Konfliktne situacije student 3 - Aleksandar Stevanović

1. Pregledi koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika

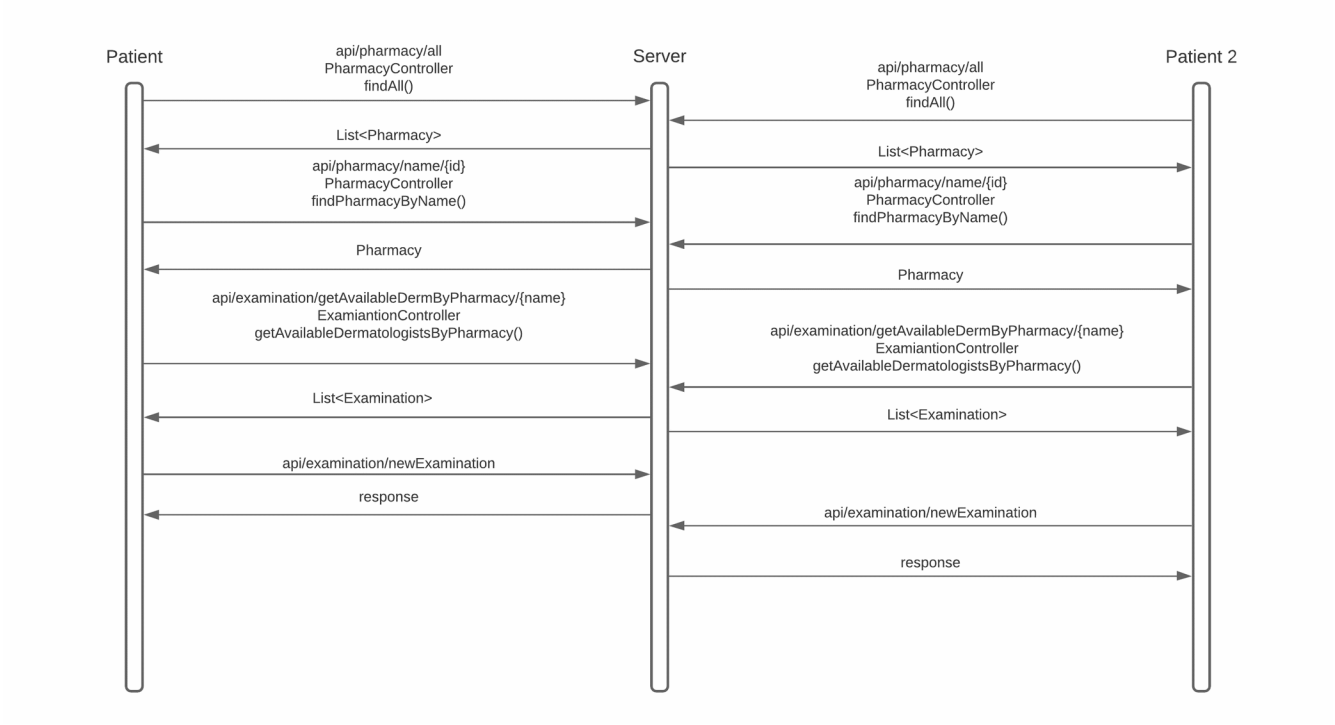
Opis:

