TEMA : Konkurentni pristup resursima bazi

STUDENT : Slađana Savković, RA37/2017

PREDMET : Internet softverske arhitekture

Fakultet tehničkih nauka

2020/2021.

Opis problema

U aplikaciji su rješavane konflikte situacije konkurentnog pristupa bazi, prilikom definisanja novog termina za doktora (dermatologa/farmaceuta) od strane administratora apoteke, pacijenta ili doktora, zakazivanje predefinisanog termina kod dermatologa od strane pacijenta i dermatologa i smanjivanje količine propisanog lijeka nakon obavljenog pregleda pacijenta od strane doktora. Da bi se bolje shvatilo izloženo rješenje, potrebno je napomenuti način na koji se doktori i termini skladište u bazi podataka. Svi doktore, opisuje klasa *Doctor* za koju se kreira istoimena tabela u bazi podataka, a razlikovanje dermatologa od farmaceuta se vrši na osnovu vrijednosti polja *typeOfDoctor* (Dermatologist, Pharmacist). Termin je opisan klasom *Appointment*, skladišti se u istoimenoj tabeli, a njegovo stanje opisuje atribut status (Created, Scheduled, Canceled, Finished, Unperformed). Razlikovanje termina kod dermatologa i farmaceuta se zasniva isključivo na stranom ključu *doctorId*, prenijetom iz tabele doktor, na osnovu koga je moguće odrediti vrstu doktora sa kojim je termin povezan. Na osnovu ovakvog modela, implementirana logika funkcionalnosti koje su u vezi sa doktorom, je takođe zasnovana na principu da „nije svjesna“ na kojeg doktora se odnosi, pa je izbjegnuto ponavljanje istih metoda za dermatologa i farmaceuta.

