Konkurentni pristup resursima u bazi

1.Odgovaranje na zahteve za godišnji odmor

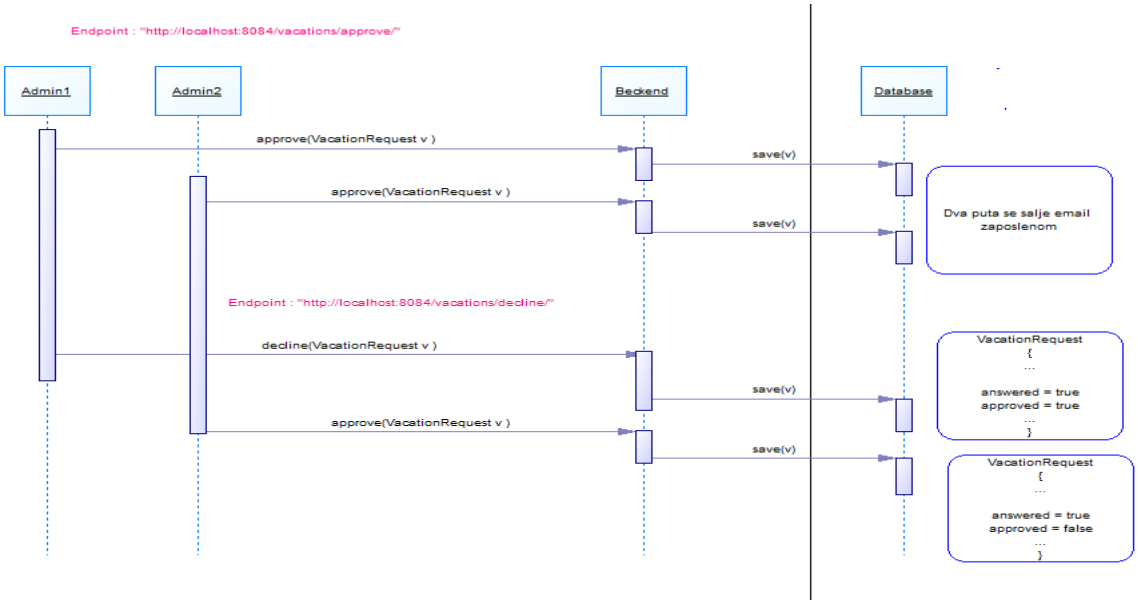
Dermatolog i farmaceut apoteke mogu da traže zahtev za godisnji odmor koji onda potvrđuje ili odbija administrator apoteke.

Opis situacije koja dovodi do konflikta

U apoteci je zaposlen veći broj administratora i može da se desi da vise administratora pokušava istovremeno da odgovori na isti zahtev za odmor.

U tom slučaju može doći do nekonzistentosti sistema, jer jedan administrator može da potrvdi odmor, a drugi da obije.

Tokovi zahteva klijenta I odgovori servera koji dovode do kofliktne situacije



Rešenje

Za realizaciju ovog rešenja korišćen je pristup Optimistic Locking.

Optimistic Locking je metoda za rešavanje problema konkurentnosti. Zasniva se na principu da se transakcije izvršavaju normalno bez zaključavanja podataka. Svaka transakcija proverava da li je neka druga transakcija menjala podatak koji se čita u trenutnoj transakciji. Ako je napravljena izmena, transakcija se prekida tj. poziva se Optimistic Lock Exeption.

Konkretno za pomenuti primer, kada jedan administrator pošalje odgovor na zahtev za godišnji odmor, polje “answer” klase vacationRequest će se postaviti na vrednost “true”. Ukoliko tada drugi administrator apoteke pokuša da odgovori na isti zahtev, vratiće mu se poruka o tome da je odgovor već poslat i transakcija će se prekinuti.

Optimistic Locking metoda je primerena za ovakvu situaciju iz razloga što ne dovodi do zaključavanja podataka i u slučajevima kada imamo mnogo zahteva, administratori neometano mogu da šalju odgovore, a da pritom konzistentnost sistema bude zadovoljena.