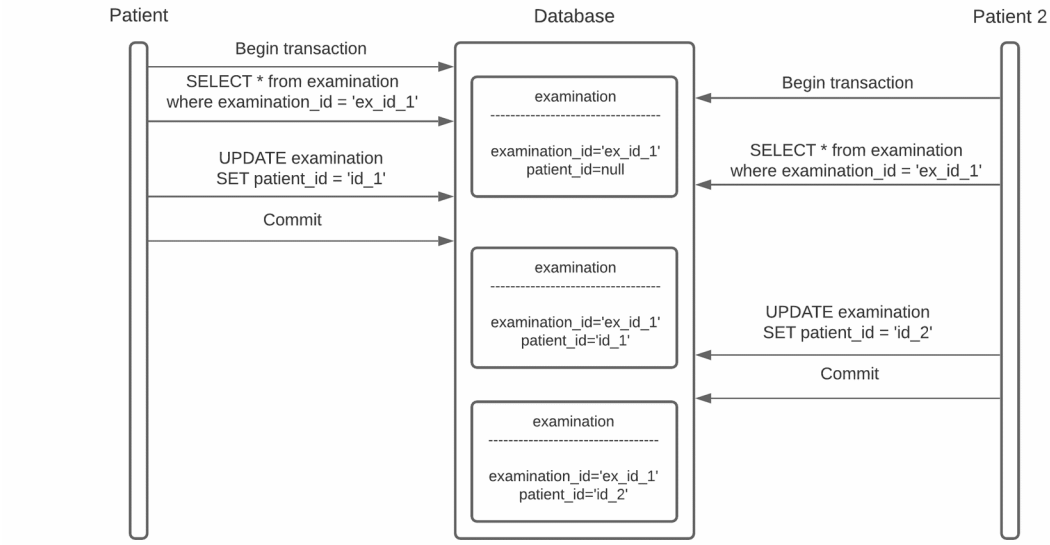
U modelu naše aplikacije, predefinisani pregledi definisani su od strane administratora apoteke, tako što on odabere zaposlenog, odnosno dermatologa za kog želi da kreira pregled, a zatim i datum, vreme i trajanje pregleda, kao i cenu. Ono što taj pregled razdvaja od zakazanog pregleda je to što je pacijent u okviru pregleda nedostajuća vrednost (*null*). Korisnici našeg sistema (pacijenti) imaju na svojoj početnoj stranici opciju za zakazivanje pregleda. Nakon njenog odabira bivaju preusmereni na listu svih apoteka, gde biraju onu u okviru koje žele da zakažu pregled. U okviru apoteke biraju neki od gore pomenutih unapred definisanih termina, zajedno sa samim dermatologom. Gde klikom na isti vrše zakazivanje.

Do same konfiktne situacije, upravo može doći baš na tom mestu: ukoliko više različitih pacijenata u isto vreme odabere isti termin i pokuša da ga zakaže.

Klikom na dugme za zakazivanje, korisnik šalje zahtev serveru, tačnije "gađa" *api/examination/newExamination/* end point u okviru **ExaminationController**-a, odnosno metodu *scheduleExamination()*. Nakon toga poziva se **PatientService**, odnosno njegova metoda *findByEmail()* koja dobavlja pacijenta na osnovu prosleđenog email-a. Dalje se obračunavaju popusti i pogodnosti koje pacijent ima na dati pregled. A zatim se pozivaju metode *findAllByPatient()* i *findByEmployeePharmacyTimeDate()* **ExaminationService**-a, da bi se proverilo eventualno preklapanje već postojećih termina dermatologa i pacijenta sa odabranim terminom. Ako su sve provere uspešne, pregled se zakazuje i trajno čuva u bazi, upotrebom metode *save()* **ExaminationService**-a pacijentu se nazad vraća zakazan pregled i stiže mu potvrdan email, u suprotnom server baca izuzetak.

Dijagram toka:





Oba korisnika započinju transakciju neposredno jedan za drugim. Čitaju podatke koji su u tom trenutku nepromenjeni, a zatim prvi korisnik vrši zakazivanje pregleda i komituje svoje izmene. Što dovodi do toga da su podaci koje drugi korisnik u tom trenutku poseduje nekonzistentni. Nakon toga drugi korisnik nesvestan promena koje su u međuvremenu nastale, takođe vrši zakazivanje pregleda i time "pregazi" promene koje je načinio prvi korisnik i dovodi sistem u nekonzistentno stanje.