1. Pregledi koji su unaprijed definisani ne smiju biti rezervisani od strane više različitih korisnika

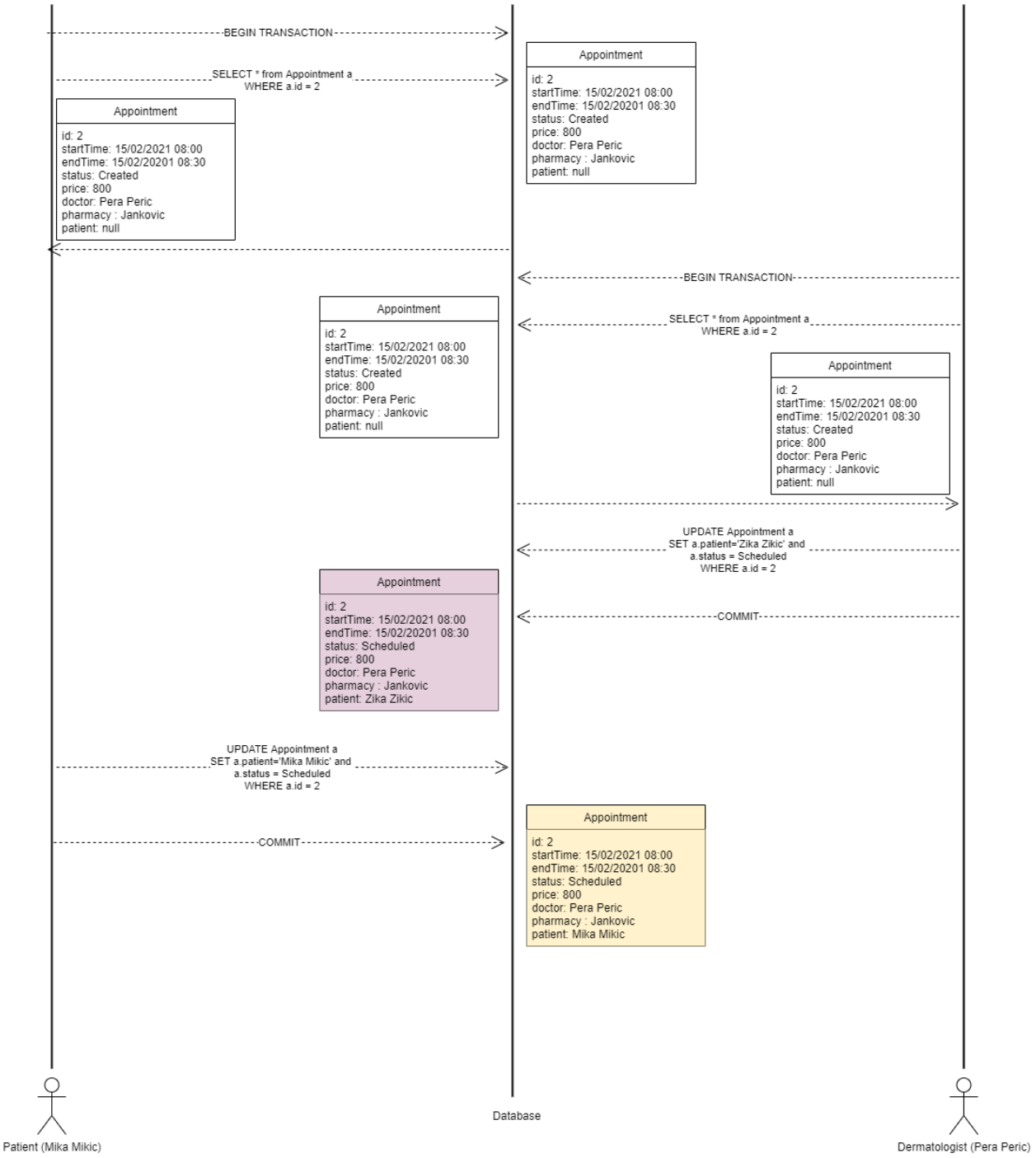
Do ove konfliktne situacije dolazi kada dva ili više razlicitih korisnika pokuša da zakaže isti termin koji je unaprijed definisan za određenog dermatologa. Mogući korisnici su pacijenti i dermatolozi. Na početku svaki korisnik dobavi istu verziju definisanog pregleda, ali prije nego što jedan korisnik rezerviše dati pregled (setuje pacijenta i status pregleda), u međuvremenu drugi korisnik već to obavi i sačuva podatke o pregledu. Zatim prvi korisnik pregazi verziju koju je ažurirao drugi korisnik i na taj način, poslednji korisnik koji dođe do resursa, ažurira ga i sačuva u bazi.

Ovaj problem je rješavan uz pomoć optimističkog zaključavanja resursa koji je podložan istovremenoj izmjeni od strane više korisnika. U klasu *Appointment*, dodano je polje anotirano sa *@Version*, pa se prije svakog ažuriranja pregleda u bazi (poziv metode *save()*), vrši automatska provjera da li je verzija pregleda koju korisnik želi da izmijeni, ista ona koja se nalazi u bazi. U slučaju da su podaci o pregledu u međuvremenu ažurirani, prijavljuje se greška korisniku.

Pretpostavka koja je korišćena jeste da će se u realnosti rijetko dešavati slučajevi da dva ili više korisnika istovremeno žele da rezervišu isti termin, za razliku od češće situacije kada više korisnika istovremeno želi da zakaže različite termine. Odabir optimističkog zaključavanja ovdje predstavlja optimalnije rješenje, jer je izbjegnuta situacija zaključavanja resursa koji se jako često koristi.

