KONKURENTAN PRISTUP RESURSIMA U BAZI KONFLIKTNE SITUACIJE

Po specifikacijama zadaci studenta 1 su bili da reši sledeće konflikte situacije:

- više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta
- više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji

Dodatne konfliktne situacije koje su rešavane:

- više istovremenih klijenata ne može da pošalje zahtev za registraciju sa istom email adresom
- klijent ne može istovremeno da ažurira svoje podatke sa više različitih uređaja

→ **Za izvršavanje rezervacija I brzih rezervacija se pozivaju iste metode, pa se isto odnosi za konfliktne situacije pod tačkama 1 I 2.**

Konfliktne situacije 1 i 2

Opis

Klijenti mogu da rezervišu vikendice/brodove/instruktore. Ukoliko bi 2 klijenta u isto vreme vršila proveru dostupnosti entiteta obe provere bi vratile validan rezultat te bi nastao problem oko zauzetosti ako bi uneseni termin bio isti ili bar preklapajući. Ovo predstavlja problem dodavanja novih torki u isto vreme koje zajedno dovode do nevalidnog stanja u bazi.

Solucija

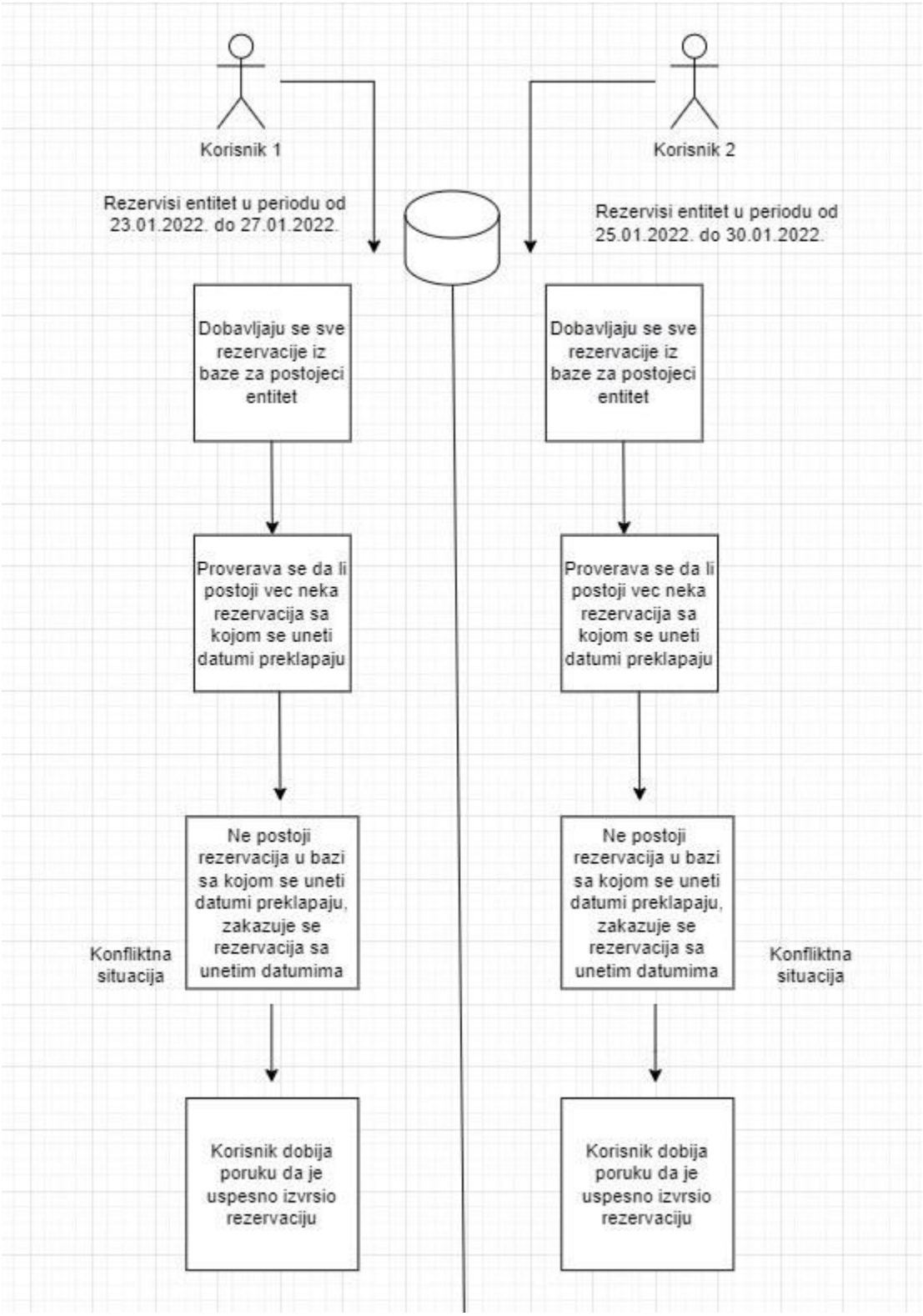
Problem možemo rešiti tako što su u klasama servisa postavljene *@Transactional* anotacije iznad metoda *add(HomeReservationDTO dto, Long clientId), add(BoatReservationDTO dto, Long clientId), add(AdventureReservationDTO dto, Long clientId)* i u sklopu njih odrađeno je rukovanje izuzecima, u skladu sa tim korisniku će biti prikazane odgovarajuće poruke. Takođe odrađeno je pesimističko zaključavanje.