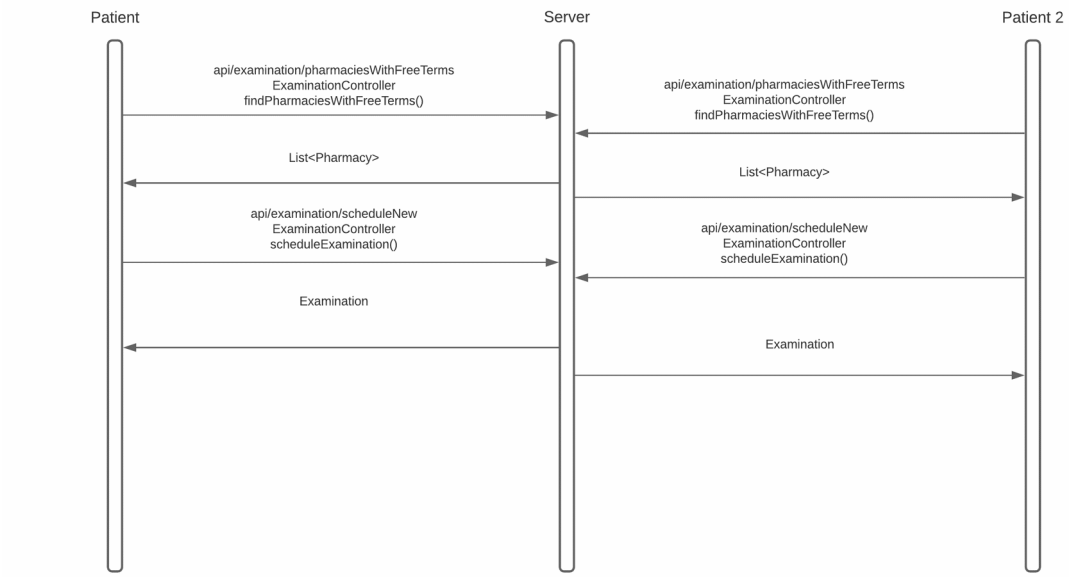
Rešenje:
Kako bi se opisana situacija (Lost update) izbegla, a sistem ostao u konzistentnom stanju, primenjena je strategija optimističkog zaključavanja resursa tako što je u klasu Examination dodato polje version tipa Long, naznačeno anotacijom @Version, u kom čuvamo trenutnu verziju izmene reda u tabeli Examinations, odnosno našeg konkretnog Examination-a. Dodata je takođe i anotacija@Transactional(readOnly = false) iznad metode *scheduleExamination()*, **Examination Controllera**, u kojoj je sama logika i implementirana. Nakon navedenih izmena, kada prvi korisnik započne transakciju i zakaže pregled, pokušaj drugog korisnika da zakaže isti rezultiraće greškom. Do toga dolazi zato što se u međuvremenu dogodila promena i polje version se uvećalo za 1, što je jasan indikator da je neko drugi već načinio promenu odnosno zakazivanje pregleda.

2. Više istovremenih korisnika aplikacije ne može da zakaže savetovanje u istom terminu kod istog farmaceuta (termini se ne smeju ni preklapati)

Opis:
U modelu naše aplikacije. Korisnici našeg sistema (pacijenti) imaju na svojoj početnoj stranici opciju za zakazivanje savetovanja. Nakon njenog odabira bivaju preusmereni na formu gde mogu izvršiti savetovanje. U okviru forme biraju datum i vreme za koje žele da zakažu pregled, nakon čega im se nudi lista apoteka koje imaju slobodnog farmaceuta u tom terminu. Odabirom apoteke, vrši se zakazivanje ukoliko su svi uslovi ispunjeni, tj. ako nema preklapanja sa drugim terminima.

