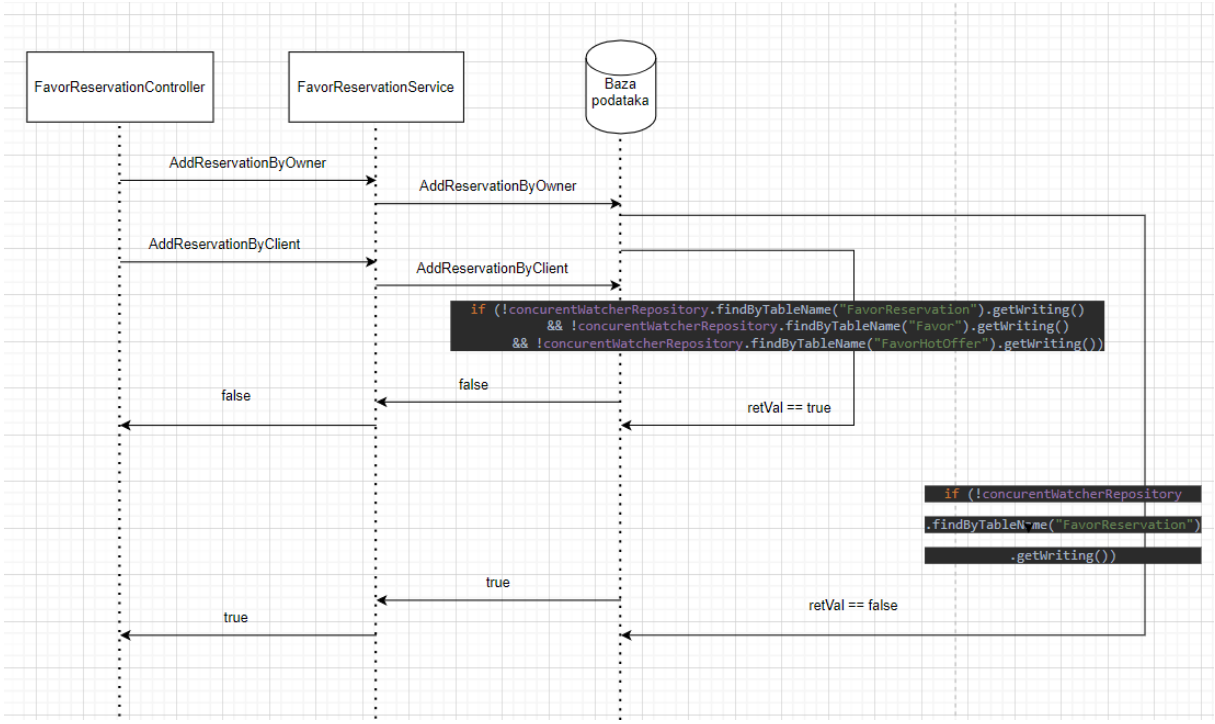
# 4.4 Konkurentni pristup resursima u bazi

1. Instruktor i korisnik ne mogu u isto vreme da naprave rezervaciju

-Opis situacije: Instruktor može u sklopu dogovora sa klijentom koji trenutno ima aktivnu rezervaciju da napravi dogovor i da kreira još jednu rezervaciju instruktorove usluge za datog klijenta. Sa druge strane, u isto vreme je moguće da drugi klijent, korisnik sistema, kreira rezervaciju za pomenutu instruktorovu uslugu. Time dolazi do preklapanja rezervacija i do pokušaja upisa dve rezervacije istovremeno u bazu.

Tok zahteva:




Rešenje konfliktne situacije: Pomenuta konfliktna situacija rešena je uz pomoć repozitorijuma concurentWatcherRepository, uz pomoć kojeg je implementirano privremeno zaključavanje resursa. Ukoliko je resurs slobodan, prvi korisnik koji mu pristupi(instruktor ili klijent) će zauzeti taj resurs, zaključati ga i na taj način onemogućiti ostale korisnike da pristupaju tom resursu sve dok je on zauzet.

Prikaz u kodu(za instruktora):

```
@Transactional(readOnly = false)
public Boolean addReservationByOwner(FavorReservation favorReservation){
    if (!concurentWatcherRepository.findByTableName("FavorReservation").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("FavorReservation");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

```
        favorReservationRepository.save(favorReservation);
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        sendNotificationForReservation(favorReservation);
        return true;
    }
    else {
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        return false;
```

Kao što možemo primetiti, metode unutar klase FavorReservationServiceImpl su anotirana sa @Transactional. Na prvoj slici je prikazana prvo provera da li je resurs slobodan, ukoliko jeste resurs se zaključava sve dok se kreira nova rezervacija. Nakon uspešnog/neuspešnog kreiranja rezervacije, pomenuti resurs se otključava nakon čega je moguć ponovni pristup datom resursu.

Prikaz u kodu(za klijenta):

