Primer rešavanja konfliktnih situacija

Internet softverske arhitekture, Milan Đurić, SW 12-2017

1. Obrada zahteva za odsustvo

• Opis konfliktne situacije

Jedna od traženih funkcionalnosti aplikacije jeste da se omogući članu medicinskog osoblja (lekaru ili medicinskoj sestri) da podnese zahtev za odsustvo. Nakon podnošenja zahteva, administrator klinike može da odobri ili odbije zahtev. Ograničenja koja su nametnuta na nivou aplikacije su sledeća:

1. Član medicinskog osoblja ne može da ima dva odobrena odsustva za vremenske intervale koji se preklapaju.
2. Lekar ne može da ima termin pregleda (bilo da je pregled rezervisan ili slobodan) u vremenskom intervalu koji se preklapa sa intervalom odobrenog zahteva za odsustvo.
3. Lekar ne može da bude angažovan kao „dodatni lekar" u sklopu operacije čiji vremenski interval se preklapa sa intervalom odobrenog zahteva za odsustvo.

• Pristup problemu

Možemo primetiti da se ograničenje 1 tiče lekara i medicinske sestre, dok se ograničenja 2 i 3 tiču samo lekara. Sva tri ograničenja potrebno je proveriti isključivo pri **odobravanja zahteva** za odsustvo. To znači da se postupak kreiranja zahteva za odsustvo od strane člana medicinskog osoblja, kao i postupak odbijanja istog, odvija bez provera koje se tiču ova tri ograničenja.

• Konkurentni pristup resursima

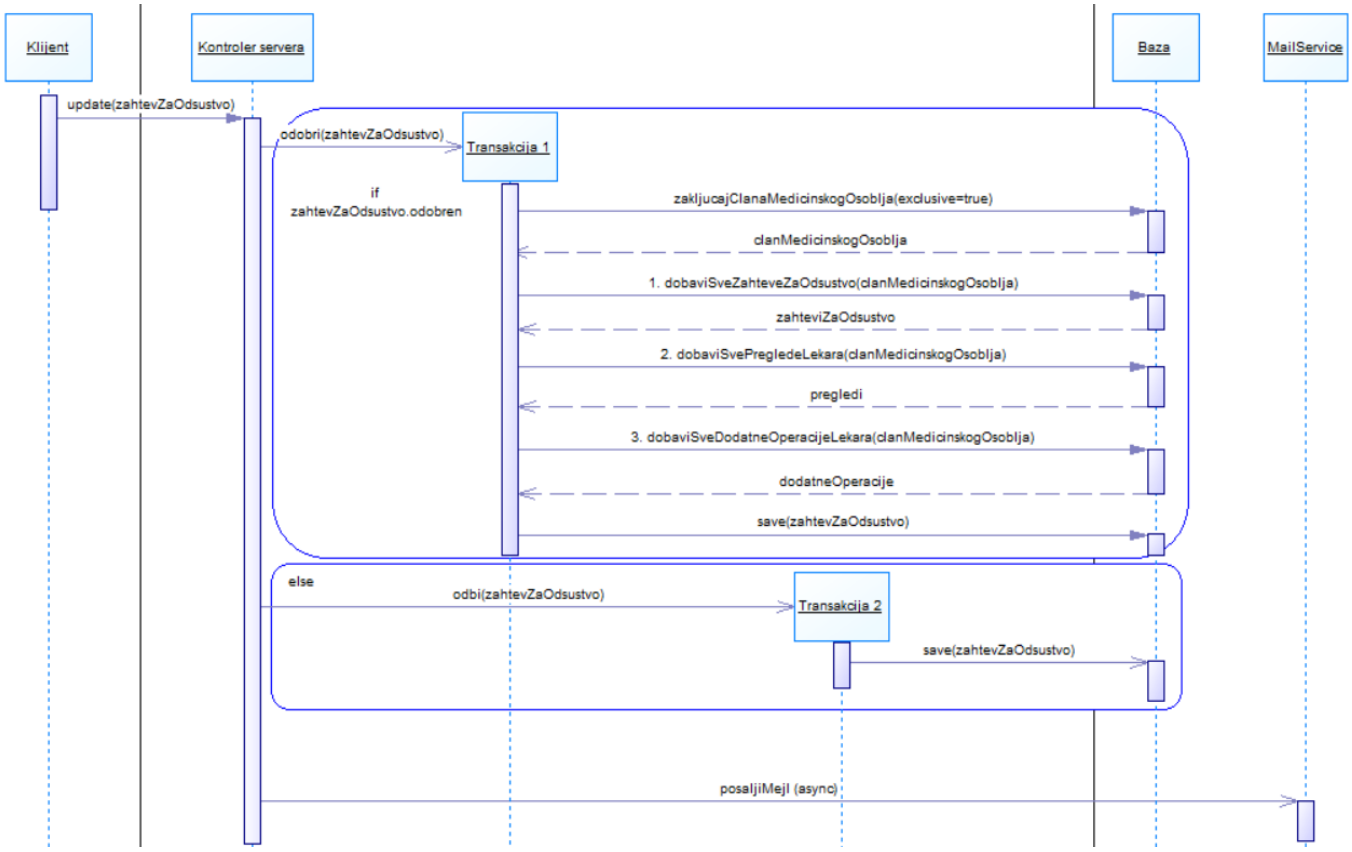
Usled prirode same aplikacije, potpuno je moguće da se za vreme odobravanja zahteva za odsustvo odvija još jedan proces u sistemu koji može da naruši njegov integritet. Na primer, prilikom provere ograničenja 2, moguće je da neki drugi administrator klinike kreira slobodan termin pregleda kod lekara čiji zahtev za odsustvo je upravo u procesu odobravanja, i to baš u vremenskom intervalu koji se preklapa sa terminom odsustva. Da bih sprečio nekonzistencije i narušavanje integriteta aplikacije, koristio sam transakcije sa striktnijim nivoima izolacije i tehnikama zaključavanja. Sam postupak odobravanja ili odbijanja zahteva za odsustvo vizualizovan je na slici (Slika 1), a u narednom paragrafu ću detaljnije objasniti logiku kojom sam se služio.

