KONKURENTNOST

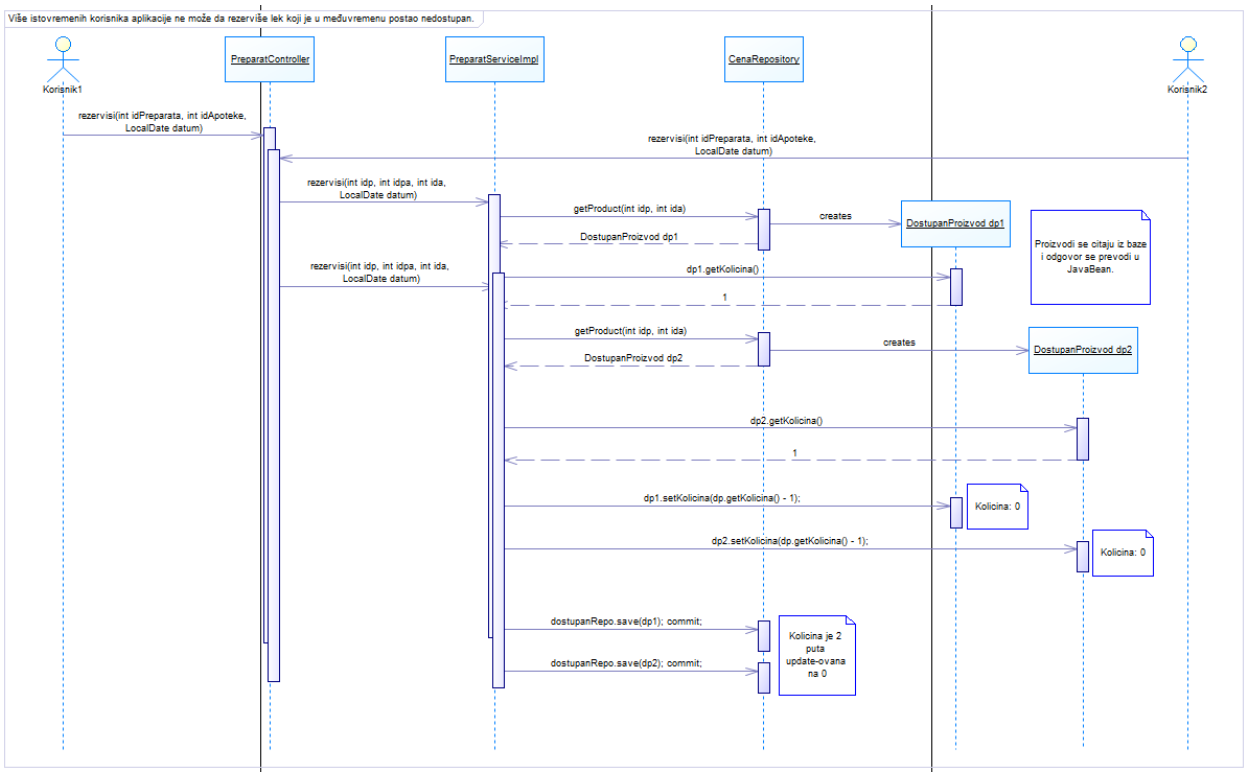
ISA _2021-tim6

Ivan Mršulja
SW-65-2018

***Napomena:** Sve servisne klase su anotirane sa @Transactional anotacijom, tako da se pomenute *commit* i *rollback* operacije obavljaju nad čitavom metodom a ne samo nad pojedinačnim upitom (što ne bi imalo smisla). Takođe, ovim putem se doseg pesimističkog zaključavanja prenosi na čitavo tijelo metode.

Scenario 1:

- Više istovremenih korisnika aplikacije ne može da rezerviše lijek koji je u međuvremenu postao nedostupan.



