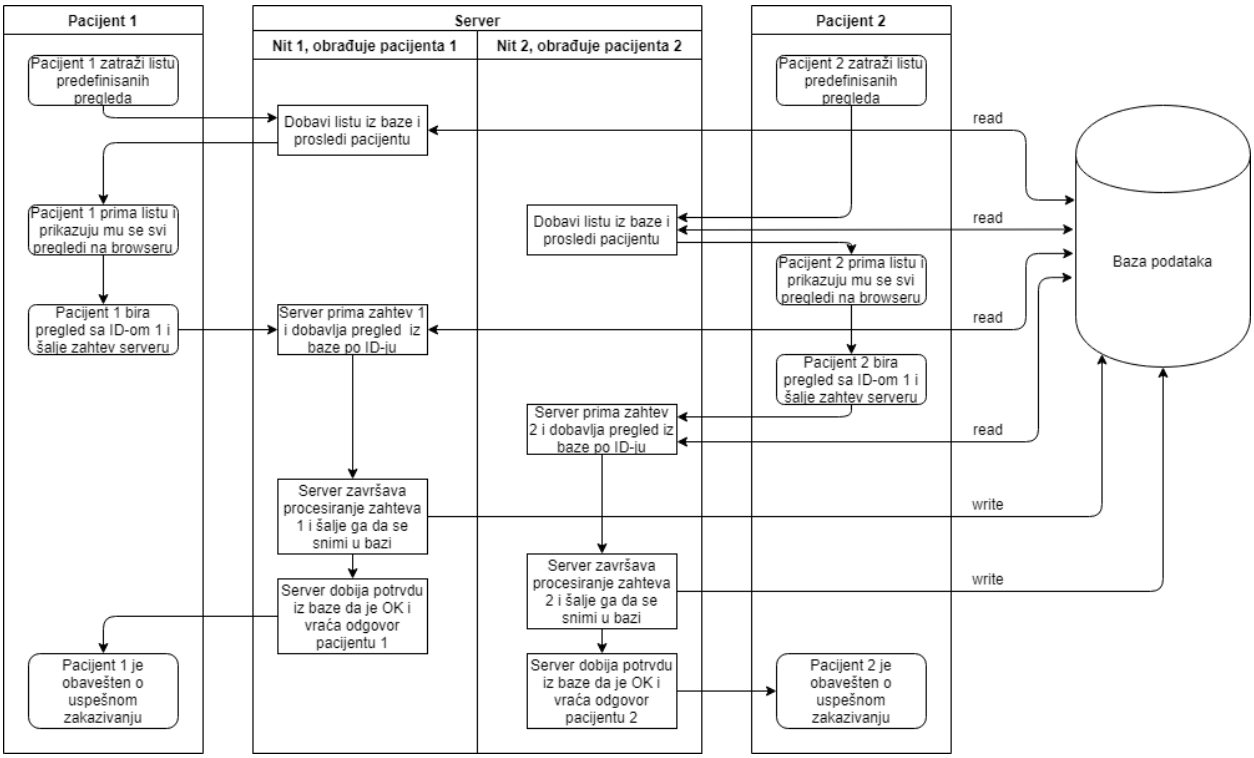
Rešavanje konfliktnih situacija prilikom konkurentnog pristupa – student 1

Situacija 1: više pacijenata u isto vreme pokušavaju da zakažu isti predefinisani pregled

Do konfliktne situacije dolazi u sledećem slučaju – dva pacijenta u okvirno istom vremenu otvore stranicu za prikazivanje predefinisanih pregleda, i oba pacijenta odluče da zakažu pregled (na primer) sa ID-om 1. Oba zahteva su poslata na server u približno istom vremenu, i njihovo izvršavanje je paralelno. Algoritam na serveru za obradu je sledeći – dobavi pregled iz baze, promeni mu status sa Predefinisan na Zakazan, stavi referencu na pacijenta za koji je vezan u svoje polje, snimi ga nazad u bazu i obavesti korisnika da li je čuvanje uspešno. Konflikt se dešava jer, kada se pacijent 1 stavi kao pacijent za navedeni pregled i sačuva u bazu, druga nit, koja i dalje ima objekat u stanju Predefinisano, obavi isti posao, ali stavi za pacijenta pacijenta 2, i čuva u bazu. Ovim dolazi do situacije da pacijent 1 dobija poruku kako je zakazao pregled, a zapravo je pacijent 2 uspešno zakao pregled.