Pored toga što je u klasu *Appointment* dodano polje verzije, servisna metoda *void schedulePredefinedAppointment(int id, int patientId)* anotirana je kao transakciona sa propagacijom setovanom na vrijednost *REQUIRES_NEW* kako bi se uvijek pokrenula nova transakcija prilikom poziva metode. U metodi se dobavlja termin na osnovu id-a, zatim se vrši provjera da li ga je neko već zakazao i da li je pacijent za koga se pregled zakazuje, slobodan u izabranom terminu. Ako su svi uslovi ispunjeni, setuju se polja i pregled se čuva u bazi. Ova servisna metoda je pozvana iz istoimene metode *AppointmentController*-a, gdje su obrađeni izuzeci koji mogu da se jave pri pozivu servisne metode. End point koji se gadja je http://localhost:8080/api/appointment/{id}/patient/{patientId}/schedule.

U klasi *SchedulePredefinedAppointmentTest* je napisan test koji demonstrira konkurentno rezervisanje pregleda od strane dvije niti i bacanje *OptimisticLockingFailure* izuzetka.



Slika 1. Konfliktna situacija istovremenog zakazivanja predefinisanog pregleda od strane različitih korisnika

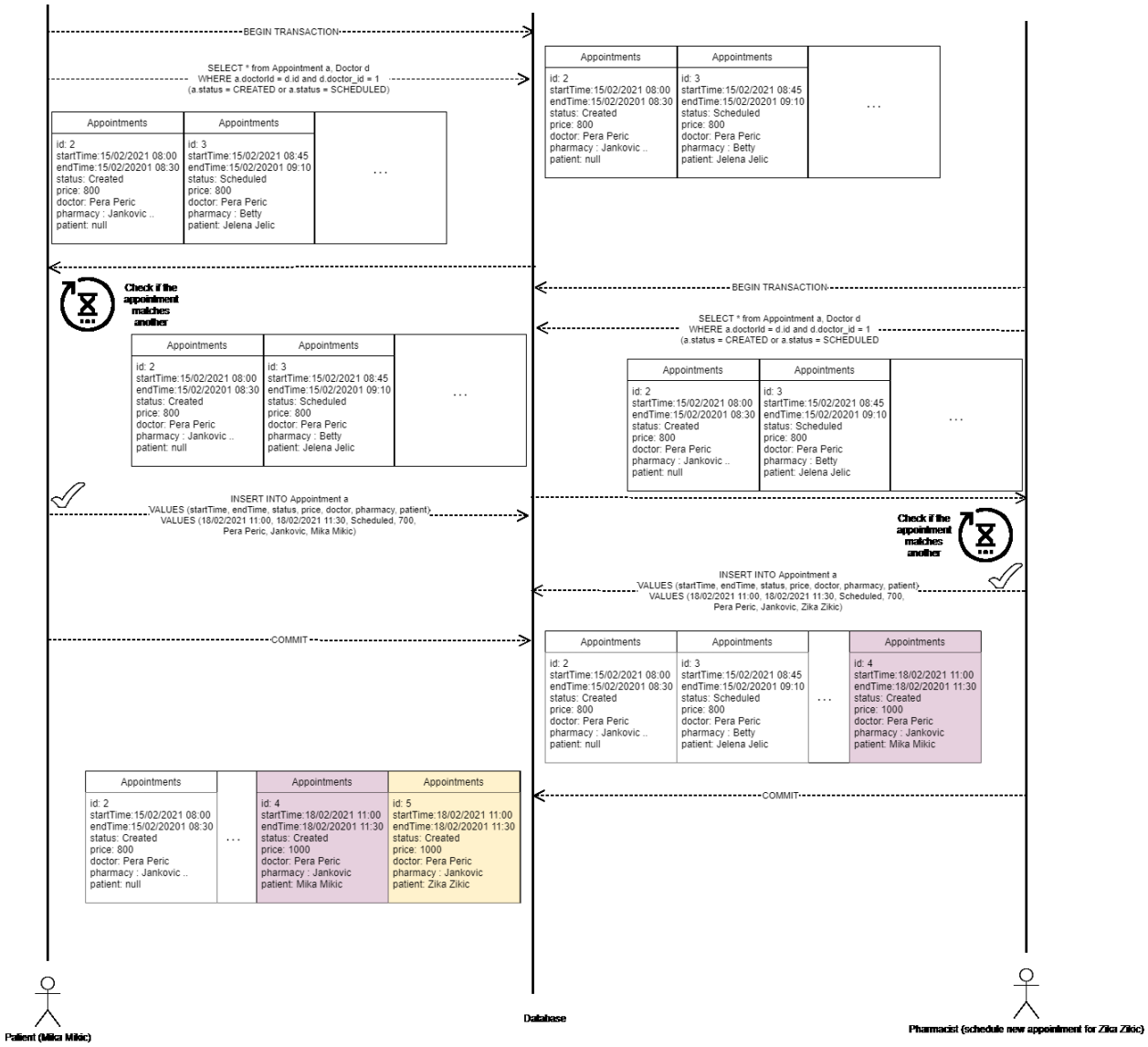
2. Više istovremenih korisnika aplikacije ne može da zakaže savjetovanje u istom terminu kod istog farmaceuta (termini se ne smiju ni preklapati) i jedan farmaceut ne može istovremeno da bude prisutan na više različitih savjetovanja