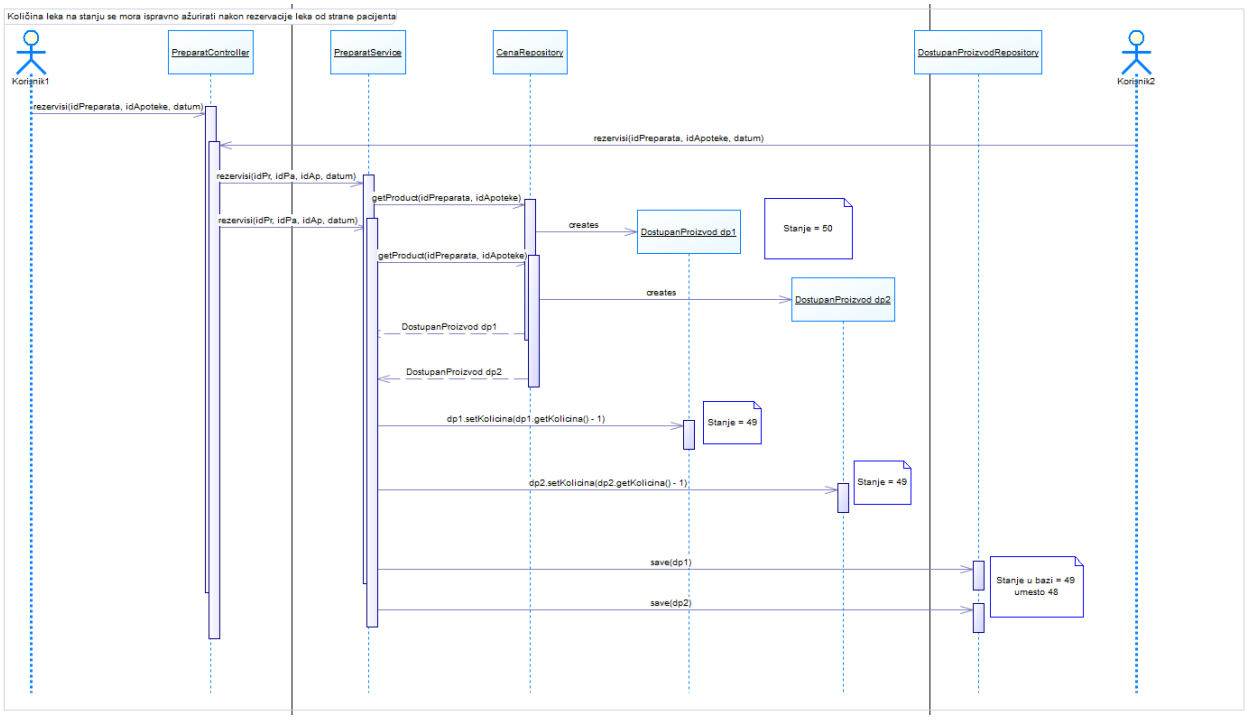
Dva korisnika istovremeno pokušavaju da rezervišu lijek kod kojeg je trenutno stanje u apoteci = 1. Prilikom čitanja iz baze, stanje je 1 za oba korisnika te provjera dostupnosti prolazi, oba stanja se lokalno ažuriraju na 0 i promjene se *commit*-uju na bazu. Ovim su oba korisnika uspjeli rezervisati lijek a finalno stanje u bazi je 0.

Rješenje:

Uveden je mehanizam optimističkog zaključavanja. U klasi Rezervacija, dodat je atribut *version* koji modeluje verziju trenutno dobavljene instance. Prilikom svakog ažuriranja rezervacije i *commit*-a na bazu provjerava se da li je trenutna verzija objekta ista kao i verzija koja se nalazi u bazi, ukoliko jeste, verzija se povećava za 1, ukoliko nije (što znaci da trenutna verzija više ne oslikava realno stanje u bazi), dolazi do bacanja *OptimisticLockingFailureException*-a koji se hvata u kontroleru PreparatController, i koji dalje korisniku vraća poruku o grešci. Korisnik nakon toga može ponovo poslati zahtjev za kreiranje nove rezervacije. Također, kao dobra praksa, metoda *findOneById(int id)* u klasi RezervacijaRepository je anotirana sa *@Lock(LockModeType.OPTIMISTIC)*. Ova anotacija implicitno štiti od non repeatable read anomalije. Timeout je također eksplicitno podešen na 0 anotacijom.