Implementacija opisanog rešenja

```
@Override
public Object approve(VacationRequest v )
{
    VacationRequest vacationRequest = vacationRepository.findById(v.getId()).get();

    if( vacationRequest.isAnswered() )
    {
        throw new OptimisticLockException();
    }
    v.setApproved(true);
    v.setAnswered(true );
    return vacationRepository.save(v);
}

@Override
public Object decline(VacationRequest v )
{
    VacationRequest vacationRequest = vacationRepository.findById(v.getId()).get();
    if( vacationRequest.isAnswered() )
    {
        throw new OptimisticLockException();
    }

    v.setApproved(false);
    v.setAnswered(true);
    return vacationRepository.save(v);
}
}
```

2.Kreiranje slobodnih termina za pregled kod dermatologa

Administrator apoteke ima mogućnost da kreira slobodne termine za pregled kod dermatologa.

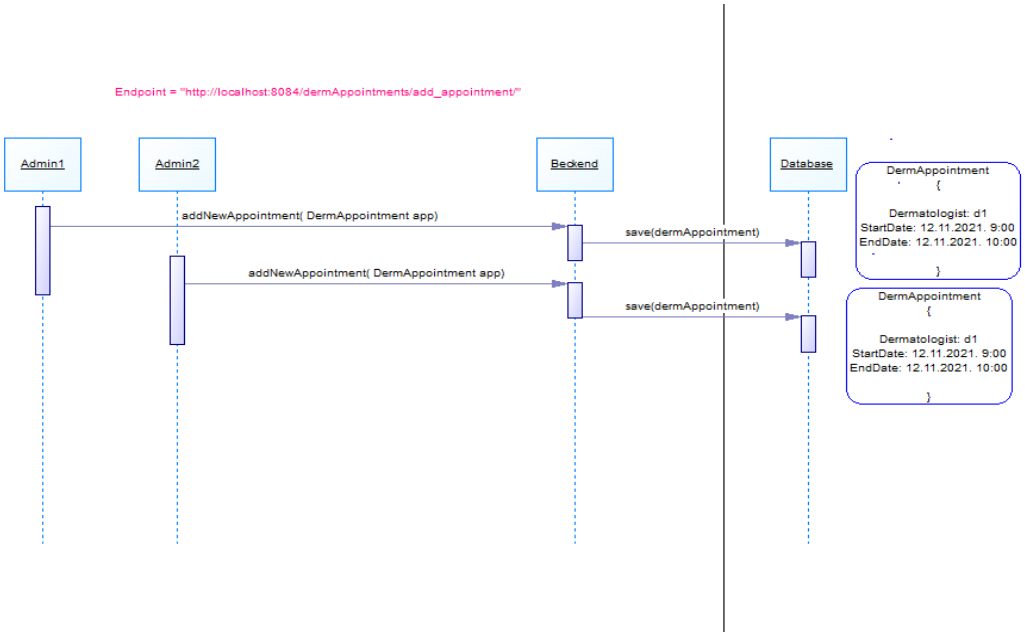
Preduslov za ovakvu akciju je da termin pregleda u celosti obuhvata radno vreme dermatologa. Nisu na raspolaganju ni datumi kada je dermatolog na godišnjem odmoru. Takodje, ne smeju da se preklapaju slobodni termini za istog dermatologa.

Opis situacije koja dovodi do konflikta

Dva administratora apoteke u isto vreme pokušavaju da definisu slobodne termine kod istog dermatologa i termini obuhvataju isti vremenski interval.

Jedan dermatolog ne može istovremeno da bude prisutan na dva pregleda.

Tokovi zahteva klijenta I odgovori servera koji dovode do kofliktne situacije



Rešenje

Za realizaciju ovog rešenja korišćen je pristup Pessimistic Locking. Ideja je da ukoliko drugi korisnik proba da menja podatke istovremeno sa prvim, transakcija će biti onemogućena, jer su zaštićeni podaci u bazi zaključani. Budući da je ovde kritičan momenat kada administratori pokušavaju da zakažu pregled kod istog dermatologa, zaštićeni podatak će biti dermatolog. Ukoliko jedan administrator pristupa konkretnom dermatologu, drugi administrator to neće moći.