Koraci su objašnjeni dijagramom ispod:

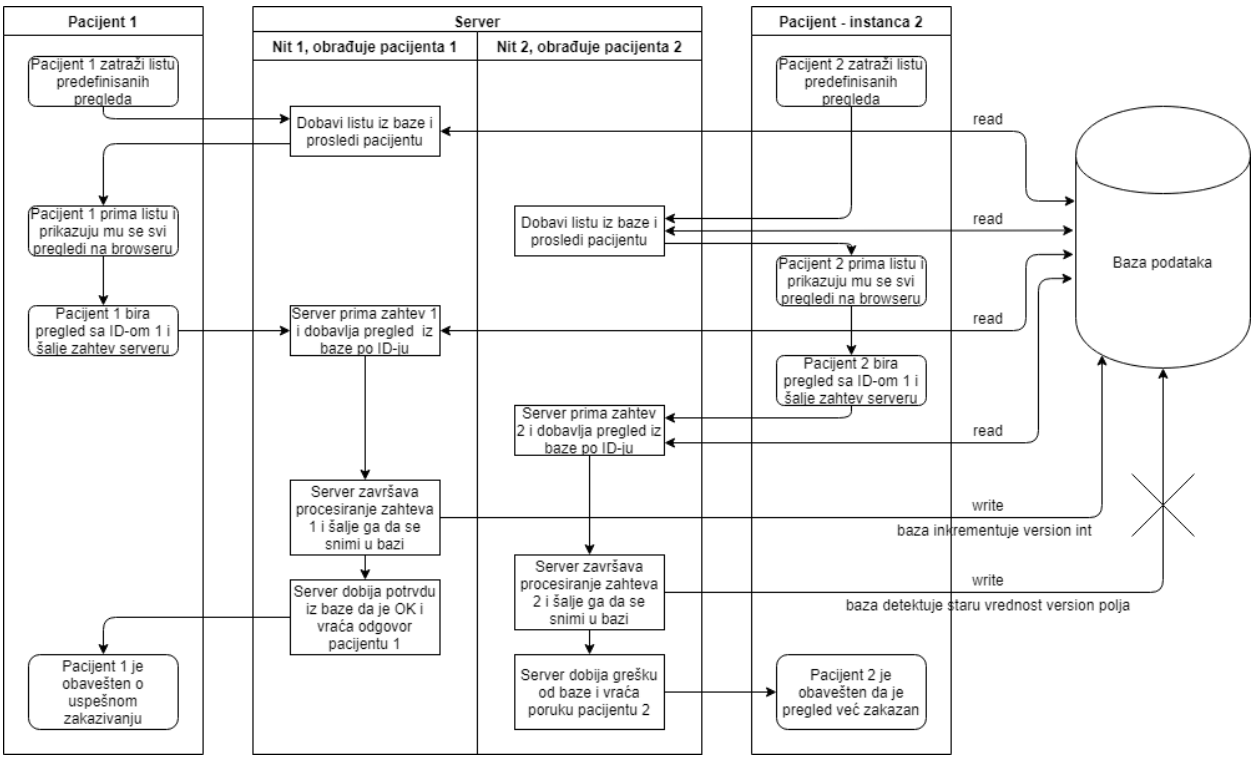


Naravno, ovakvo ponašanje je nedopustljivo – pacijent ne može biti obavešten da je zakazao pregled, a u realnosti nije uspeo da ga zakaže čak i ako je prvi zatražio taj pregled.