Scenario 2:

- Količina lijeka na stanju se mora ispravno ažurirati nakon rezervacije lijeka od strane pacijenta.



Dva korisnika istovremeno pokušavaju da rezervišu lijek kod kojeg je trenutno stanje u apoteci = N. Prilikom čitanja iz baze, stanje je N za oba korisnika te provjera dostupnosti prolazi, oba stanja se lokalno ažuriraju na N-1 i promjene se *commit*-uju na bazu. Ovim su oba korisnika uspjeli rezervisati lijek a finalno stanje u bazi je N-1 umjesto N-2.

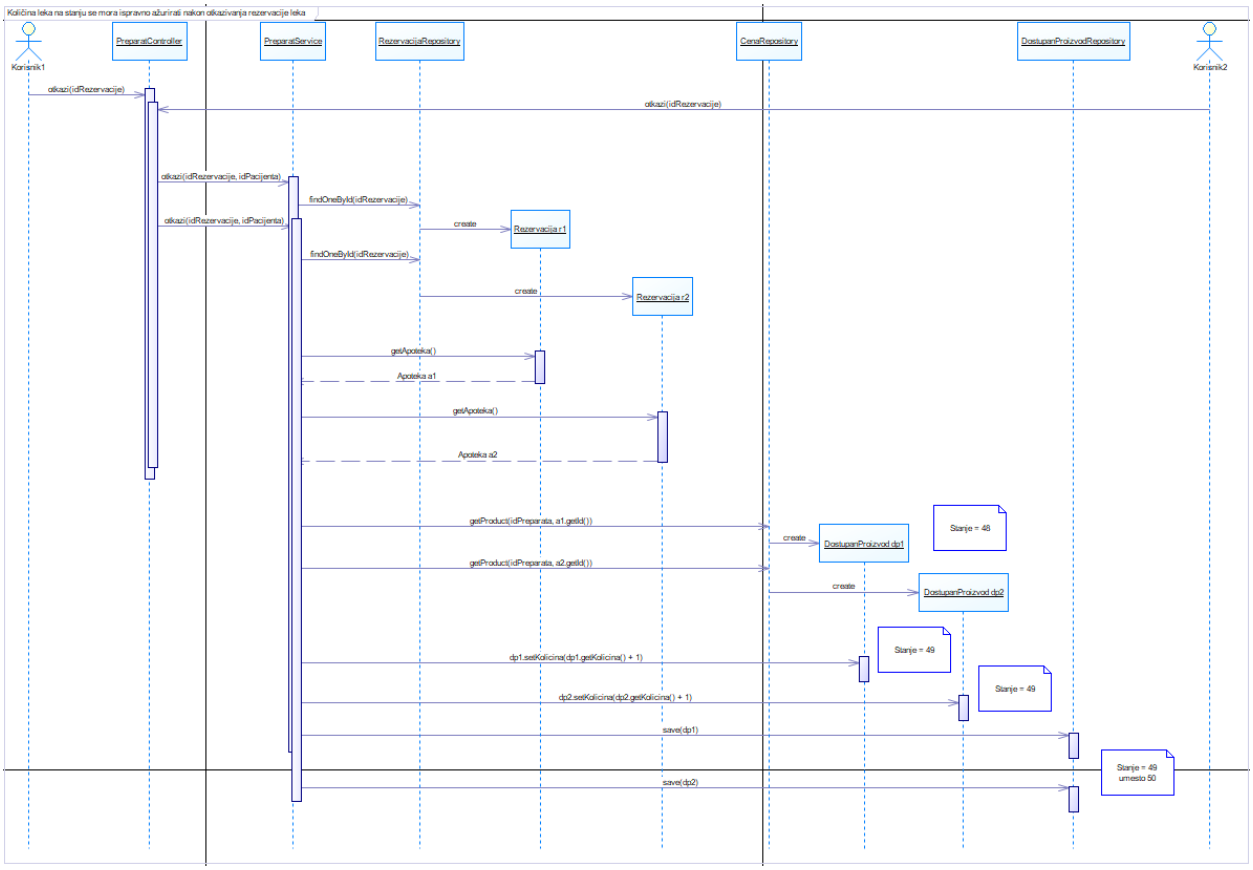
Rješenje:

Uveden je mehanizam optimističkog zaključavanja. U klasi Rezervacija, dodat je atribut *version* koji modeluje verziju trenutno dobavljene instance. Prilikom svakog ažuriranja rezervacije I *commit*-a na

bazu provjerava se da li je trenutna verzija objekta ista kao i verzija koja se nalazi u bazi, ukoliko jeste, verzija se povećava za 1, ukoliko nije (što znači da trenutna verzija više ne oslikava realno stanje u bazi), dolazi do bacanja *OptimisticLockingFailureException*-a koji se hvata u kontroleru PreparatController, i koji dalje korisniku vraća poruku o grešci. Korisnik nakon toga može ponovo poslati zahtjev za kreiranje nove rezervacije. Također, kao dobra praksa, metoda *findOneById(int id)* u klasi RezervacijaRepository je anotirana sa *@Lock(LockModeType.OPTIMISTIC)*. Ova anotacija implicitno štiti od non repeatable read anomalije. Timeout je također eksplicitno podešen na 0 anotacijom.

Scenario 3:

- Količina lijeka na stanju se mora ispravno ažurirati nakon otkazivanja rezervacije lijeka.



Dva korisnika istovremeno pokušavaju da otkazuju lijek kod kojeg je trenutno stanje u apoteci = N. Oba stanja se lokalno ažuriraju na N+1 i promjene se *commit*-uju na bazu. Ovim su oba korisnika uspjeli otkazati rezervaciju lijeka a finalno stanje u bazi je N+1 umjesto N+2.

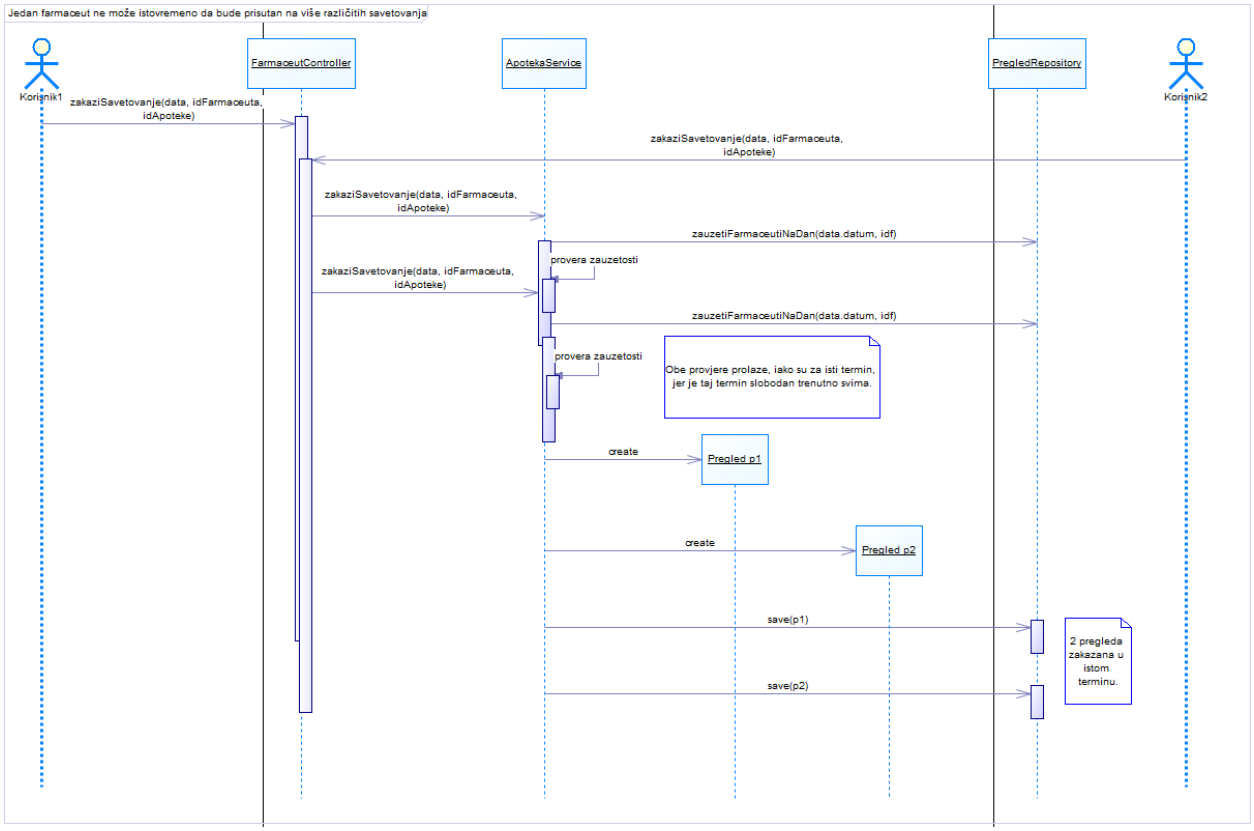
Rješenje:

Uveden je mehanizam optimističkog zaključavanja. U klasi Rezervacija, dodat je atribut *version* koji modeluje verziju trenutno dobavljene instance. Prilikom svakog ažuriranja rezervacije I *commit*-a na bazu provjerava se da li je trenutna verzija objekta ista kao i verzija koja se nalazi u bazi, ukoliko jeste,

verzija se povećava za 1, ukoliko nije (što znaci da trenutna verzija više ne oslikava realno stanje u bazi), dolazi do bacanja *OptimisticLockingFailureException*-a koji se hvata u kontroleru PreparatController, i koji dalje korisniku vraća poruku o grešci. Korisnik nakon toga može ponovo poslati zahtjev za otkazivanje rezervacije. Takođe, kao dobra praksa, metoda *findOneById(int id)* u klasi RezervacijaRepository je anotirana sa *@Lock(LockModeType.OPTIMISTIC)*. Ova anotacija implicitno štiti od non repeatable read anomalije. Timeout je takođe eksplicitno podešen na 0 anotacijom.

Scenario 4:

- Jedan farmaceut ne može istovremeno da bude prisutan na više različitih savjetovanja.



Dva korisnika u isto vrijeme pokušavaju da zakažu termin savjetovanja. Prilikom čitanja iz baze, farmaceut je obojici prikazan kao slobodan, te validacije prolaze. Lokalno se kreira novi termin savjetovanja i *commit*-uje se na bazu. Oba korisnika su uspjela da zakažu termin savjetovanja u isto vrijeme.

