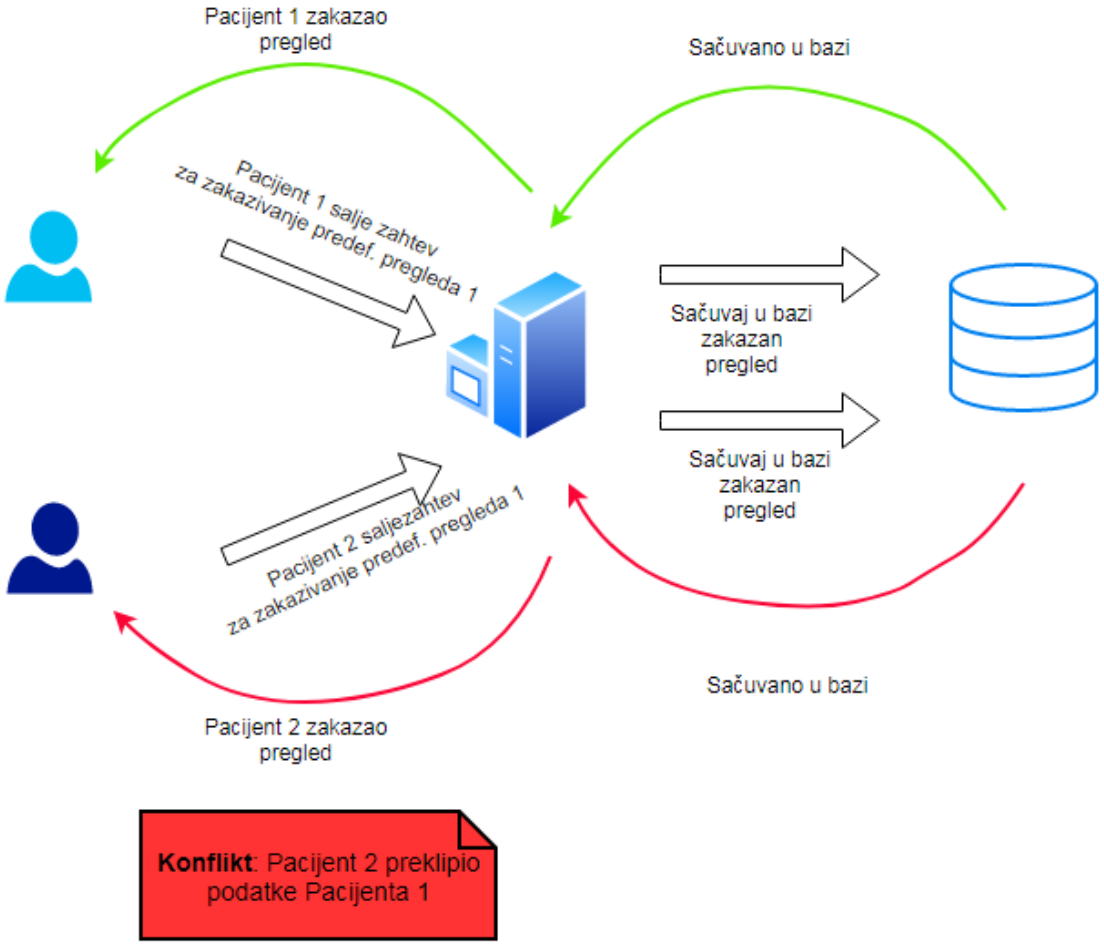
Konfliktne situacije Student 1

Konflikt 1: Dva ulogovana korisnika (pacijenta) žele da rezervišu isti predefinisani termin. Problem se javlja kada jedan od njih rezerviše termin, tada drugom pacijentu nije ažurirana stranica sa predefinisanim terminima i on može da se odluči za isti termin.

Tok zahteva i odgovora:



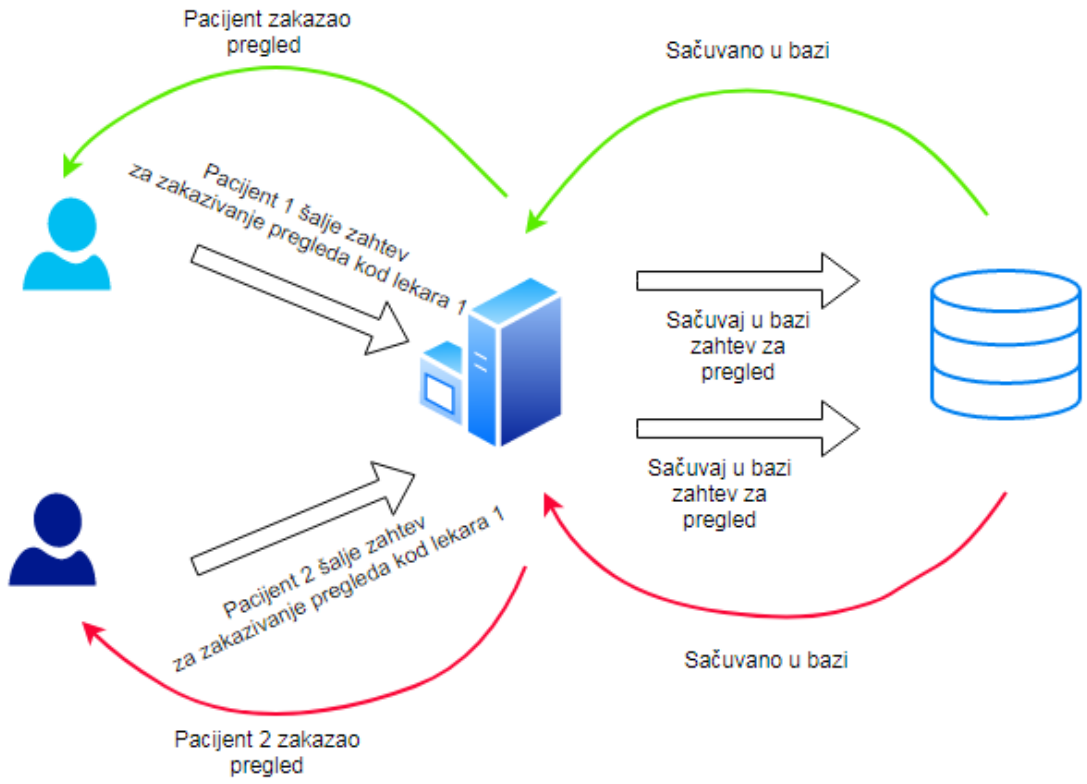
Slika 1 Tok zahteva i odgovora, konflikt 1

Rešenje: Svaki predefinisani termin ne sadrži podatke o pacijentu što znači da kada pacijent zakaže neki od tih termina u bazi se za taj termin vezuju podaci o pacijentu koji ga je zakazao. Kako bi rešila ovaj konflikt, koristim optimističko zaključavanje. U klasi `Pregled` dodajem atribut `version` sa anotacijom `@Version`. Ovaj atribut će biti inkrementiran od strane *Hibernate*-a ukoliko je moguće zakazati pregled tj. ako je verzija u trenutku zakazivanja ista kao i na početku transakcije. U suprotnom, baca se izuzetak i nije moguće zakazati konkretan termin. Da bi se ovaj proces mogao posmatrati kao transakcija neophodno je u klasi `PregledService` dodati anotaciju `@Transactional` iznad klase i iznad metode koja treba da se ponaša kao transakcija. Na ovaj način kontrolišemo pristup i ažuriranje podataka u slučaju kada više korisnika želi da pristupi istim podacima.

Optimističko zaključavanje nije striktno zaključavanje i neće narušiti performanse sistema.

Konflikt 2: Dva ulogovana korisnika (pacijenta) žele da rezervišu istu satnicu i datum za pregled kod istog lekara. Problem se javlja kada jedan od njih pošalje upit za pregled, tada drugom pacijentu nije ažuriran prikaz satnica konkretnog lekara i on može da se odluči za isti termin.

Tok zahteva i odgovora:

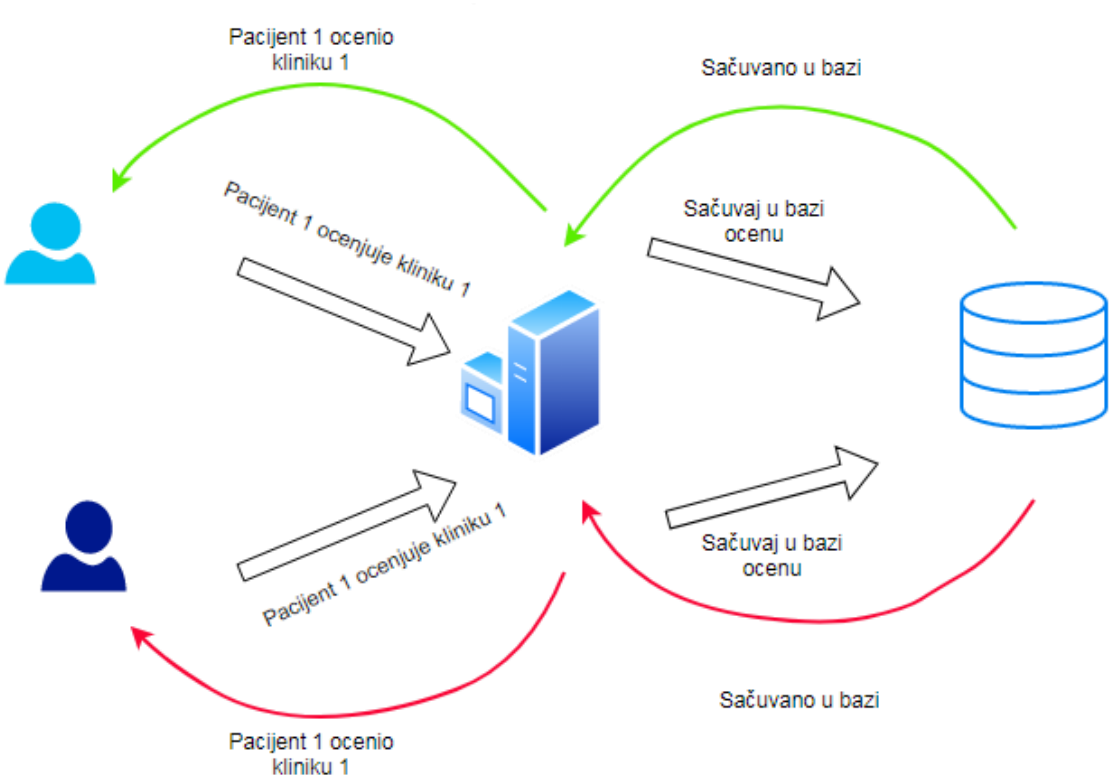


Slika 2 Tok zahteva i odgovora, konflikt 2

Rešenje: S obzirom da je za generisanje satnice neophodno uzeti u obzir zahteve za odmor, radno vreme lekara za taj dan, ukoliko radi, zakazane preglede i operacije koje ima da obavi, kao i predefinisane preglede koji su mu već dodeljeni, onda u trenutku odabira satnice može doći do više konflikata. Konflikti će se javiti ukoliko se kreira nov zahtev za odmor, izmeni radno vreme lekara, doda nov predefinisani pregled ili obradi prethodno kreirani zahtev za pregled/operaciju, kao i kada dva različita pacijenta žele da zakažu pregled kod istog lekara u istom ili terminu koji se preklapaju – što je prikazano na Slika 2. U tom slučaju ove konflikte bi rešilo pesimističko zaključavanje. Kada pacijetn šalje zahtev za zakazivanje pregleda na serveru se obrađuje taj zahtev tako što se šalje upit bazi za pronalaženje lekara koji će obaviti pregled. Lekar se pronalazi na osnovu metode `findByEmail(String email)` pa je neophodno dodati anotaciju `@Lock`, uz odabir tipa zaključavanja, na tu metodu repozitorijuma koja će vratiti objekat za zaključavanje. Pesimističko zaključavanje garantuje ispravan rad ali sa druge strane narušava performanse.

Konflikt 3: Korisnik je sa jednim nalogom ulogovan na dva uređaja, što posmatramo kao dva korisnika sa istim nalogom. Problem se javlja kod dodele ocene prvi put za kliniku i/ili lekara.

Tok zahteva i odgovora:



Slika 3 Tok zahteva i odgovora, konflikt 3

Rešenje: Recimo da je korisnik na oba uređaja na stranici za ocenjivanje. Korisnik 1 želi da prvi put oceni kliniku/lekara i korisnik 2 nad istim nalogom želi da oceni kliniku/lekara. Tada bi se u bazi desilo da za isti nalog i kliniku/lekara imamo dve ocene. Ovaj konflikt je rešen na sledeći način. Nakon slanja zahteva za čuvanjem unete ocene proverava se da li prethodno postoji stara ocena. Međutim to nije dovoljno. Potrebno je da u trenutku upisivanja u bazu ne postoji ocena koja povezuje taj nalog i kliniku/lekara. Zbog toga uvodim ograničenje - constraint nad tabelom ocena_klinika i ocena_lekar. Ovo ograničenje obezbeđuje jedinstvenost para nalog - klinika i nalog - lekar u konkretnom slučaju pacijent_id - klinika_id i pacijent_id - lekar_id.