Klijent (administrator klinike) koristi isti REST endpoint i za odobravanje i za odbijanje zahteva za odsustvo. Ukoliko zahtev treba da se odobri, iz kontrolera se poziva metoda servisa za odobravanje zahteva za odsustvo koja se izvršava u okviru transakcije (transakcija 1). U okviru ove transakcije treba da se izvrše provere ograničenja 1, 2 i 3. Ako bismo posmatrali samo ograničenja 2 i 3, dovoljno bi bio samo da se na početku ove servisne metode dobavi lekar iz baze podataka u PESSIMISTIC_READ režimu, odnosno sa deljenim lock-om. Kako se u okviru transakcije za dodavanje pregleda lekar dobavlja u PESSIMISTIC_WRITE režimu, odnosno sa ekskluzivnim lock-om, zaključujemo da na ovaj način ne bi bilo moguće da se transakcija za odobravanje zahteva za odsustvo (transakcija 1) i transakcija za dodavanje pregleda (nije prikazana na slici) izvršavaju konkurentno nad istom instancom lekara. Iz tog razloga ne bi moglo ni da dođe do narušavanja ograničenja 2 i 3. Međutim, kako u sistemu može postojati više neobrađenih zahteva za godišnji odmor istog lekara (i to u vremenskim intervalima koji se preklapaju), za ograničenje 1 nije dovoljno samo dobaviti lekara u PESSIMISTIC_READ režimu. Iz tog razloga se pri odobravanju zahteva za godišnji odmor, lekar dobavlja na početku transakcije iz baze podataka u PESSIMISTIC_WRITE režimu, kako ne bi mogli simultano da odobravamo više različitih zahteva za odsustvo istog lekara, jer nismo

sigurni da li se njihovi vremenski intervali preklapaju. Napominjem da mislim da je ovo rešenje mnogo bolje nego da smo samo stavili nivo izolacije SERIALIZABLE, zato što ovako možemo paralelno da odobravamo zahteve za odsustvo različitih lekara. Strategija optimističkog zaključavanja ovde ne bi pomogla pri pitanju ograničenja 1, jer se ovo ograničenje odnosi na više različitih zahteva za odsustvo.