Popunjavanjem i potvrdom forme, korisnik šalje zahtev serveru, tačnije "gađa" *api/examination/pharmaciesWithFreeTerms/* end point u okviru **ExaminationController**-a, odnosno metodu *findPharmaciesWithFreeTerms()*. Nakon toga poziva se **ExaminationService**, odnosno njegova metoda *findPharmaciesWithFreeTerm()* koja dobavlja savetovanja za navedeni termin. Zatim se kroz ista iterira i "kupe" se apoteke za koje su savetovanja vezana. Vraća se odgovor u vidu liste apoteka pacijentu. Pacijent zatim odabirom apoteke vrši rezervaciju termina tako što šalje zahtev ka serveru. "Gađa" end point *api/examination/scheduleNew* u okviru **ExaminationController**-a, odnosno metodu *scheduleExamination()*. Nakon toga, pozivaju se **PatientService** i **ExaminationService** kako bi se pribavili termini savetovanja i proverilo preklapanje, a zatim ako je sve u redu *save()* metoda **ExaminationService**-a, a pacijentu se nazad vraća zakazan pregled i stiže mu potvrdan email, u suprotnom server baca izuzetak.

Dijagram toka:



Kao i u prethodnom slučaju, oba korisnika započinju akcije neposredno jedan za drugim. Čitaju podatke koji su u tom trenutku nepromenjeni, a zatim prvi korisnik vrši zakazivanje savetovanja u određenom terminu i potvrđuje svoje izmene. Što dovodi do toga da su podaci koje drugi korisnik u tom trenutku poseduje nekonzistentni. Nakon toga drugi korisnik nesvestan promena koje su u međuvremenu nastale, takođe vrši zakazivanje savetovanja čije vreme se potencijalno može preklapati sa promenama koje je načinio prvi korisnik i dovodi sistem u nekonzistentno stanje, jer jedan farmaceut ne može imati termine koji se preklapaju.

Rešenje:

Kako bi se opisana situacija izbegla, a sistem ostao u konzistentnom stanju, primenjena je strategija pesimističkog zaključavanja resursa tako što je na sve dosadašnje promene(version polje u klasi Examination), dodate anotacije, @Transactional(readOnly = false) iznad metode *scheduleExamination()*, **Examination Controllera,** kao i @Lock(LockModeType.PESSIMISTIC_READ)) iznad *findByUserId()* metode **MedicalStuffRepository**-a, kao i @QueryHints({QueryHint(name="javax.persistence.lock.timeout", value="2000")}), koja doprinosi tome da transakcija ne bude odmah odbijena, već da se na odgovor čeka određen vremenski period izražen u ms. U ovom slučaju 2000 ms, odnosno 2s. Na ovaj način zabranjuje se čitanje resursa, sve dok postoji druga transakcija, koja je prethodno pristupila tom istom resursu. Samim tim zabranjen je pristup tom farmaceutu, kao i sve ostalo što nakon toga sledi (zakazivanje termina i dodavanje u radni kalendar farmaceuta). Čime je problem rešen, jer se ostatak provera vrši programski.

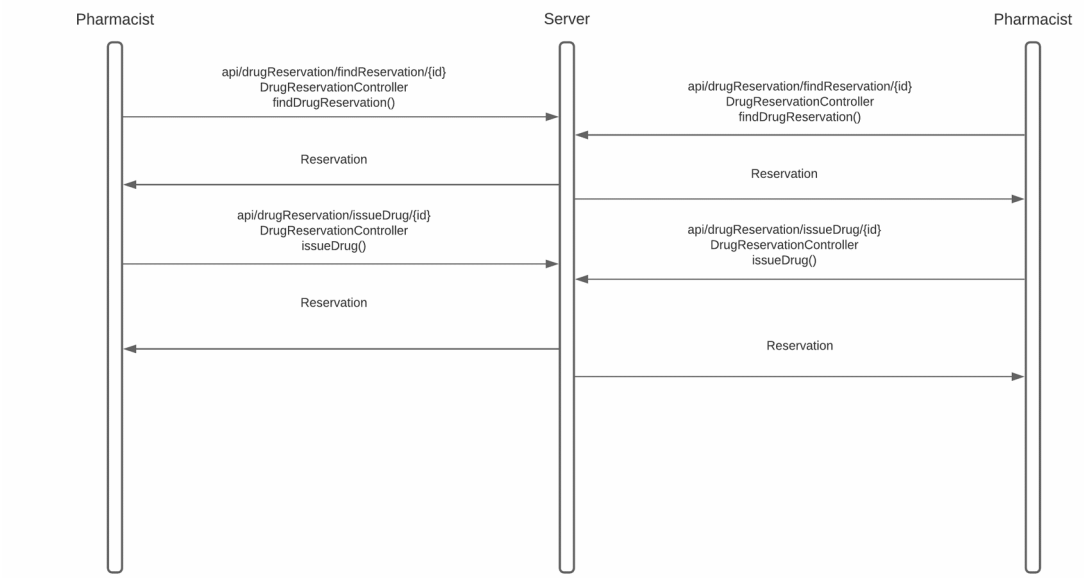
3. Više farmaceuta ne može istovremeno izdati isti lek rezervisan od strane pacijenta

