Internet softverske arhitekture – Konkurentni pristup bazi

Branislav Lazarević IN22/2017 –Administrator apoteke

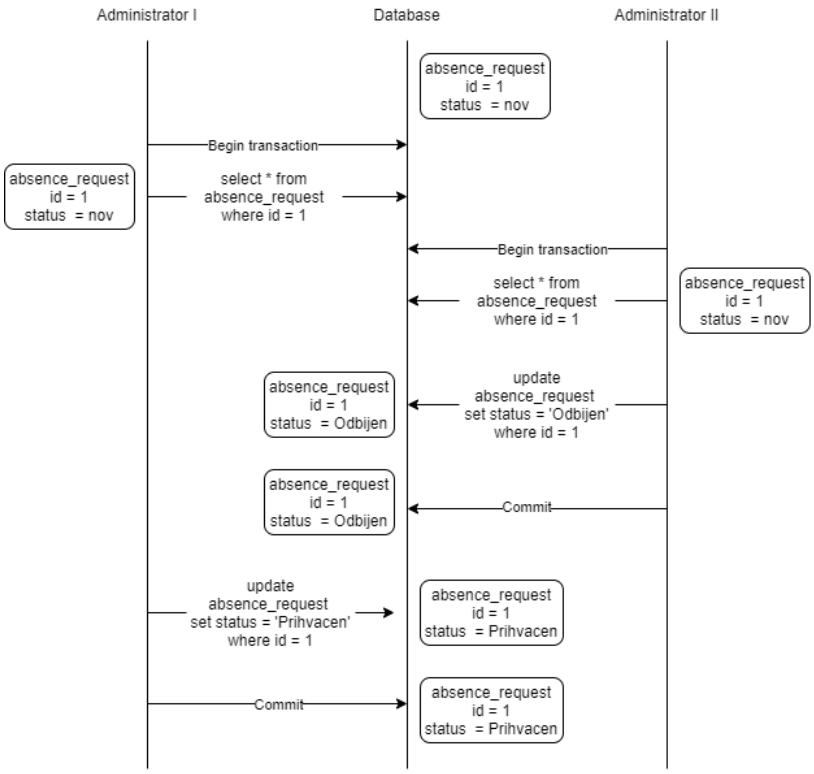
1. Odgovor na zahtev za godišnji odmor

Opis funkcionalnosti:

Farmaceut apoteke šalje zahtev za godišnji odmor za određen vremenski interval kada neće moći da ima zakazana savetovanja. Administrator apoteke daje odgovor na taj zahtev odnosno potvrđuje ili odbija zahtev za godišnji odmor. Zahtev se zatim upisuje u bazu i ukoliko je prihvaćen farmaceutu se ne mogu zakazati savetovanja u periodu odmora. Nakon rešavanja zahteva farmaceutu se takođe šalje mejl.

Problem:

Apoteka može da ima više administratora apoteke i svaki od njih može da vidi pristigle zahteve za odmor. U takvom okruženju moguće je da više administratora istovremeno pošalju odgovor, koji može biti i različit, i tako doći do štetnih preplitanja niti koje menjaju stanje sistema što dovodi do nekonzistentnosti baze podatka i našeg sistema. Takođe se po završetku čuvanja i slanja odgovora šalje mejl zaposlenom da li je njegovo odsustvo prihvaćeno ili odbijeno u slučajima preplitanja dolzi do slanja više mejlova za isti zahtev koji opet mogu biti različiti i dovesti do dodatne konfuzije kod radnika.



Rešenje:

Kako bi se ovaj problem izbegao odlučeno je da prvi poslat zahtev za odgovor na zahtev bude onaj koji će biti prihvaćen i zabeležen. To je omogućeno pesimističkim zaključavanjem reda u bazi koji odgovara tom zahtevu i provere da li je na taj zahtev odgovoreno. Ukoliko nije funkcija dalje nastavlja sa radom i izvršava se po planu. Ukoliko drugi administrator takođe proba istovremeno to da uradi, to neće uspeti pošto je red u bazi zaključan. Pregledanje zahteva za odmore nije česta operacija i neće biti problem ukoliko se redovi zaključavaju da bi se upisao odgovor i iz tog razloga je odabran ovaj pristup.

Endpoint: /absence/answer

```
@Transactional(readOnly = false)
public AbsenceRequest getAbsenceForUpdate(Long id){
    return absenceRequestRepository.findOneById(id);
}

@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public AbsenceRequest answer(AbsenceDTO absenceDTO) {
    Long adminID = getAdminID();
    AbsenceRequest absenceRequest = getAbsenceForUpdate(absenceDTO.getId());
    if(!absenceRequest.getStatus().equals("nov")){
        throw new ResourceConflictException(1L,"Promenjen u medjuvremenu!");
    }
    absenceRequest.setStatus(absenceDTO.getStatus());
    if(absenceDTO.getStartDate().equals("Odbijeno")){
        absenceRequest.setAnswerText(absenceDTO.getAnswerText());
        mailService.absenceDeclinedNotification(absenceRequest);
    }
    else {
        mailService.absenceAcceptedNotification(absenceRequest);
    }
    absenceRequest.setAdminId(adminID);
    absenceRequestRepository.save(absenceRequest);
    return absenceRequest;
}
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select ar from AbsenceRequest ar WHERE ar.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
AbsenceRequest findOneById(@Param("id")Long id);
```