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public HomeReservation add(HomeReservationDTO dto) throws Exception{
```

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public BoatReservation add(BoatReservationDTO dto) throws Exception {
```

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
@Override
public AdventureReservation add(AdventureReservationDTO dto) {
```

Dijagram toka



Konfliktna situacija 3

Opis

Kada korisnik pravi nalog sa email adresom, prvo se izvršava provera o jedinstvenosti unesenog emaila, u slučaju da je email zauzet korisniku će biti javljena greška u vidu poruke da adresa već postoji. U situaciji da dva korisnika istovremeno pokušavaju da pošalju zahtev za registraciju sa iste email adrese, može se desiti da oba zahteva prodju, što bi dovelo do nevalidnosti u bazi.

Solucija

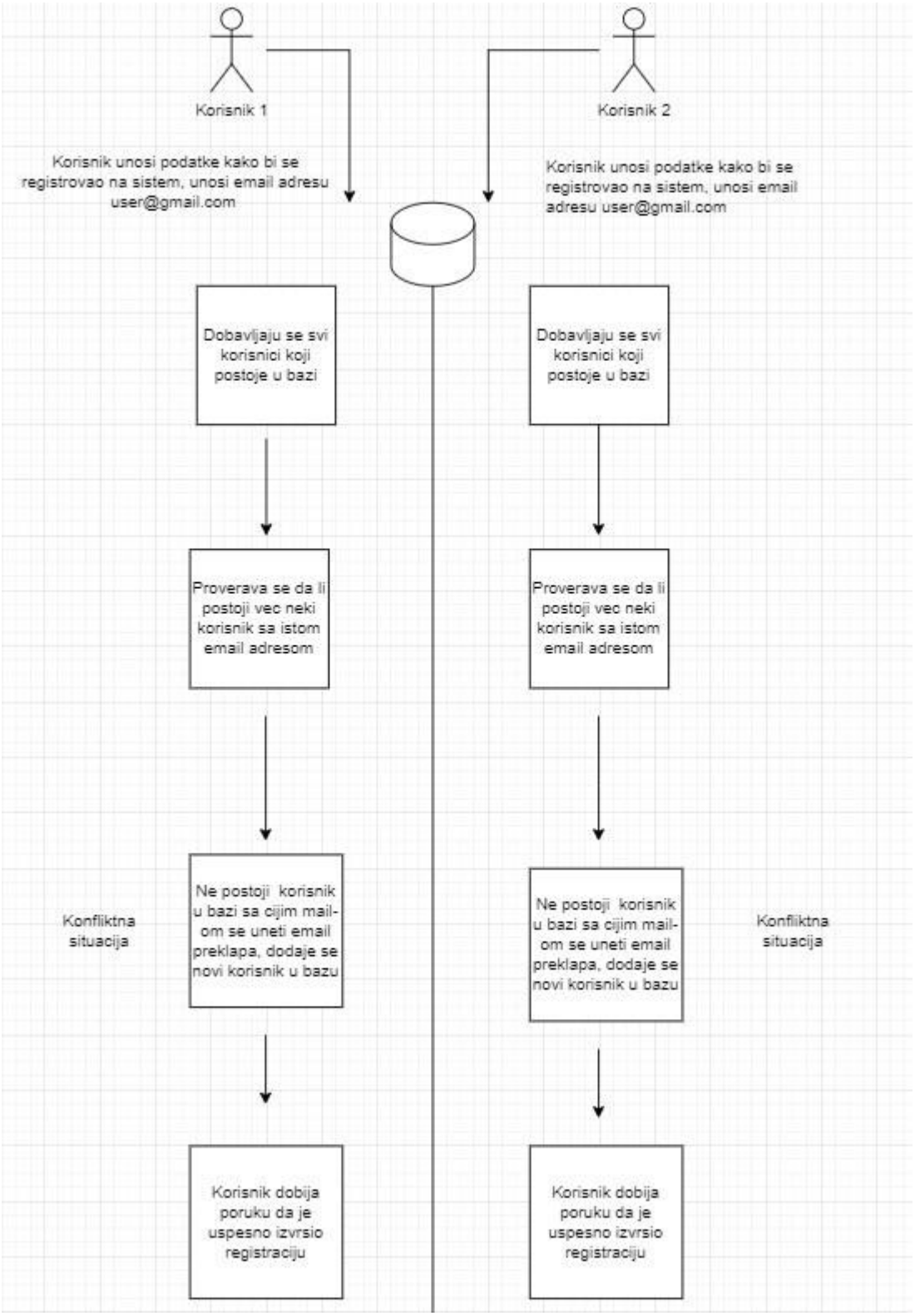
Problem možemo rešiti tako što su u klasama servisa postavljene *@Transactional* anotacije iznad metoda *clientRegistration(RegistrationDTO registrationDTO), houseOwnerRegistration(RegistrationDTO registrationDTO)*, *boatOwnerRegistration(RegistrationDTO registrationDTO)* I izvršeno je pesimističko zaključavanje baze i dolazi do *PessimisticLockingFailureException*-a.

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public User clientRegistration(RegistrationDTO registrationDTO) {
```

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public User houseOwnerRegistration(RegistrationDTO registrationDTO) {
```

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public User boatOwnerRegistration(RegistrationDTO registrationDTO) {
```

Dijagram toka



Konfliktna situacija 4

Opis

Korisnik može biti prijavljen na svoj nalog sa više uređaja I može da ažurira informacije o svom nalogu. Problem bi nastao ako bi se istovremeno sa dva uređaja menjali podaci, oba zahteva bi prošla I došlo bi do nevalidnog stanja u bazi.

Solucija

Problem možemo rešiti tako što su klasama odgovarajućih servisa postavljene *@Transactional* anotacije iznad metoda *edit(UserDTO userDTO)*

Dijagram toka