Do konfliktne situacije dolazi kada više korisnika pokuša u isto vrijeme da zakaže novi termin kod istog farmaceuta. Korisnik može da bude pacijent, ali i farmaceut koji za nekog pacijenta zakazuje novi termin kod sebe. Svaki od korisnika pročita podatke o zakazanim terminima kod farmaceuta, izvrši provjeru da li se neki od termina poklapa sa postojećim terminom pacijenta/farmaceuta i ako su svi uslovi ispunjeni, termin se dodaje u bazu. U opisanoj situaciji, vrši se dodavanje nove torke u tabelu što ne predstavlja dijeljeni resurs koji je moguće zaključati, pa je korišćen pristup zaključavanja pomoćnog resursa koji se koristi prilikom provjere uslova o dodavanju novog savjetovanja.

U *Appointment* repozitorijumu rađeno je pesimističko zaključavanje metode kojom se iz baze dobavlja kolekcija svih zauzetih termina određenog doktora. Pri tome zaključavanje je "ekskluzivno" pa je onemogućeno i čitanje i promjena objekta koji je zaključan. Pored toga, postavljen je timeout od 0s kojim se naglašava da korisnik ne čeka na zadati resurs, ako ga već neko "drži" jer postoji šansa da se obavi dodavanje nove torke (termina). Iako ovaj pristup daje lošije performanse u odnosu na optimističko zaključavanje, garantuje ispravan rad prilikom dodavanja novih savjetovanja.

Servisna metoda u kojoj se dešava logika oko provjera uslova i zakazivanja termina je *void checkDoctorAvailabilityAndAddAppointment(…)* u *AppointmentService* klasi, a end point koji se gađa prilikom iniciranja zahtjeva je http://localhost:8080/api/appointment/schedule.

U klasi *CreateNewAppointmentTest* napisan je test koji demonstrira konkurentno kreiranje termina od strane dvije niti i bacanje *PesimisticLockingFailure* izuzetka.

