Konkurentni pristup resursima u bazi

Primer 1.

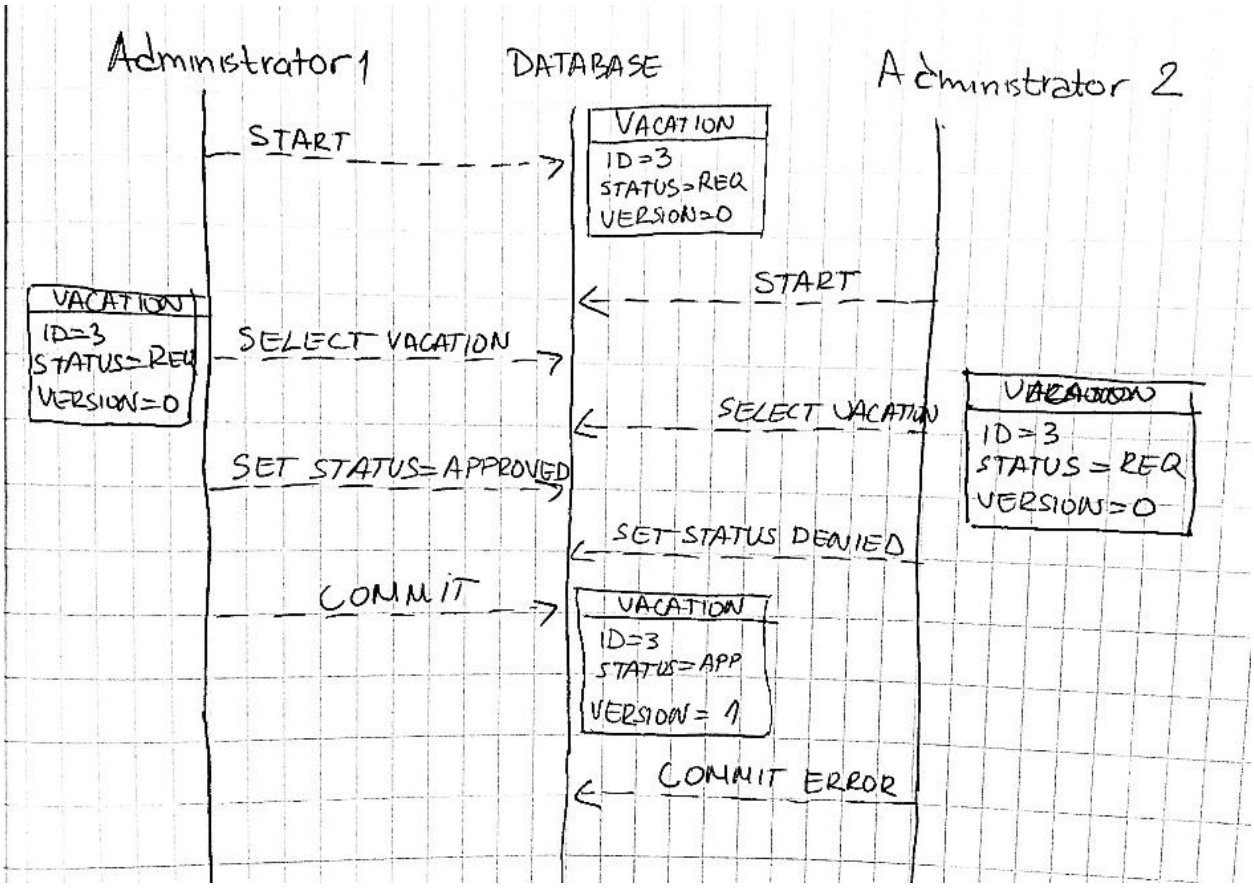
Opis: Dva administratora apoteke pokušavaju da pristupe istom zahtevu za godišnji odmor koji se nalazi u bazi. Jedan želi da odbije zahtev, a drugi da prihvati.

