Rešavanje konfliktnih situacija prilikom konkurentnog pristupa – student 2

Do konfliktnih situacija dolazi kada više korisnika konkurentno može pristupati istom resursu i menjati ga. Ovakvih situacija u projektu ima mnogo (svako editovanje ili brisanje može dovesti do konflikta), ali odabrane su i odrađene sledeće tri situacije za koje mislim da su dovoljno reprezentativne.

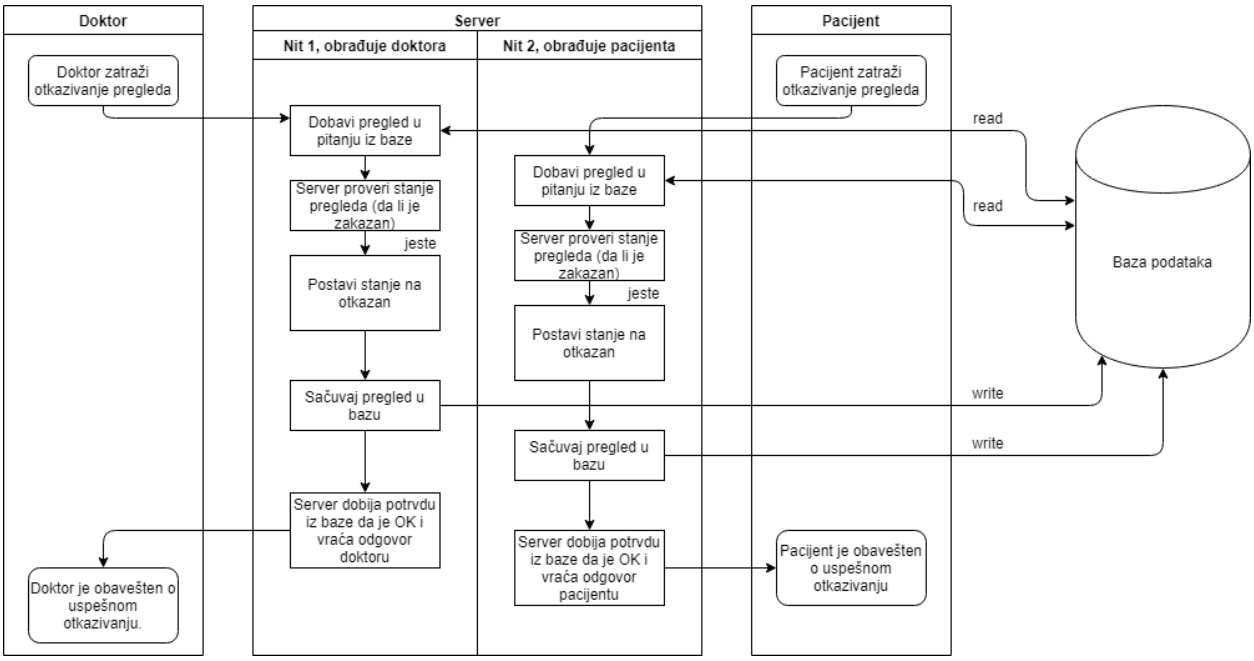
Situacija 1: Doktor i pacijent u isto vreme pokušavaju da otkažu isti pregled

Do konfliktne situacije dolazi u sledećem slučaju – pacijent i doktor u približno isto vreme pokušavaju da otkaži isti pregled (na primer) sa ID-om 1. Oba zahteva su poslata na server u približno isto vreme i njihovo izvršavanje je paralelno.

Algoritam na serveru za obradu je sledeći: Dobavi odgovarajući pregled iz baze, promeni mu status sa *Scheduled* na *Canceled*, snimi ga u bazu i obavesti korisnika da li je otkazivanje uspešno izvršeno. Do problema dolazi što u kratkom vremenskom periodu i lekar i pacijent dobave isti pregled iz baze i njegova izmena se vrši u dve različite niti – posledica toga je konflikt. Pretpostavimo da je lekar uspeo prvi da sprovede otkazivanje. Pacijent i dalje „obrađuje“ pregled koji je sada već otkazan, ali u njegovoj verziji dobavljenog objekta i dalje važi status *Scheduled*. Tako dolazimo do situacije da pacijent dobija poruku da je uspešno otkazao pregled, iako je doktor to uradio pre njega.