Što se tiče postupka odbijanja zahteva za odsustvo, on se izvršava u okviru transakcije 2. U ovom slučaju ne moramo da proveravamo nikakva ograničenja. Ono što međutim treba da sprečimo, jeste da se isti zahtev za odsustvo simultano obrađuje (na primer da jedan administrator pokrene odbijanje, a drugi odobravanje istog zahteva za odsustvo). Razlog zašto nam je za ovo potrebna dodatna konfiguracija je taj što hibernate neće izvršiti kaskdadno zaključavanje referenciranih entiteta (između ostalog zahteva za odsustvo) prilikom zaključavanja torke lekara u transakciji za odobravanje zahteva za odsustvo (transakcija 1). Prvi pristup bi bio da zahtev za odsustvo koji treba da se odbije dobavimo iz baze u PESSIMISTIC_WRITE režimu i samim tim sprečimo njegovo konkurentno obrađivanje. Drugi pristup je da koristimo optimističko zaključavanje. Kako smatram da odbijanje zahteva za odsustvo neće biti česta operacija u sistemu, odlučio sam se za optimističko zaključavanje.

Napominjem da se obe transakcije (1 i 2) vrše kao readOnly=false sa nivoom izolacije READ_COMMITTED (ovaj nivo izolacije je podrazumevani za većinu baza podataka, ali ne i za MySQL koji je korišćen u ovom projektu, tako da sam morao eksplicitno da ga postavim).



Slika 1: Postupak ažuriranja(odobravanja ili odbijanja) zahteva za odsustvo. Upiti 2 i 3 se izvršavaju samo ako je član osoblja koji je podneo zahtev zapravo lekar.

2. Obrada upita za pregled

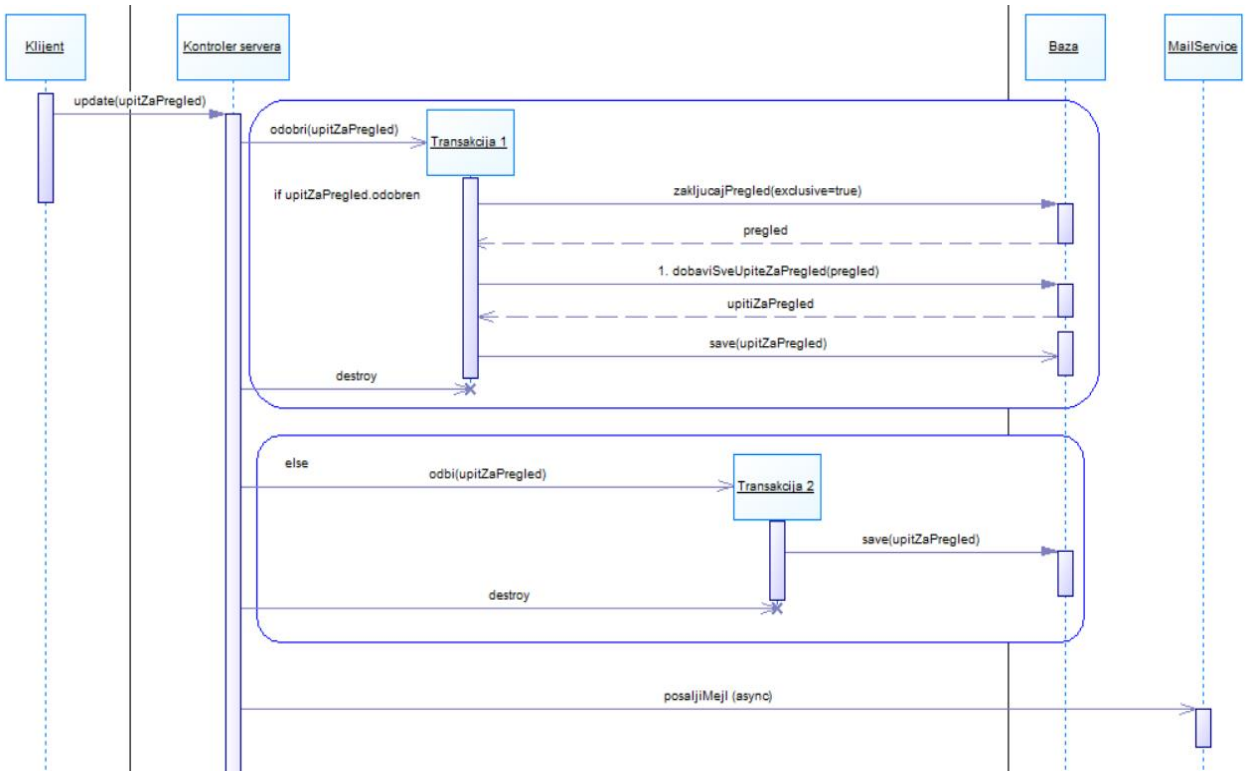
Napomena: Iako piše u specifikaciji da ne može više pacijenata da zatraži upit za pregled kod istog lekara, ja sam se ponašao kao da može.