Endpoint:

http://localhost:8051/sysAdmin/approveVacation,
http://localhost:8051/sysAdmin/denyVacation

Problem: Nije moguće istovremeno prihvatiti i odbiti zahtev, javlja se lost update problem.

Rešenje: Optimističko zaključavanje - dodato polje za kontrolu verzija u klasi *Vacation* koje će se inkrementirati nakon svake promene stanja resursa. Pre izvršavanja transakcije, proverava se da li je verzija izmenjena u odnosu na onu kod preuzetog resursa. Ukoliko se verzije razlikuju, transakcija neće biti izvršena. U ovom slučaju, razlikovaće se status zahteva za godišnji odmor, te će administrator apoteke biti sprečen da ponovo prihvati/odbije zahtev ili da prihvati već odbijen zahtev i obrnuto.



Primer 2.

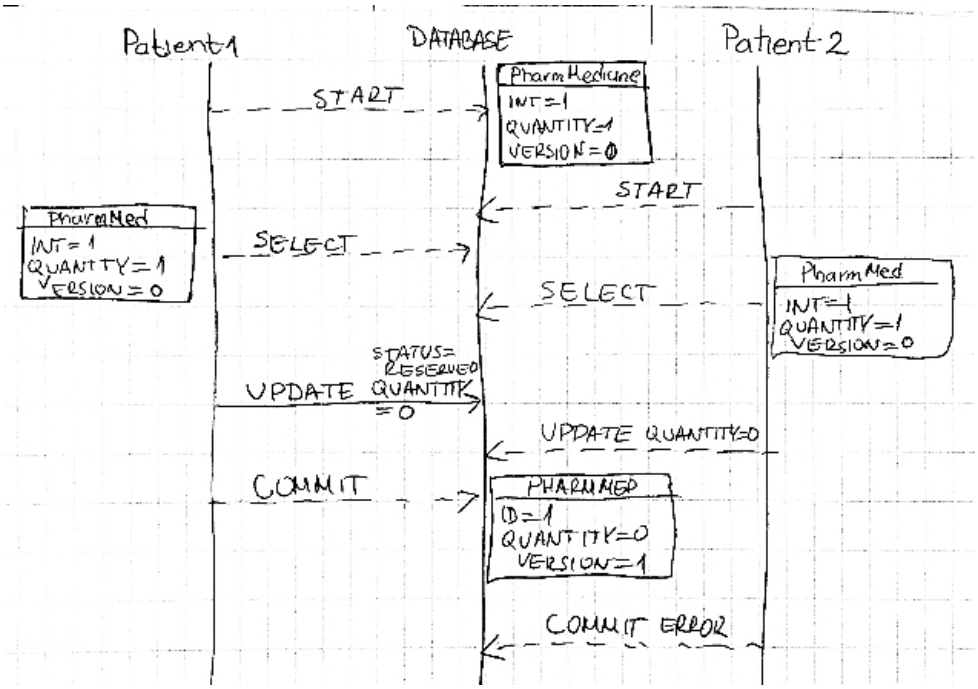
Opis: Dva pacijenta pokušavaju da rezervišu isti lek u apoteci, a količina leka na stanju je 1.

Endpoint:

http://localhost:8051/medicine/reserveMedicine

Problem: Nije moguće rezervisati lek koji u tom trenutku ne postoji u ponudi - lost update problem

Rešenje: Optimističko zaključavanje - dodato polje za kontrolu verzija u klasi *PharmacyMedicine* koje će se inkrementirati nakon svake promene stanja resursa, u ovom slučaju, količine. Pacijent neće moći da rezerviše lek ukoliko je verzija povećana u odnosu na preuzetu, te se transakcija neće izvršiti.



Primer 3.