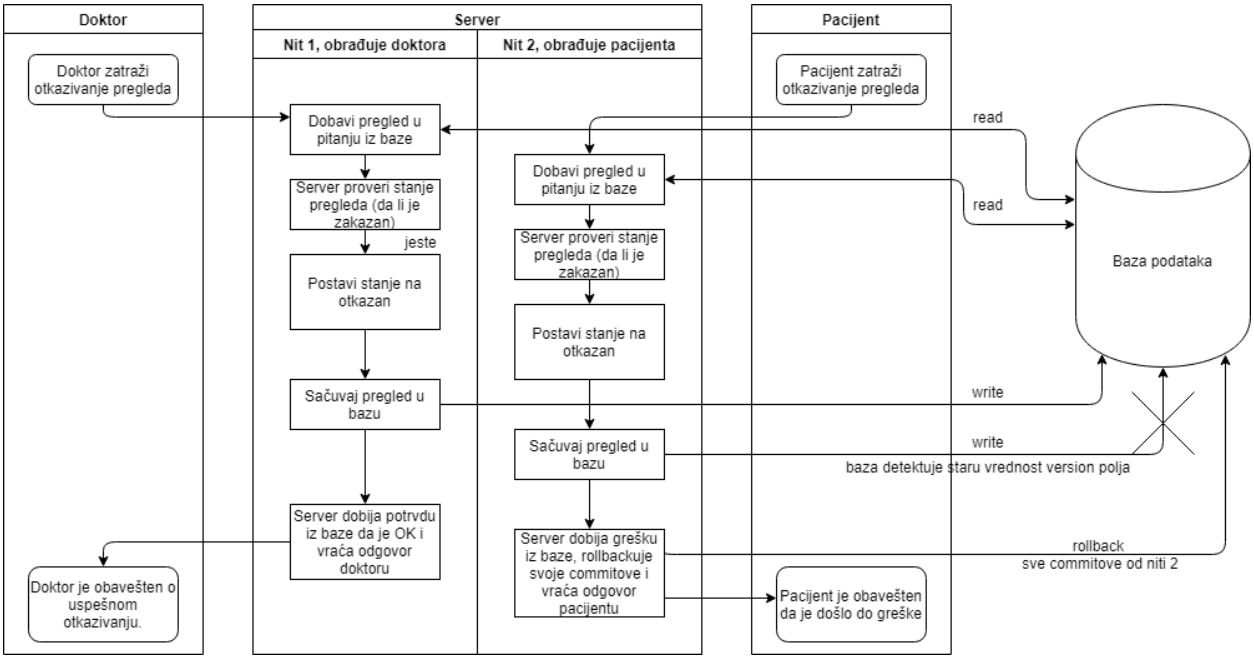
Koraci su objašnjeni dijagramom ispod:

(dijagram)



Potrebno je uvesti mehanizme koji bi sprečili konfliktne situacije – u ovom slučaju uvode se transakcije i optimističko zaključavanje. Transakcije odgovaraju ovakvim tipovima problema jer, ili će se izvršiti komplento, ili se neće uopšte izvršiti. Razlog za odabir optimističkog zaključavanja (a ne pesimističkog) je taj što nema potrebe za sprečavanjem pristupa bazi tokom izvršavanja otkazivanja i jer se optimističkim zaključavanjem postižu bolje performanse. Dodato je polje version – celobrojna vrednost koja služi za proveravanje da li je objekat bio izmenjen u međuvremenu.