Implementacija opisanog rešenja

```
public interface DermatologistRepository extends JpaRepository<Dermatologist,Long> {

    @Lock(LockModeType.PESSIMISTIC_WRITE)
    @Query("select w from Dermatologist w where w.id = :id")
    Dermatologist findOneForUpdate(@Param("id") Long id);
}
```

3.ODGOVARANJE NA PONUDE OD STRANE DOBAVLJAČA

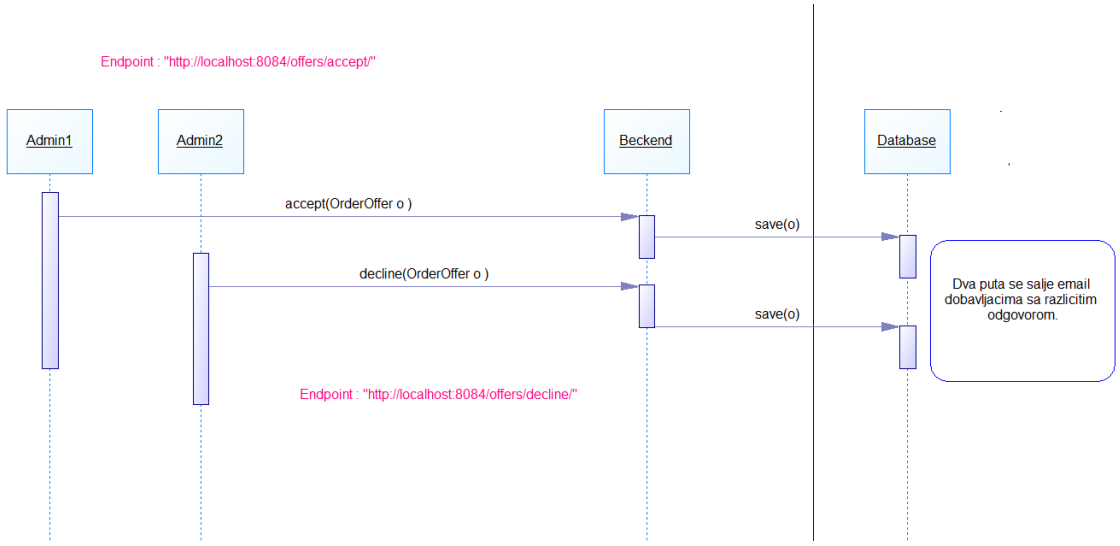
Administrator apoteke piše narudžbine za lekove koji su neophodni apoteci u kojoj je zaposlen. Ponudu kreira na način da odabere listu svih lekova koje želi I količinu svakog od lekova. Nakon sto kreira narudžbinu, dobavljači šalju ponude. Administrator ima pristup svim pristiglim ponudama za sve narudžbine koje je kreirao I može da prihvati neku od njih. Kada

odabere konkretnu ponudu, ona postaje potvrđena, a sve ostale pristigle ponude za istu narudžbinu će postati odbijene.

Opis situacije koja dovodi do konflikta

Konfliktna situacija koja može nastati je da više administratora istovremeno pokuša da odgovori na ponude za istu narudžbinu. U takvoj situaciji može nastati konfuzija oko toga koje ponude su potvrđene, a koje odbijene i nastaje nekonzistentnost sistema. Takođe, može da se desi da administrator odbije konkretnu ponudu i pošalje email dobavljaču sa obavestenjem da ponuda nije primljena, a u isto to vreme drugi administrator da potrvdi ponudu i ponvo pošalje email, sada sa potvrdnim odgovorom.

Tokovi zahteva klijenta I odgovori servera koji dovode do kofliktne situacije



Rešenje

Za realizaciju ovog rešenja takodje je korišćen pristup Optimistic Locking koji je prethodno opisan.

Implementacija opisanog rešenja