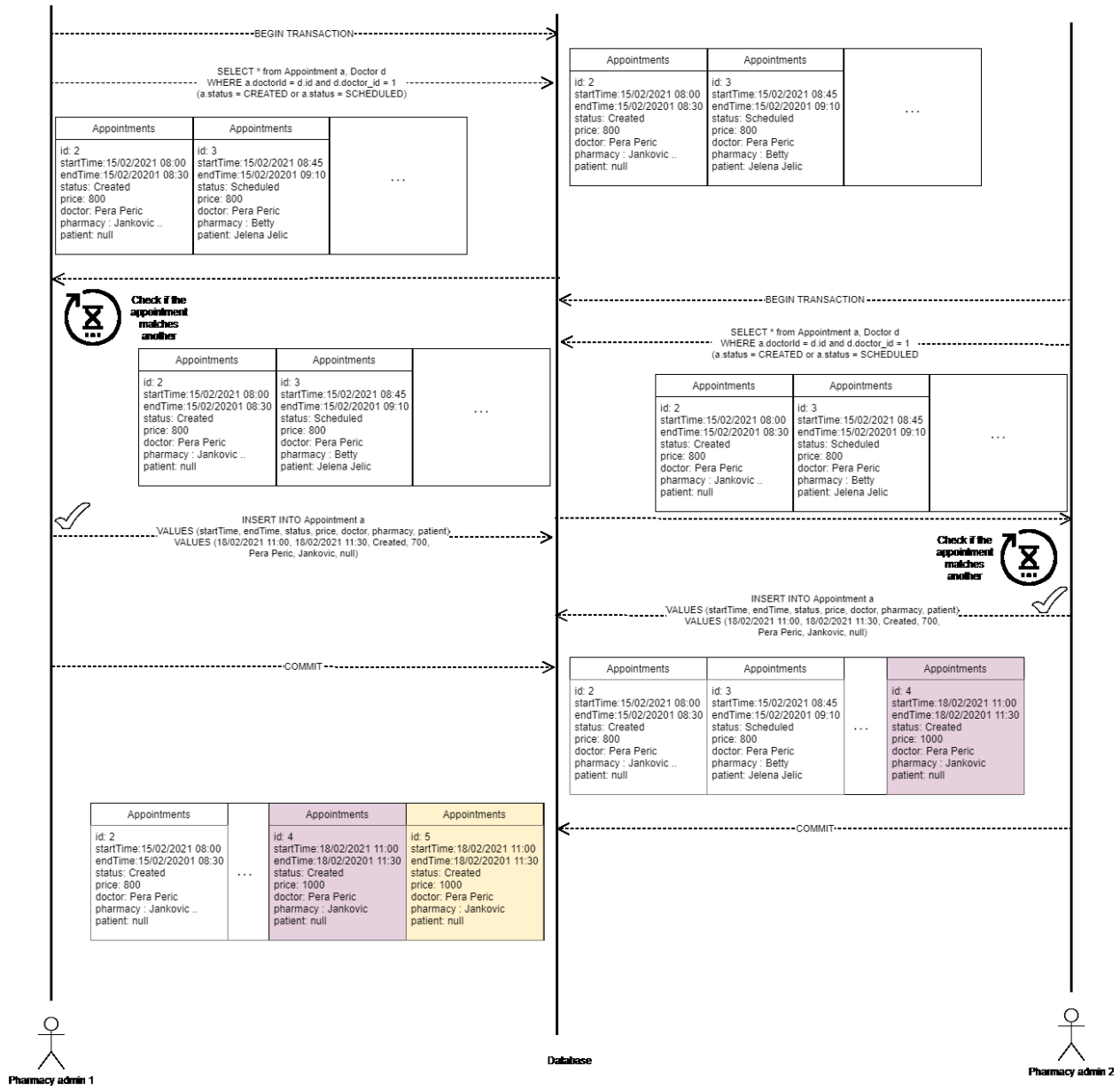


Slika 2. Konfliktna situacija istovremenog zakazivanja istog termina kod farmaceuta od strane više korisnika

3. Jedan dermatolog ne može istovremeno da bude prisutan na više različitih pregleda

Do konfliktne situacije dolazi kada više korisnika pokuša da zakaže novi termin kod dermatologa. Korisnici mogu biti pacijenti, dermatolozi, ali i administrator apoteke koji definiše slobodne termine za dermatologe. Kao što je u uvodu već opisano, sistem je projektovan na način da metode koje su zadužene za kreiranje novih termina ili zakazivanje postojećih, nisu svjesne koji ih korisnik poziva, kao ni to na koju vrstu doktora se odnosi termin. Kao posljedica fleksibilnosti sistema, rješenje ove konfliktne situacije je već podržano nakon rješavanja prethodne dvije.