- ### Opis konfliktne situacije

U sistemu postoje dve vrste upita za pregled. Prva vrsta je upit za unapred definisani pregled. Ova vrsta upita se odnosi na pregled koji već ima dodeljene sve neophodne resurse (lekara, salu, tip pregleda). Pri obradi ovog upita, jedino treba voditi računa da se ne odobri više od jednog upita za isti pregled, kao i da se svaki upit obradi samo jednom. Sa druge strane, pacijenti mogu i da pošalju upit za pregled tako što specificiraju samo željeno vreme početka, tip pregleda i lekara. U ovom slučaju administrator, ukoliko odobri upit, treba da kreira konkretan objekat pregleda u sistemu, tako što će mu dodeliti neku od slobodnih sala i promeniti lekara ili vreme pregleda ako je to potrebno. Kako je ovde potrebno zapravo kreirati pregled, treba voditi računa o još nekim stvarima o kojima ćemo pričati kasnije.

- ### Obrada upita za unapred definisani pregled

Kao što je već napomenuto u prethodnom pasusu, za ovaj upit za pregled je karakteristično to što je objekat pregleda već kreiran. To znači da su sala, lekar kao i stanica sigurno validni i ne narušavaju konzistentnost podataka (inače pregled uopšte ne bi ni bio kreiran). Pri ***odobravanju*** upita za pregled, jedino ograničenje koje treba proveriti jeste da jedna instanca pregleda nema više od jednog odobrenog upita za pregled. Usled prirode same aplikacije, potpuno je moguća situacija u kojoj dva različita administratora pokreću istovremeno odobravanje dva različita upita (od dva različita pacijenta) za isti pregled. U takvom scenariju moguće je da dođe do štetnog preplitanja i da se prethodno ograničenje naruši. Iz tog razloga, na početku transakcije za odobravanje upita za unapred definisani pregled, instanca ***pregled*** se dobavlja u PESSIMISTIC_WRITE režimu, što praktično ima za posledicu sprečavanje konkurentnog odobravanja više od jednog ***upita*** za istu instancu pregleda. Sa druge strane, postupak ***odbijanja*** upita za pregled treba samo da spreči simultanu obradu(odbijanje-odbijanje ili odbijanje-odobravanje) istog upita za pregled, a ta situacija se prepušta hibernate-u da je detektuje upotrebom tehnike optimističkog zaključavanja upita za pregled. Uvid u postupak obrade upita za unapred definisani pregled nalazi se na Slika 2. Kao i u slučaju obrade zahteva za odsustvo, obe transakcije sa slike se izvršavaju sa READ_COMMITTED nivoom izolacije.