```
@Transactional(readOnly = false)
public Boolean addReservationByClient(FavorReservation favorReservation) {
    if (!concurentWatcherRepository.findByTableName("FavorReservation").getWriting()
            && !concurentWatcherRepository.findByTableName("Favor").getWriting()
            && !concurentWatcherRepository.findByTableName("FavorHotOffer").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("FavorReservation");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

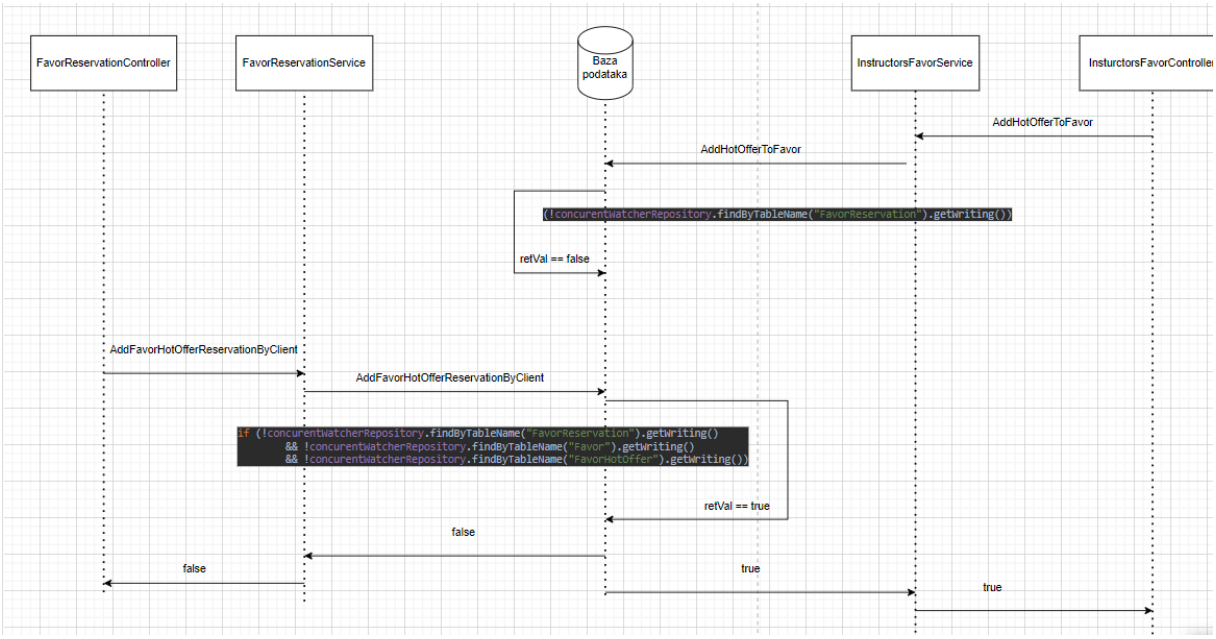
```
        favorReservationRepository.save(favorReservation);
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        sendNotificationForClientReservation(favorReservation);
        return true;
    } else {
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        return false;
    }
```

Što se tiče klijenta, kod njega se vrši više provera koje podrazumevaju da li se instruktorova usluga koju želi da rezerviše trenutno modifikuje ili briše (!concurentWatcherRepository.findByTableId("Favor").getWriting()), kao i da li se vrši upis nove brze akcije vezane za tu uslugu(!concurentWatcherRepository.findByTableId("FavorHotOffer").getWriting()), dok se dalji postupak zaključavanja i oslobadjanja resursa vrši analogno kreiranju nove rezervacije od strane instruktora.

2. Instruktor ne može da napravi brzu akciju u isto vreme kad i klijent vrši rezervaciju postojećeg entiteta

-Opis situacije: Instruktor ima mogućnost da u okviru svake usluge kreira brze akcije koje ostali korisnici mogu rezervisati jednim klikom. Ukoliko instruktor definiše novu akciju u isto vreme kada i drugi klijent vrši rezervaciju dolazi do preplitanja prilikom upisa u bazu.

Tok zahteva:




Rešenje konfliktne situacije: Pomenuta konfliktna situacija rešena je na isti način kao i prethodno opisana konfliktna situacija, samo se vrši zaključavanje drugih resursa. Sve dok je resurs zaključan, nijedan drugi korisnik ne može da mu pristupi.