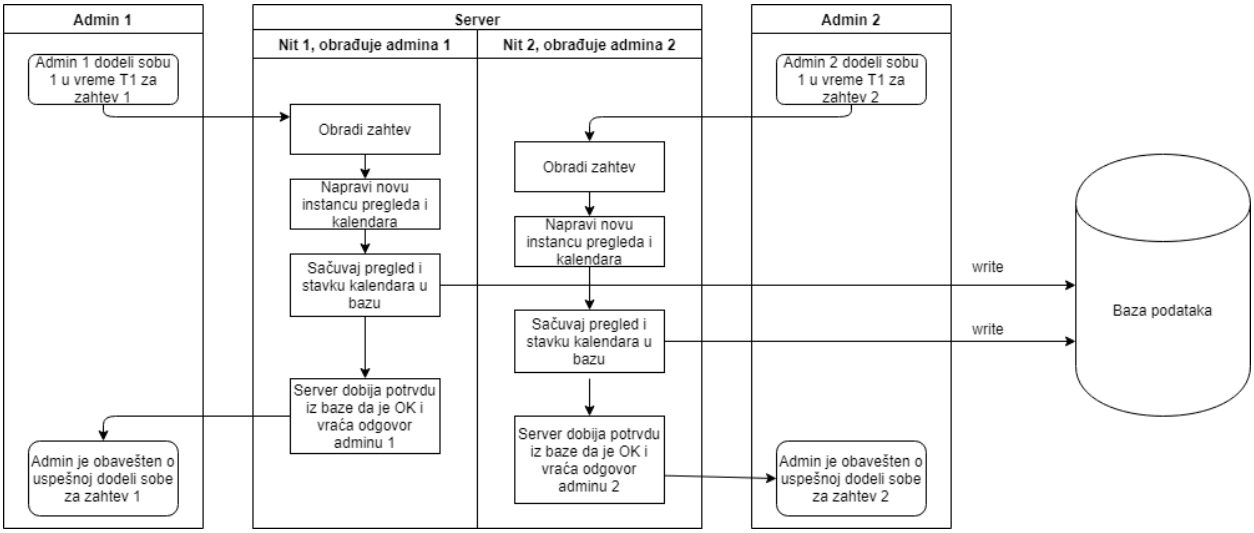
Konkretno u ovoj situaciji doktor koji je prvi uspeo da otkaže pregled bi uspešno sačuvao taj objekat u bazu i tada bi se inkrementovala vrednost verzije. Nakon toga, pacijent bi pokušao da sačuva taj isti pregled u bazu, ali sada bi verzija tog objekta bila drugačija od onog u bazi i ne bi došlo do čuvanja. Transakcija koju je započeo pacijent bi bila rollbackovana i ponovno otkazivanje bi bilo sprečeno uz adekvatnu povratnu poruku pacijentu.



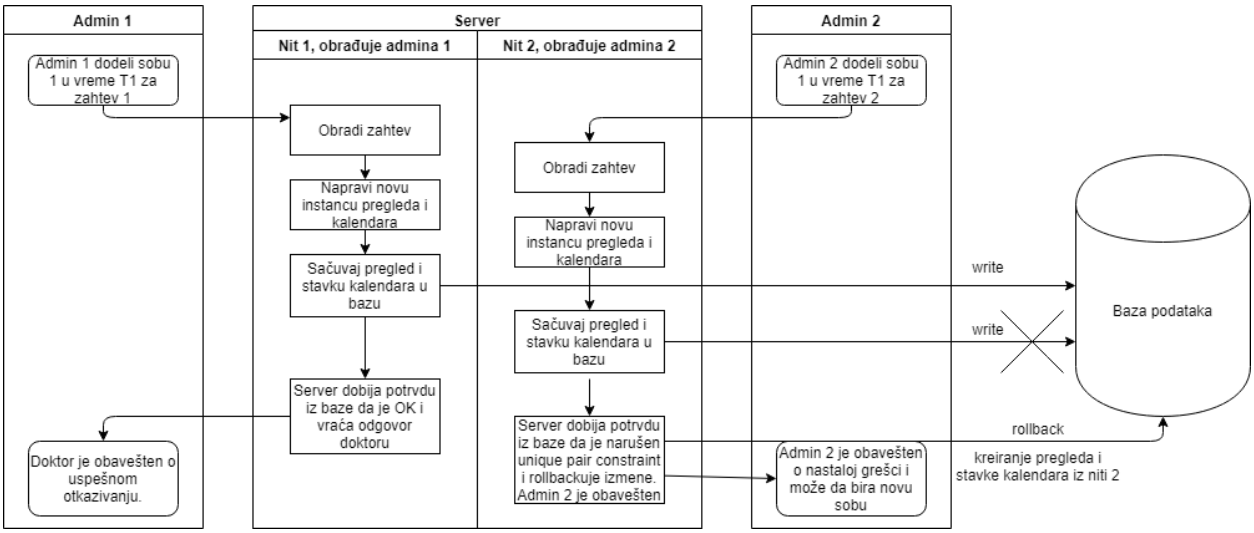
Situacija 2: Admini žele da dodele istu sobu za preglede koji se održavaju istog dana u isto vreme

Admini dobijaju zahteve za preglede koje treba da odobre i da dodele sobu koja je slobodna u zatraženom terminu (ili prvom slobodnom sledećem, ukoliko za zahtevani termin ne postoji ni jedna slobodna soba).

Algoritam na serveru za obradu je sledeći: Dobavi odgovarajući request iz baze, napravi instancu Appointmenta (pregled) i instancu stavke kalendara (CalendarEntry) za doktora u specifično vreme i specifičnu sobu. Zatim sačuvaj sve u bazu.