Opis: Pacijent pretražuje apoteke u sistemu, dok istovremeno administrator sistema registruje novu apoteku

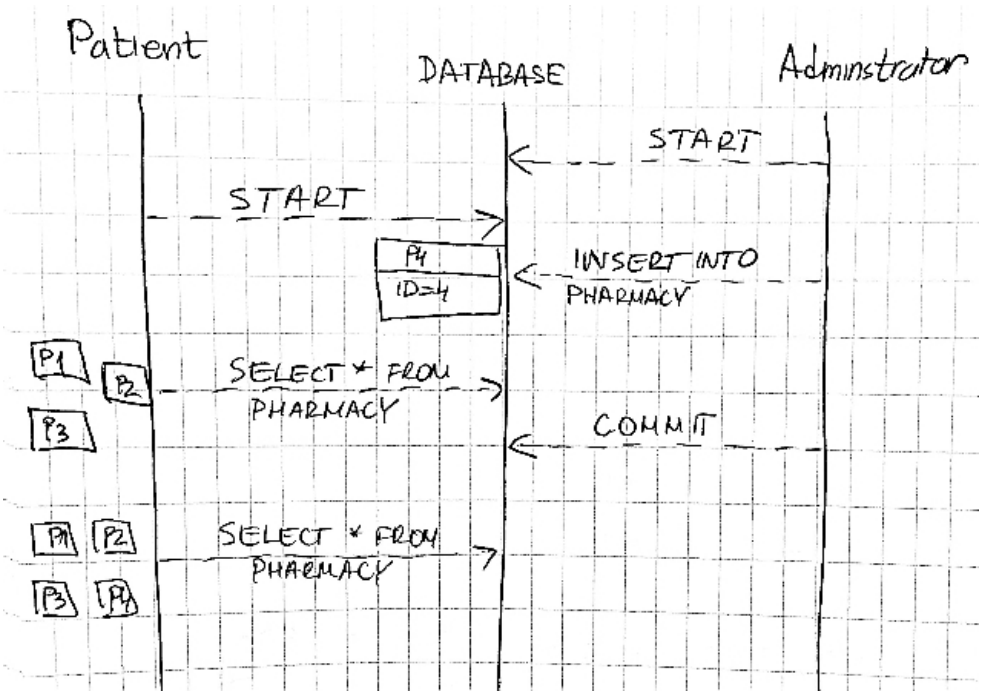
Endpoint:

http://localhost:8051/pharmacy/searchPharmacy,

http://localhost:8051/pharmacy/createNewPharmacy

Problem: Prilikom ponovnog slanja istog zahteva, pacijent će dobiti različit odgovor od servera ukoliko apoteka koju je administrator sistema registrovao zadovoljava kriterijume iz zahteva - phantom read problem

Rešenje: Definiše se *serializable* nivo izolacije tako da administratoru sistema neće biti dozvoljena registracija nove apoteke koja zadovoljava kriterijume iz zahteva pacijenta sve dok se transakcija koju je započeo pacijent ne završi.



Primer 4.