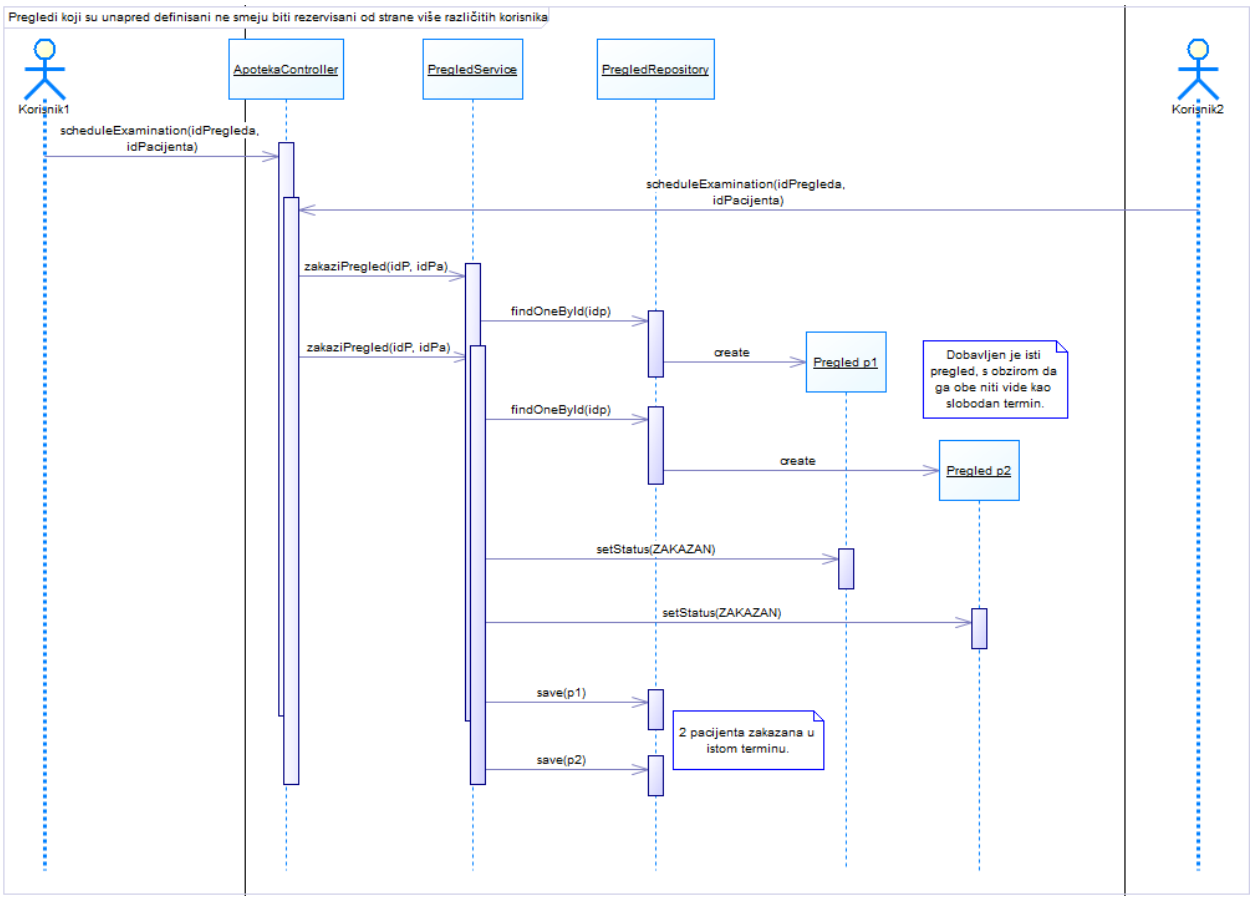
Rješenje:

Uveden je mehanizam pesimističkog zaključavanja. Prilikom dobavljanja farmaceuta metodom *findOneById(int id)* u klasi FarmaceutRepository vrši se pesimističko zaključavanje tabele farmaceuta (pessimistic read). Upotrebljen je pessimistic read kako bi drugi mogli čitati podatke o farmaceutu ali ukoliko se iz 2 (ili više) niti pokuša istovremeno ažuriranje tabele ona koja ne drži ključ će dobiti

PessimisticLockingFailureException. Ovaj izuzetak dalje hvata controller FarmaceutController koji korisniku vraća poruku da zakazivanje nije uspjelo.

Scenario 5:

- Pregledi koji su unaprijed definisani ne smeju biti rezervisani od strane više različitih korisnika.



Dva korisnika istovremeno šalju zahtjev za zakazivanje unaprijed definisanih termina pregleda. Prilikom čitanja iz baze status pregleda je SLOBODAN te validacije prolaze. Lokalno se status ažurira na ZAKAZAN I promjene se *commit*-uju na bazu. Oba korisnika su uspjela da zakažu isti termin pregleda kod dermatologa.

Rješenje:

Uveden je mehanizam optimističkog zaključavanja. U klasi Pregled, dodat je atribut *version* koji modeluje verziju trenutno dobavljene instance. Prilikom svakog ažuriranja rezervacije i *commit*-a na bazu provjerava se da li je trenutna verzija objekta ista kao i verzija koja se nalazi u bazi, ukoliko jeste, verzija se povećava za 1, ukoliko nije (što znači da trenutna verzija više ne oslikava realno stanje u bazi), dolazi do bacanja *OptimisticLockingFailureException*-a koji se hvata u kontroleru ApotekaController, i koji dalje korisniku vraća poruku o grešci. Takođe, kao dobra praksa, metoda *findOneById(int id)* klase PregledRepository je anotirana sa *@Lock(LockModeType.OPTIMISTIC)*. Ova anotacija implicitno štiti od non repeatable read anomalije. Timeout je takođe eksplicitno podešen na 0 anotacijom.