Moguća situacija je da će dva admina u relativno slično vreme, obrađujući zahteve, pokušati da dodele istu sobu za npr. 2 različita pregleda koja se održavaju u istom terminu. Svaki odobreni zahtev za pregled menja svoj status is *Requested* u *Scheduled* i postaje stavka kalendara kod lekara – odn. dodaje se u tabelu *calendar_entry*. Pošto se ovo dešava u gotovo isto vreme oba admina će imati istu listu slobodnih termina za sobe iz klinike – metoda koja dobavlja prve slobodne termine za sobe će se izvršiti u gotovo isto vreme. Ono što je ključno u ovom problemu jeste da se ne sme desiti da u tabeli *calendar_entry* postoje „dva reda“ koja će sadržati iste vrednosti za početno vreme održavanja pregleda i id sobe.

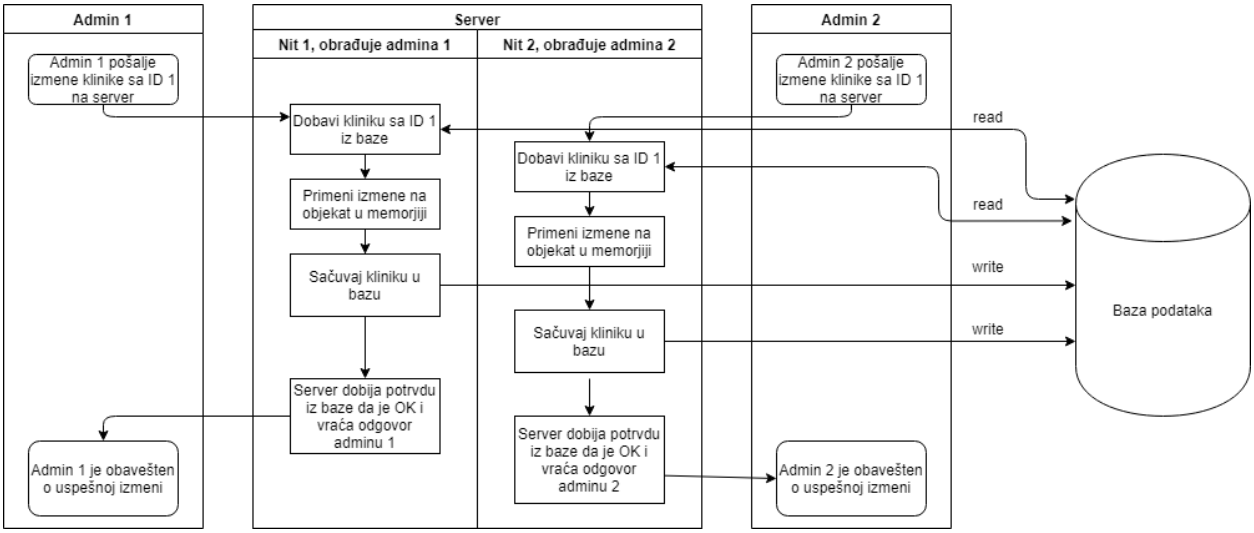


Uvođenjem *UniqueConstraint* anotacije nad kolonama *start_date* i *room* sprečeno je da uopšte dođe do insertovanja ovakvih objekata, a ukoliko dođe do pokušaja čuvanja ovakvog (nekonzistentnog) objekta, transakcija se neće izvršiti do kraja i rollback-ovaće se sve izmene nad bazom (insertovanje novog Appointment objekta). Admin će biti obavešten odgovarajućom porukom.

Situacija 3: Admini pokušavaju da izmene informacije o klinici u isto vreme

Svaki admin ima pristup informacijama o klinici za koju je zadužen i može da menja njene podatke. Na primer, neka su admin1 i admin2 u približno isto vreme pokušali da izmene podatke o klinici čiji ID je 1. Oba zahteva su poslata na server u približno isto vreme i njihovo izvršavanje je paralelno.

Algoritam na serveru je sledeći: Dobavi odgovarajuću kliniku iz baze, izmeni njene podatke, sačuvaj izmene u bazu, obavesti korisnika da li je izmena uspešno izvršena. Do problema dolazi što u kratkom vremenskom periodu i admin1 i admin2 dobave istu kliniku iz baze i njena izmena se vrši u dve različite niti, što dovodi do konflikta.



Problem se rešava uvođenjem verzije za kliniku i optimističnog zaključavanja. Pretpostavimo da je admin1 prvi uspešno izmenio kliniku i sačuvao u bazu. Verzija je inkrementovana i ne poklapa se sa onom koja kod objekta (klinike) koju obrađuje admin2. Kako sada admin 2 ima zastarelu verziju klinike, neće uspeti da sačuva svoje izmene i dobiće odgovarajuću poruku.