Opis:

U našoj aplikaciji, farmaceuti imaju mogućnost da vrše izdavanje rezervisanih lekova koje su pacijenti rezervisali, ukoliko rezervacija nije istekla ili ukoliko lek već nije preuzet. Funkcionalnost je dostupna sa glavne stranice dermatologa. Klikom na "Issue a medicament" otvoriće se forma za izdavanje.

U formu je potrebno uneti kod rezervacije, a zatim potražiti rezervaciju sa tim kodom ako postoji. To se postiže upućivanjem get zahteva od klijenta ka serveru, gde se "gađa" *api/drugResevation/findReservation/{id}* end point u okviru **DrugReservationController**-a, odnosno metoda *findDrugReservation()*. Nakon toga poziva se **PharmacistService**, odnosno njegova metoda *findByEmail()* koja dobavlja farmaceuta, kako bi se izvršila provera da li farmaceut koji je uputio zahtev za pronalaženje rezervacije, radi u apoteci u okviru koje je rezervacija kreirana. Zatim se poziva metoda *findDrugReservationByPatient()* **DrugReservationService**-a. Nakon čega se farmaceutu vraćaju rezultati pretrage. Zatim se ukoliko je rezervacija sa datim kodom pronađena farmaceutu nudi opcija za izdavanje leka, klikom na koju se pozivaju se ponovo upućuje zahtev od klijenta ka serveru, ali ovog puta se "gađa" *api/drugResevation/issueDrug/{id}* end point u okviru **DrugReservationController**-a, odnosno metoda *issueDrug()*. Poziva se metoda *issueDrug()* **DrugReservationService**-a, u kojoj se status rezervacije postavlja na završenu rezervaciju i vrši se čuvanje rezervacije nazad u bazu. Pacijentu se ako je sve prošlo po planu šalje mejl sa potvrdom preuzimanja leka.

Dijagram toka:



Do konfliktne situacije potencijalno može doći u slučaju da dva farmaceuta istovremeno probaju da izdaju istu rezervaciju.

Rešenje:
Kako bi se opisana situacija izbegla, a sistem ostao u konzistentnom stanju, primenjena je strategija optimističkog zaključavanja resursa tako što je u klasu User dodato polje version tipa Long, naznačeno anotacijom @Version, u kom čuvamo trenutnu verziju izmene reda u tabeli Users, odnosno našeg konkretnog User-a. Isto je urađeno i u klasi DrugReservation. Dodata je takođe i anotacija@Transactional(readOnly = false) iznad metode *issueDrug()*, **DrugReservationService**-a, u kojoj je sama logika i implementirana. Nakon navedenih izmena, kada prvi farmaceut započne transakciju i izda rezervaciju, pokušaj drugog farmaceuta da izda istu rezultiraće greškom. Do toga dolazi zato što se u međuvremenu dogodila promena i polje version se uvećalo za 1, što je jasan indikator da je neko drugi već načinio promenu odnosno izdavanje leka.