Pacijenti i dermatolozi gađaju end point http://localhost:8080/api/appointment/schedule dok je za administratora apoteke obezbijeđen http://localhost:8080/api/appointment/create. Jedina vidljiva razlika između termina koje administrator unaprijed definiše, jesu polja status koje je postavljeno na vrijednost *Created* i pacijent koji je null. Za razliku od toga, prilikom zakazivanja novog termina od strane pacijenta i dermatologa, status termina je *Scheduled*, a polje pacijenta je postavljeno na baš onog pacijenta za kojeg se zakazivanje obavlja.



Slika 3. Konfliktna situacija istovremenog kreiranja istog termina kod dermatologa od strane više administratora

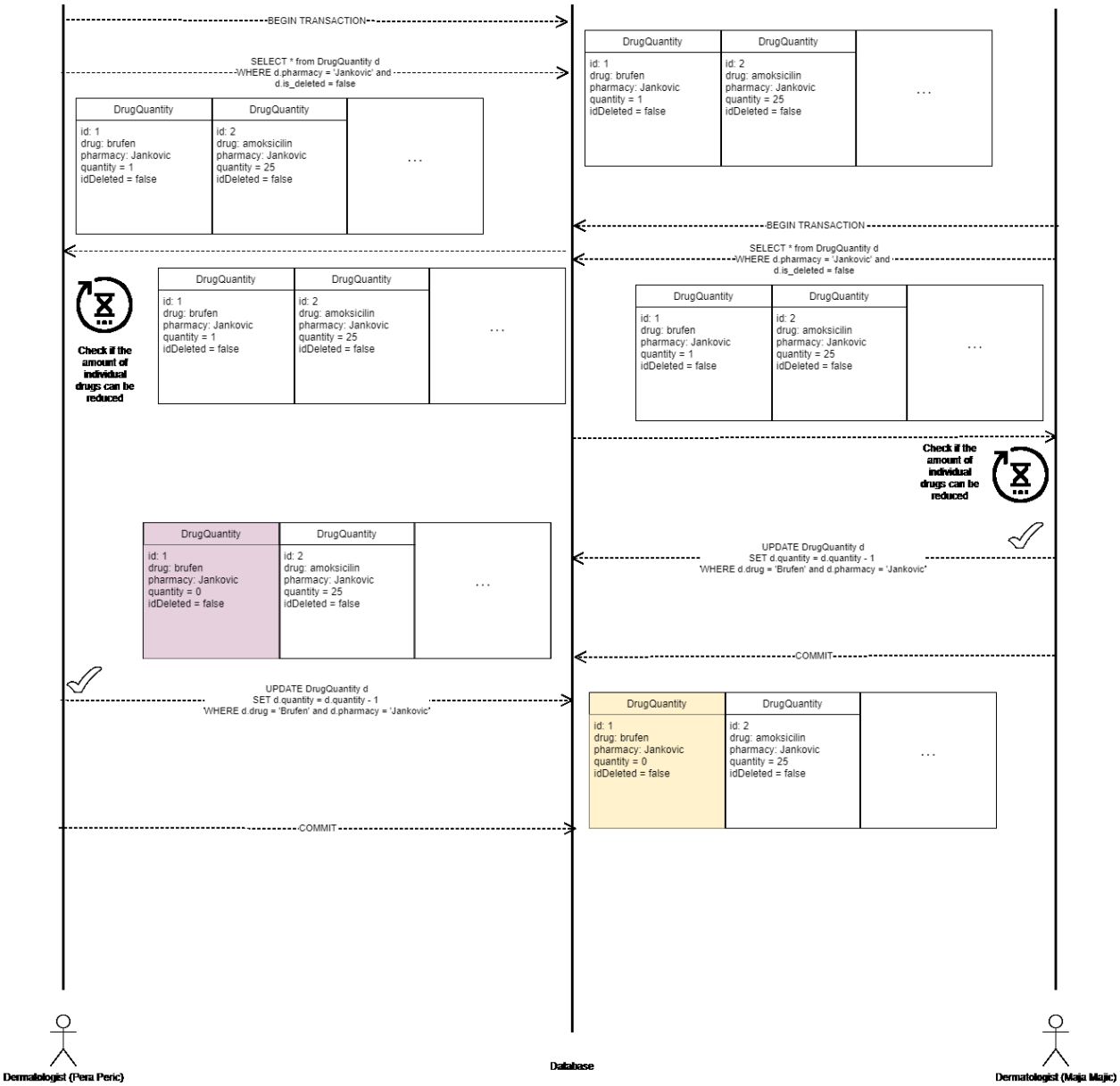
4. Ažuriranje stanja lijeka u apoteci u situaciji istovremenog pokušaja smanjivanja količine nakon obavljenog pregleda od strane doktora