Ovaj problem se rešava uvođenjem transakcija i optimističnog zaključavanja. Izmene u modelu koje zahteva optimistično zaključavanje jesu dodavanje polja version, koje predstavlja celobrojni broj. Ono služi da se, kada server pokuša da pošalje izmenjen objekat u bazu na čuvanje, proveri da li je neko izmenio taj objekat u međuvremenu, i ako nije, onda se uvećava to polje, da bi se svaka sledeća izmena koja je sa manjim brojem onemogućila. Ovim se postižu dobre performanse, jer ne moramo da zaključavamo bazu od pristupa dok se ne izvrši transakcija, nego možemo da pokušamo da izvršimo izmenu, i ako ne uspe, promene se neće sačuvati i korisnik će biti obavešten o nastaloj situaciji.

Koraci koji se sada dešavaju su slični kao u prethodnom slučaju, sa izmenom da, kada prva nit sačuva pregled u bazi, vrednost polja version u bazi se automatski uveća za jedan. Sada, kada druga nit pokuša da sačuva svoje izmene, njena vrednost version polja u memoriji je manja od one u bazi, i time se otkriva konflikt i obustavlja čuvanje. Pacijentu 2 se vraća poruka kako je neko već zakazao taj pregled.

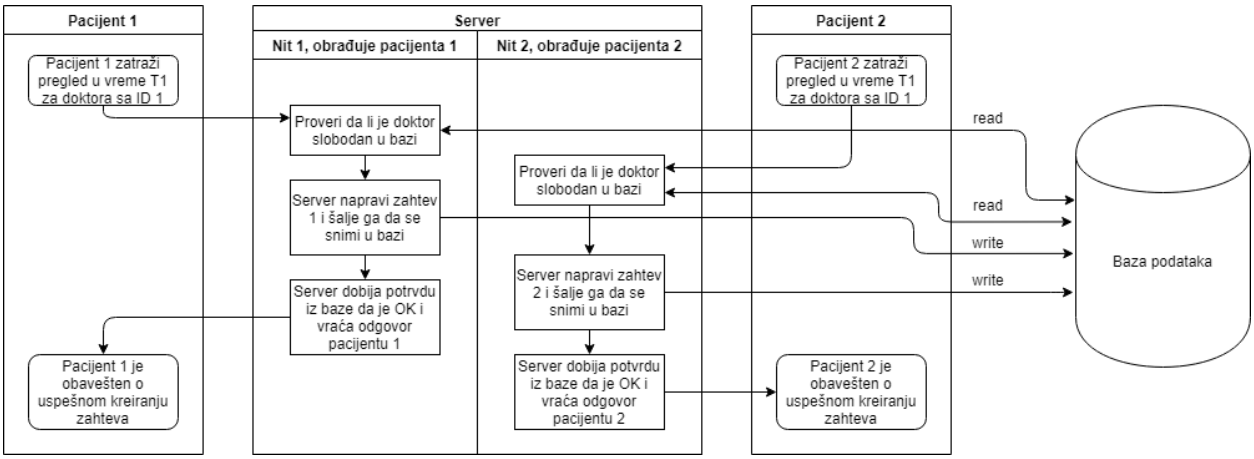
Koraci su opisani dijagramom ispod:



Situacija 2: više pacijenata u isto vreme traže pregled kod istog lekara

Do konfliktne situacije dolazi u sledećem slučaju – dva pacijenta odaberu isti termin za pregled u isto vreme i pošalju zahteve na server. Algoritam na serveru je sledeći – proveri da li je doktor slobodan za to vreme dobavljanjem objekta zahteva iz baze, ako jeste (taj objekat ne postoji), napravi zahtev za to vreme i sačuvaj ga u bazu. Do konflikta dolazi ako se, prilikom paralelnog izvršavanja, prvi zahtev sačuva u bazu, a nit koja obrađuje drugi zahtev je već prošla proveru, i ona sačuva svoj zahtev u bazu. Sada je doktor zahtevan na dva pregleda odjednom u isto vreme, što ne smemo da dopustimo.