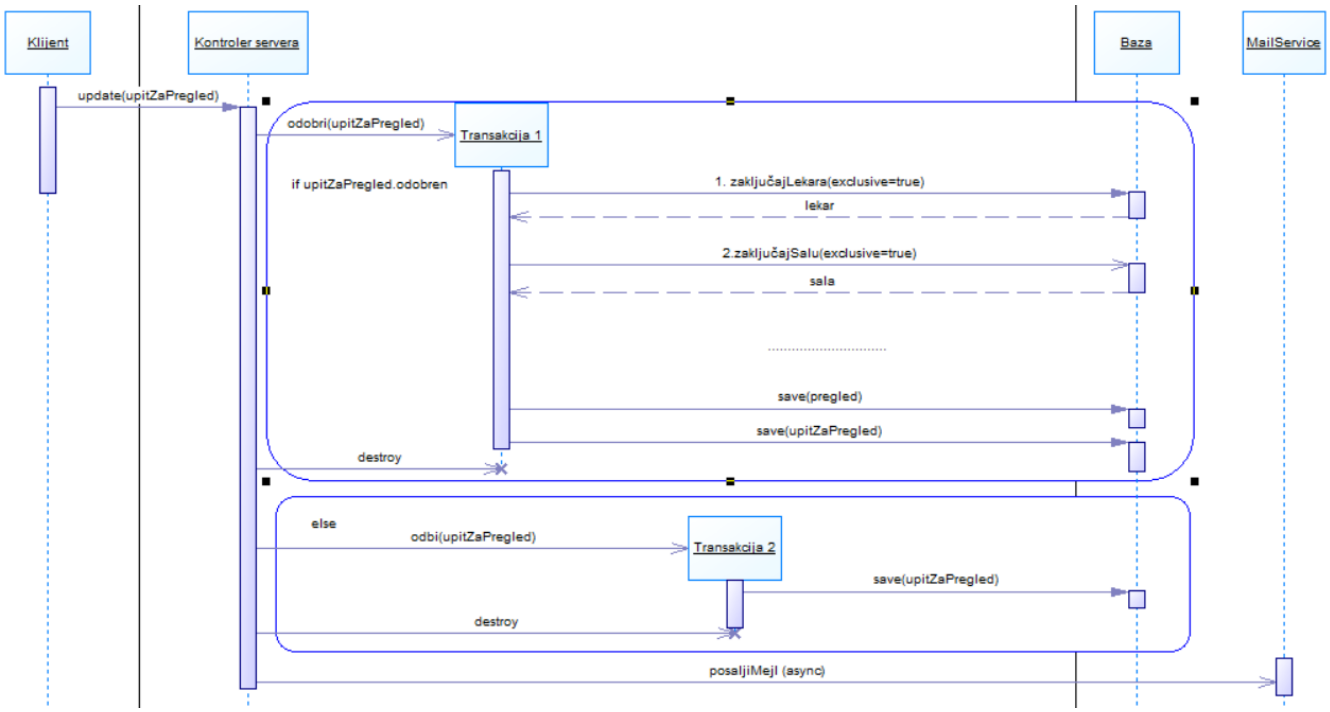


Slika 2: Obrada upita za unapred definisani pregled

- Obrada upita za nedefinisani pregled

U slučaju da administrator želi da ***odobri*** upit za nedefinisani pregled, prilikom odobravanja tog upita potrebno je kreirati instancu pregleda. Pri kreiranju pregleda, potrebno je proveriti da li je lekar već angažovan na nekom drugom pregledu ili operaciji za datu satnicu, kao i da li je sala slobodna za datu satnicu. U ovoj situaciji treba voditi računa da je moguće da neki drugi administrator konkurentno pokuša da kreira pregled kod istog lekara ili u istoj sali, i to baš za vremenski interval koji se preklapa sa našim intervalom. Iz tog razloga se na početku transakcije za dodavanje pregleda lekar i sala dobavljaju u PESSIMISTIC_WRITE režimu. Na ovaj način smo sprečili konkurentno dodavanje više različitih pregleda kod istog lekara ili u istoj sali.

Posmatrajmo sada situaciju u kojoj dva različita administratora istovremeno pokušavaju da odobre isti upit za nedefinisani pregled, pri čemu su oba administratora promenili i salu i lekara iz originalnog upita (upita koji je pacijent poslao) na taj način da su sale i lekari međusobno različiti. U ovoj situaciji, prilikom dodavanja pregleda, ove dve transakcije se neće međusobno blokirati, jer obe rade nad različitim torkama lekara i sala. To znači da će se dodavanje oba pregleda uspešno izvršiti, što svakako nije željeno ponašanje. Na svu sreću, možemo da se oslonimo na hibernate i optimističko zaključavanje nad upitima za pregled da prepoznaju konkurentu obradu istog upita za pregled i da bace *ObjectOptimisticLockingFailureException*. Na taj način bi se jedna od transakcija iz prethodno opisane situacije sigurno poništila, a samim tim bi se i poništilo jedno dodavanje pregleda. Kao i u prethodna dva slučaja, obe transakcije sa slike se izvršavaju sa READ_COMMITTED nivoom izolacije.



Slika 3 Obrada upita za nedefinisani pregled