju")) {
			throw new OdsustvoException();
		}

		odsustvo.setStatus(status);
		odsustvoRepository.save(odsustvo);
		return odsustvo;
	}
```

Трансакциони сервис најпре врши упит у коме добавља одсуство које треба да ажурира, након чега ажурира статус одсуства и чува објекте.

```
	@Transactional(readOnly = false)
	public Odsustvo findOneConc(Long id) {
		return odsustvoRepository.findOneByIdConc(id);
	}
```

Изнад можемо видети упит за добављање одсуства конкуретно.

Упит **`findOneByIdConc`** 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. Овиме смо спречили проблем који је изнад објашњен.