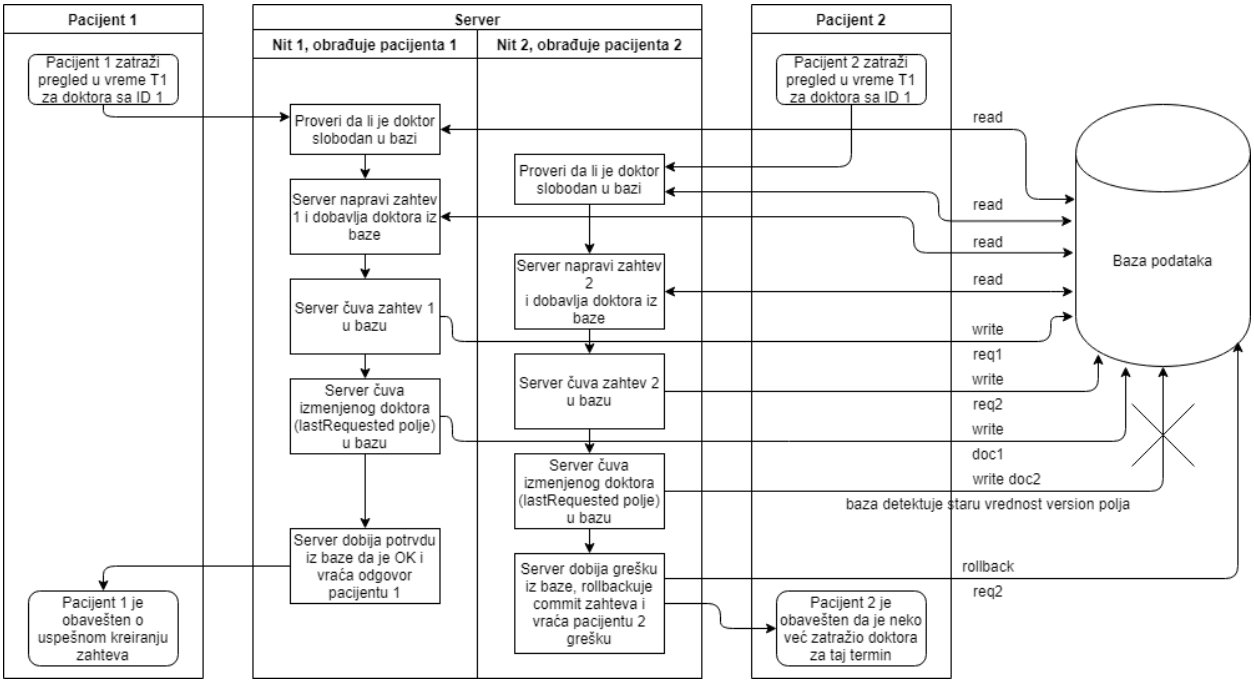
Koraci su objašnjeni dijagramom:



Rešavanje ovog konflikta je slično kao i za prethodni konflikt. Koristi se optimistčno zaključavanje, sa time da se dodaje version polje za User-a (jer će biti potrebno i za druge konflikte). Pošto mi prilikom zahteva za pregled ne menjamo zapravo doktora, dodato je još jedno polje tipa Date, pod nazivom lastRequested, da bismo imali neku izmenu koja se čuva u bazu i time omogućili inkrementovanje version polja.

Novonastali tok dešavanja je sledeći. Kada prva nit napravi zahtev, ona dobavlja doktora iz baze. Zatim mu postavlja polje lastRequested na novi date objekat (trenutno vreme), šalje zahtev na čuvanje, i konačno šalje doktora na čuvanje (automatski se inkrementuje vrednost polja version). Druga transakcija, koja ima doktora sa starom vrednošću polja version, čuva svoj objekat u bazu, i zatim pokušava da sačuva doktora, ali zbog optimističnog zaključavanja ne uspeva, i time se pokreće rollback transakcije. Ovim postižemo da baza ne može doći u nekonzistentno stanje, jer će se po potrebi rollbackovati izmene transakcije koja ne uspe da se izvrši.