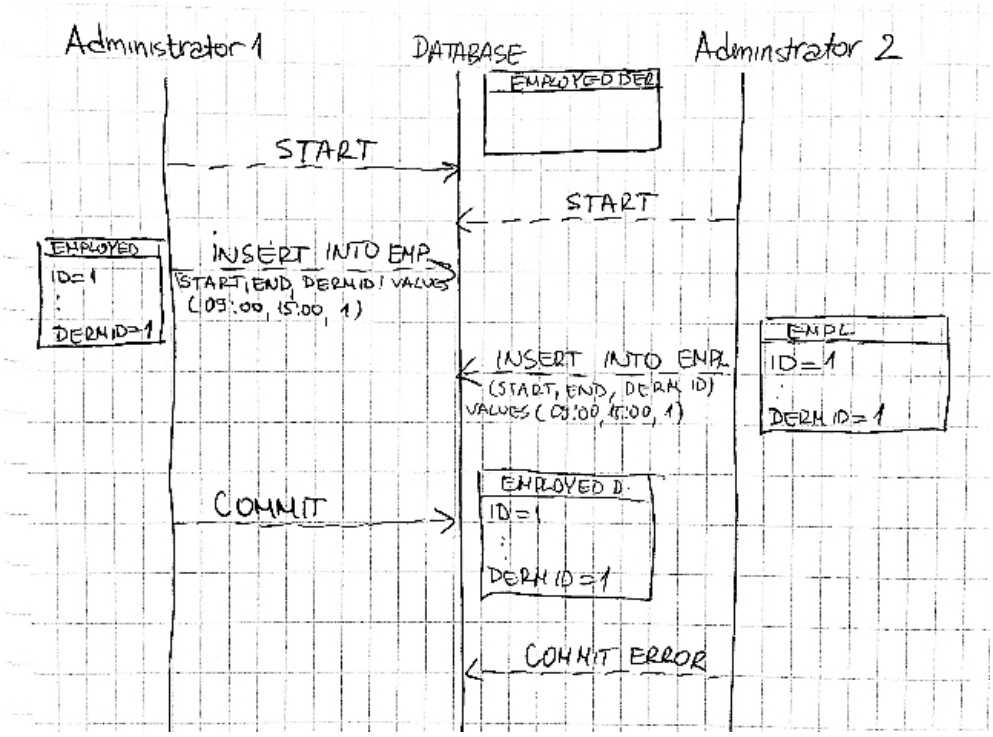
Opis: Dva administratora različitih apoteka pokušavaju da dodaju istog dermatologa kao zaposlenog u svojoj apoteci sa istim radnim vremenom.

Endpoint:

http://localhost:8051/dermatologist/addDermatologist

Problem: Implementacija programa ne dozvoljava da jedan dermatolog bude zaposlen u različitim apotekama sa istim radnim vremenom - lost update problem

Rešenje: U ovom slučaju, deljeni resurs je dermatolog. Konflikt je sprečen optimističkim zaključavanjem - uz pomoć polja *version,* koje se povećava za 1 svaki put kada se izvrši promena nad resursom, prati se stanje istog. Transakcija neće biti moguća ukoliko se vrednost polja razlikuje u odnosu na početnu.



Primer 5.

Opis: Pacijent pristupa lekovima koje apoteka ima u ponudi u trenutku kada administrator apoteke uklanja lek iz ponude.

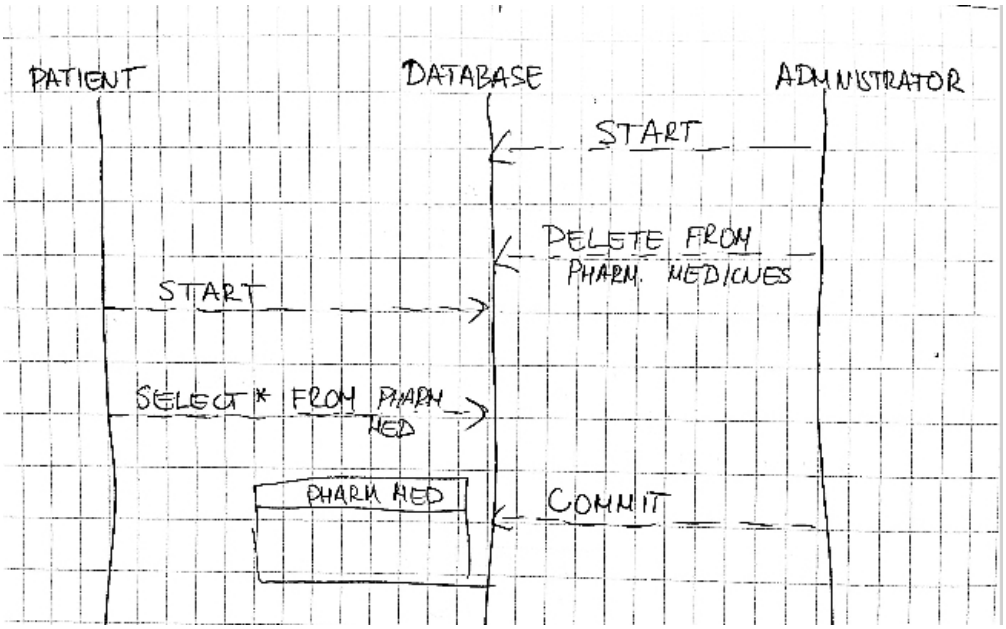
Endpoint:

http://localhost:8051/medicine/getPharmacyMedicines

http://localhost:8051/medicine/removeMedicine

Problem: Ukoliko se prilikom izvršenja transakcije administratora apoteke desi rollback, pacijentu će se prikazati podaci, odnosno lekovi, koji ne postoje u bazi (ponudi apoteke) - dirty read problem.

Rešenje: Postavljanjem nivoa izolacije na READ_COMMITED eliminiše se dirty read problem. Pacijent neće moći da vidi ponudu lekova iz apoteke sve dok administrator apoteke ne završi svoju transakciju.



Primer 6.

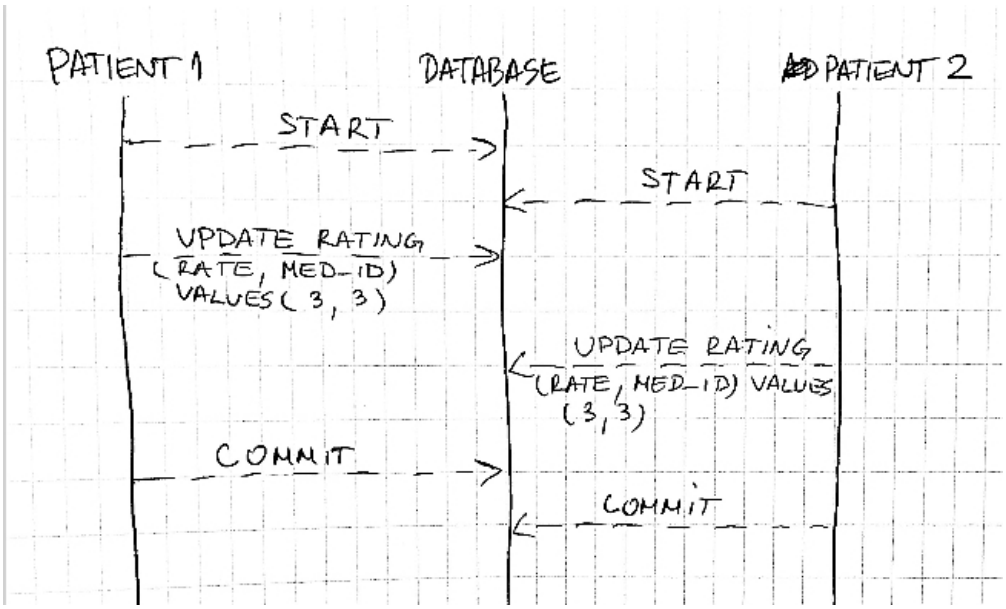
Opis: Dva pacijenta žele da ostave ocenu za isti resurs (apoteka, dermatolog, farmaceut ili lek).

Endpoint:

http://localhost:8051/rating/rateUser
http://localhost:8051/rating/ratePharmacy
http://localhost:8051/rating/rateMedicine

Problem: Može doći do konfklita prilikom izračunavanja srednje ocene jer resursu pristupa više korisnika i samim tim izračunata prosečna ocena može biti pogrešna - lost update problem.

Rešenje: Optimističko zaključavanje - dodato je polje version u klasi *Rating* koje se inkrementira nakon svake izmene resursa. Pre izvršavanja transakcije, proverava se da li je verzija izmenjena u odnosu na onu kod preuzetog resursa. Ukoliko se verzije razlikuju, transakcija neće biti izvršena.


PATIENT 1
DATABASE
PATIENT 2
START
START
UPDATE RATING
(RATE, MED_ID)
VALUES (3, 3)
UPDATE RATING
(RATE, MED_ID) VALUES
(3, 3)
COMMIT
COMMIT