Opisana konfliktna situacija se dešava u slučaju da je više doktora (dermatologa ili farmaceuta) pokušalo istovremeno da ažurira količinu istog lijeka u apoteci nakon obavljenog pregleda. Pri tome, moguća su dva ishoda:

a) Lijek je dostupan nakon obavljenog pregleda (postoji na stanju), ali je njegova količina u bazi neispravna u odnosu na realno stanje u apoteci -> više doktora je istovremeno smanjilo količinu lijeka, ali u odnosu na staro stanje, iako je u međuvremenu neko već ažurirao količinu
b) Lijek nije dostupan nakon obavljenog pregleda (nema ga na stanju) i njegova količina u bazi je 0 (ispravno stanje), ali je prepisan više od jednom pacijentu -> više doktora je istovremeno smanjilo količinu lijeka sa 1 na 0 i prepisalo ga pacijentu

Slično kao za prvu konfliktnu situaciju, za problem je iskorišćeno optimističko zaključavanje resursa koji je podložan izmjeni, što je u ovom slučaju objekat klase *DrugQuantityPharmacy*. U servisnoj metodi *Collection<DrugQuantityPharmacy> reduceDrugQuantitiesOrReturnMissingDrugs(…)* nalazi se logika oko smanjivanja količine prepisanih lijekova. Prvo se upućuje zahtjev prema bazi podataka koja vraća dostupne lijekove u apoteci, zatim se za svaki prepisan lijek smanjuje količina tog lijeka u apoteci i na kraju se ažurirane količine čuvaju u bazi. Pored toga, metoda vraća koji od prepisanih lijekova više ne postoji na stanju (ako takvi postoje), o čemu se obavještava doktor kako bi mogao prepisati neki od dostupnih lijekova. Prilikom upisa u bazu, vrši se provjera da li je je verzija koju korisnik pokušava promijeniti ista kao ona u bazi i na taj način se osigurava ažuriranje najsvježijih podataka. Ovom metodom koordiniše *add(...)* metoda *ExaminationReport* kontrolera koja je zadužena za čuvanje izvještaja sa pregleda. U slučaju da je smanjivanje količine lijeka onemogućeno, kontroler odmah vraća poruku doktoru. Tek kada smanjivanje količine bude moguće, izvještaj sa pregleda i propisana terapija se čuvaju u bazi. Ovim pristupom, baza je uvijek u validnom stanju i spriječene su situacije kada su izvještaj i terapija uspješno sačuvani, a stanje lijeka nije ažurirano. End point kojem se upuće zahtjev od strane dermatologa/farmaceuta je http://localhost:8080/api/examination-report.

U klasi *ReduceDrugQuantityTest* napisan je test koji demonstrira konkurentno ažuriranje količine lijeka u apoteci od strane dvije niti i bacanje *OptimisticLockingFailure* izuzetka.



Slika 4. Konfliktna situacija istovremenog ažuriranja količine istog lijeka u istoj apoteci od strane više dermatologa/farmaceuta