Dijagram rešenja:

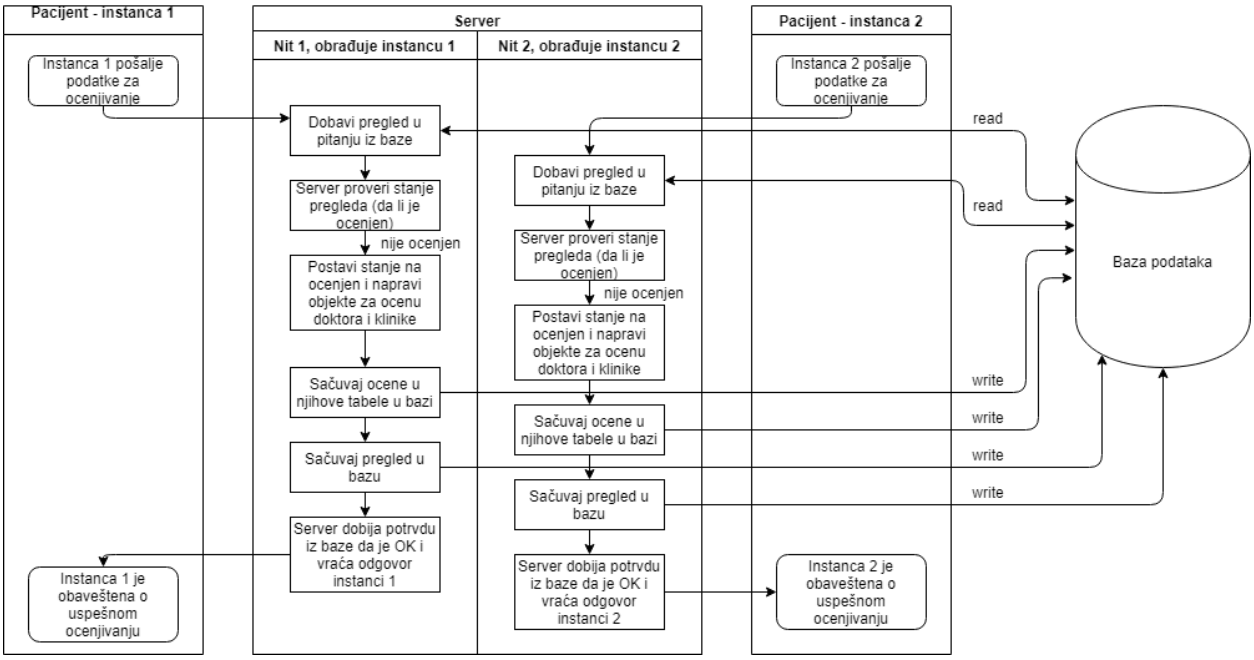


Situacija 3: pacijent sa različitih browser-a pokušava da oceni isti pregled

Ocenjivanje pregleda je napravljeno tako da pacijent može samo jednom da oceni pregled. Pošto ne postoji praćenje toga da li je pacijent već ulogovan na nekom browseru na server strani, moguće je da pacijent odjednom oceni isti pregled sa dva (ili više) browsera i time naruši konzistentnost ocena u bazi. Mogao bi se kreirati napad kojim će se sa mnogo instanci browsera poslati masivne negativne ocene i time narušiti legitimne ocene.

Algoritam za ocenjivanje je sledeći – proslede se podaci na server o oceni klinike i doktora, zatim se dobavi iz baze pregled koji se ocenjuje, i proveri se da nije već ocenjen. Ako nije, kreiraju se objekti ocena (jedan za doktora i jedan za kliniku) i snimaju u bazu, i zatim se snima pregled sa izmenjenim boolean-om za ocenjenost. Konflikt se dešava kada dve ili više niti u isto vreme prođu validaciju da nisu ocenjene, i zatim se kreiraju i snime objekti ocena, i tek onda se postavi flag koji bi sprečio ovu situaciju.

Dijagram konflikta sledi:



Rešavanjem prvog problema uvođenjem optimističnog zaključavanja smo omogućili jednostavno rešavanje ovog problema. Čuvanje objekata ocena za klijenta i doktora u njihove tabele u bazi se odvija pre nego što završimo transakciju commitovanjem izmena za pregled. Zbog ovoga, ako se desi da je nit 1 napravila objekte i snimila ih u bazu, i onda sačuvala izmenjen objekat pregleda, i onda nit 2 pokuša da sačuva svoj izmenjen objekat pregleda (nakon što je već sačuvala ocene u bazu), nit 2 će dobiti exception i rollback-ovati svoje izmene, i time smo osigurali da se pregled samo jednom može oceniti.

Dijagram rešenja: