Konfliktne situacije u ISA projektu

Ovaj pdf objasnjava nekoliko konfliktnih situacija koje sam resio. Sadrži opis situacija i nacin na koji je situacija resena. Na kraju dokumenta je crtez tokova zahteva koji dovode do ovih problema.

Zadatak po specifikaciji mi je bio da realizujem funkcionalnosti I razvijem drugacije tipove korisnika vezane za studenta 2. Ovo se vecim delom svodi na implementaciju funkcija koje mogu da obavljaju doctor I administrator klinike (u tekstu **clinicAdmin**). Stoga su se konfliktne situacije koje sam resio većim delom svodile na CRUD operacije sa podacima kojima rukovode.

Bitan deo aplikacije je I rad oko pregleda. Nekoliko funkcionalnosti se odnosilo na rezervisanje I iscitavanje pregleda, pa se I tu pojavilo nekoliko konfliktnih situacija.

Odobravanje Zahteva Za Operaciju

Zadatak

Omoguciti da, prilikom odobravanja zahteva za operaciju/pregled, ne može jedna sala da bude rezervisana u isto vreme za različite operacije/preglede.

Opis Situacije

ClinicAdmin ima pred sobom stranicu za dodljeivanje nove sobe I datuma za pregled. Klikom na dugme ClinicAdmin potvrdjuje novu sobu I datum (I opciono doktora).

Salje se zahtev serveru (poziva se metoda **saveRoomAndDate** ili **saveRoomAndDateAndDoctor** iz ClinicAdminController-a u zavisnosti od toga koja je opcija izabrana). Server vadi iz baze listu svih pregleda sa izabranom sobom I datumom (I doktorom). Ako je lista prazna, update-uje se pregled.

U Cemu je Problem?

ClinicAdmin ca1 update-uje **pregled mc1**, a **ClinicAdmin ca2** update-uje **pregled mc2** par milisekundi posle njega. Pritom se igrom slucaja desilo da I **ca1** I **ca2** zele da update-uju svoje preglede tako da budu u isto vreme I u istoj sobi (sa istim doktorom).

ca1 iz baze izvadi praznu listu, sto znaci da ne postoji pregled sa tim vremenom I datumom (I doktorom). Par milisekundi posle njega, **ca2** takodje izvuce praznu listu iz baze. **ca1** update-uje **mc1**. Posto ca2 nije nasao nijednu zauzetu sobu, on par milisekundi posle njega

sacuva **mc2**. Obojica zavrse update-ovanje I obojica su sacuvali svoje preglede.

Problem je u tome sto sada u bazi postoje dva razlicita pregleda, koji su rezervisani istog datuma, u isto vreme, u istoj sobi (I mozda cak sa istim doktorom). Posto je **ca2** pokupio listu zauzetih soba **pre nego sto je ca1 update-ovao sobu**, on nije nasao nijednu zauzeti pregled. Da je **ca2** pokusao da nadje zauzeti pregled nakon sto je **ca1** svoj update-ovao, pronasao bi **mc1** I ne bi mogao tog dana da rezervise.

Kako je Resen Problem?

Problem je resen upotrebom **@Transactional** anotacije. Ona omogucuje da, izmedju ostalog, izolujemo jednu transakciju izmedju server I klijenta. U zavisnosti od nivoa izolacije, mi mozemo izolovati do te mere da “zakljucamo” podatke dok se trenutna transakcija ne zavrsi.

Potpuno zakljucavanje transakcije se postize stavljanjem **@Transactional(isolation=Isolation.SERIALIZABLE)** anotacije iznad metode kojoj nece moci nijedan drugi klijent da pristupi u toku njenog izvrsavanja.

Jedan lekar na vise razlicitih pregleda/operacija

Zadatak

Omoguciti da jedan lekar ne može istovremeno da bude prisutan na više različitih operacija.

Opis Situacije

ClinicAdmin pravi novi pregled koji pacijenti mogu da rezervisu jednim klikom. Bira datum, vreme, duzinu trajanja pregleda, cenu, tip pregleda, sobu, doktora I unosi komentar. Klikom na dugme potvrdjuje sve sto je uneo I salje se zahtev za registrovanje novog pregleda.

Na serveru se proverava da li doctor tad ima vec rezervisan pregled. Dobavlja listu pregleda koje taj doctor ima tog datuma I u to vreme. Ako je lista prazna, registruje se novi pregled.

U Cemu je Problem?

Dva ClinicAdmin-a registruju nove preglede. Ako jedan dobije praznu listu zauzetih pregleda pre nego sto drugi sacuva svoj pregled, moze se desiti da se registruju dva pregleda u isto vreme, istog datuma, sa istim doktorom, ali drugacijeg tipa I u drugacijoj sobi.

Kako je Resen Problem?

Slicno kao I prethodni problem, resen je uz pomoc **@Transactional(isolation=Isolation.SERIALIZABLE)** anotacije. Time

se zakljucava transakcija, tako da druga transakcija nece moci da pocne pre nego sto se prva zavrsi, cime se ovaj problem resava.

ClinicAdmin potvrdjuje/odbija vec potvrdjeni/odbijeni zahtev za odmor/odsustvo

Zadatak

Omoguciti da administrator klinike ne moze da potvrdi ili odbije vec prethodno potvrdjeni ili odbijeni zahtev za odmor ili odsustvo.

Opis Situacije

ClinicAdmin ima pred sobom tabelu svih zahteva za odmor/odsustvo od strane doktora. Klikom na zahtev I na dugme za odobrenje (ili odbijanje) zahteva on salje server zahtev za update-ovanje zahteva.

Server zatim nadje zahtev za odmor, promeni approved bolean vrednost na true ili false (I, ako odbija zahtev, stavi enabled na false) I cuva u bazu.

U Cemu je Problem?

Dva ClinicAdmin-a, **ca1** I **ca2** su kliknula na isti zahtev I u skoro isto vreme su kliknuli dugme. Medjutim, jedan je odobrio zahtev za odmor, dok ga je drugi odbio. Ako je **ca2** pokrenuo transakciju posle **ca1**, registrovace se njegov zahtev, postoji sansa da, nakon sto **ca1** odobri zahtev, **ca2** ga odbije.

Kako je Resen Problem?

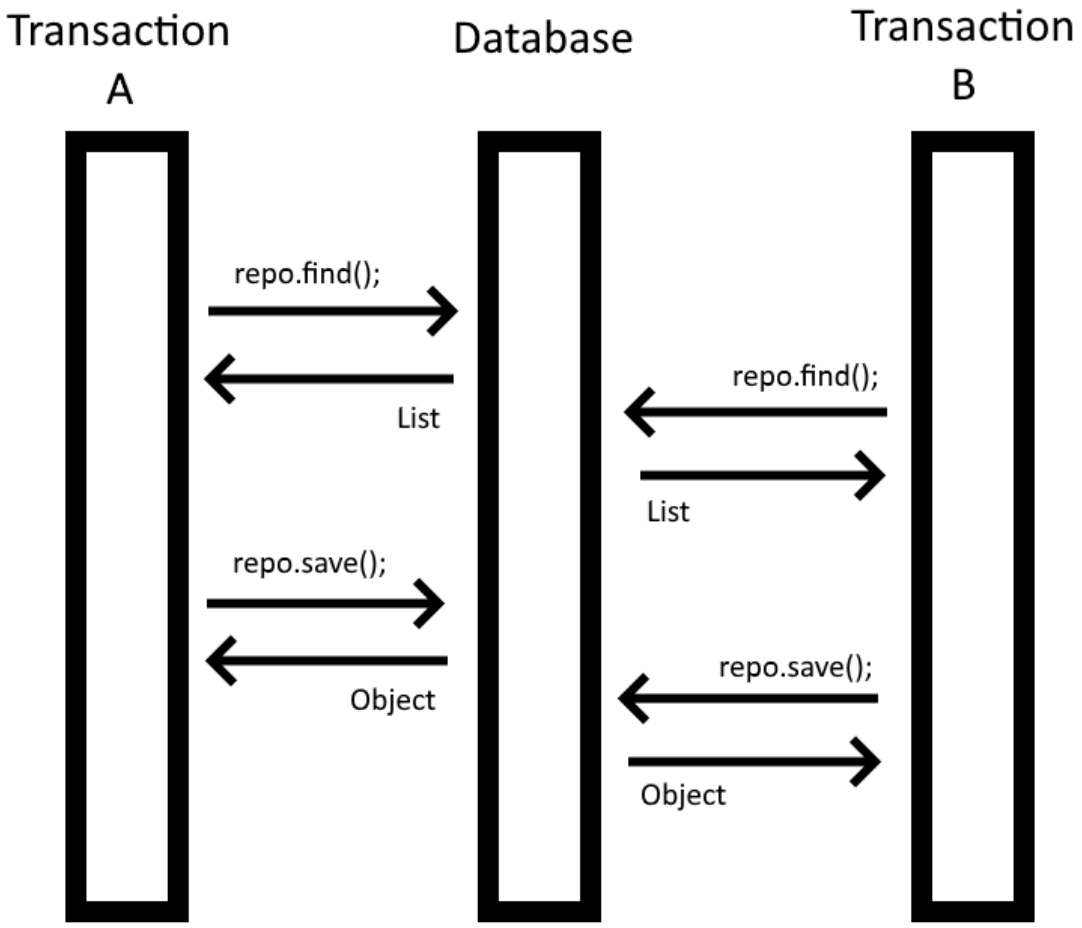
Problem je resen uz pomoc Optimistic Locking-a.

U VacationDbModel dodat je novi field, **private int version**, i dodeljena mu je **@Version** anotacija.

Pomocu **@Version** anotacije postize se to da se, pre nego sto se sacuvaju izmene za neki podatak, **proveri verzija podatka koji se cuva**. Ako su verzije iste, onda se cuva podatak, a **verzija se inkrementira**. Ukoliko nisu iste, zabranjuje se cuvanje I throw-uje se **ObjectOptimisticLockingFailureException**.

Sada, kada **ca1** sacuva izmene, on ce inkrementirati verziju podatka. Posto je **ca2** dobavio iz baze podatak pre nego sto ga je **ca1** sacuvao, **ca2** ima staru verziju podatka. Kad **ca2** bude pokrenuo save() opciju, throw-ovace se **ObjectOptimisticLockingFailureException** I bice mu zabranjeno da edituje izabrani zahtev.

Slika Problema



Sva tri problema se svode na ovaj slucaj. **A** dobavlja listu iz baze I, pre nego sto transakcija **A** sacuva svoje izmene, **B** dobavlja istu listu iz baze. Sada I **A** I **B** imaju dozvolu da menjaju podatke bez ikakve provere prilikom cuvanja.