Prikaz u kodu(za instruktora):

```
@Transactional(readOnly = false)
public Boolean addHotOfferToFavor(InstructorsFavor favor){
    if (!concurentWatcherRepository.findByTableName("FavorReservation").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("FavorHotOffer");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

```
        if(free){
            InstructorsFavor updatedFavor = setValid(favor);
            instructorsFavorRepository.save(updatedFavor);
            sendNotificationForNewHotOffer(updatedFavor);
        }

        cw.setWriting(false);
        concurentWatcherRepository.save(cw);

        return free;
    }
    else {
        return false;
```

Na slikama je prikazano zauzimanje i oslobadanje resursa od strane instruktora, ukoliko je resurs slobodan, kao i dodavanje nove brze akcije usluzi za koju je definisana.

Prikaz u kodu(za klijenta):

```
@Transactional(readOnly = false)
public Boolean addFavorHotOfferReservationByClient(FavorReservation favorReservation) {
    if (!concurentWatcherRepository.findByTableName("FavorReservation").getWriting()
            && !concurentWatcherRepository.findByTableName("Favor").getWriting()
            && !concurentWatcherRepository.findByTableName("FavorHotOffer").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("FavorReservation");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

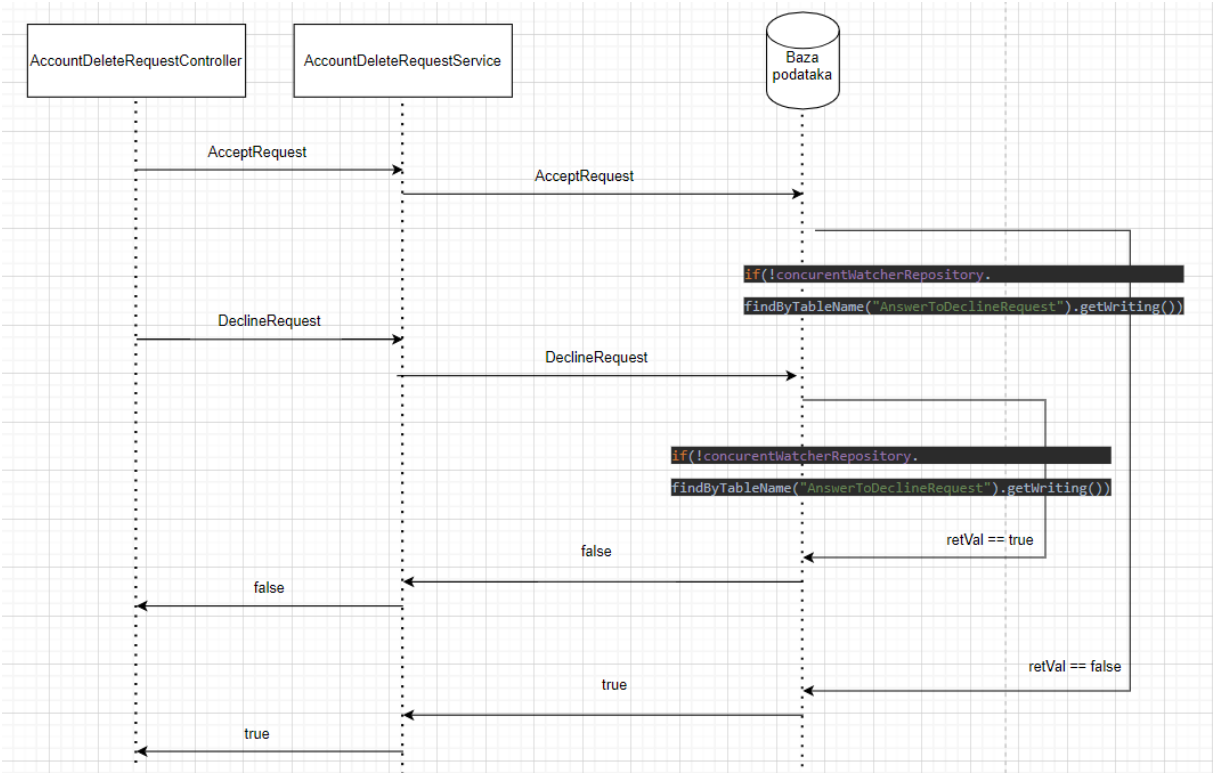
```
        favorReservationRepository.save(favorReservation);
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        sendNotificationForClientReservation(favorReservation);
        return true;
    }
    else {
        cw.setWriting(false);
        concurentWatcherRepository.save(cw);
        return false;
```

Takođe se, sa klijentske strane, vrše provere da li se trenutno modifikuje ili briše usluga koju želi da rezerviše, kao i da li se dodaju novi termini za brze rezervacije u uslugu koju klijent želi da rezerviše. Ukoliko je klijentu omogućeno da kreira brzu rezervaciju, on zaključava resurs, kreira rezervaciju, oslobađa resurs i u zavisnosti od uspešnosti kreiranja rezervacije vraća se true/false.

3. Samo jedan administrator može da odgovori na zahtev za brisanje naloga

-Opis situacije: Svaki korisnik sistema može da podnese zahtev za brisanje naloga uz obrazloženje. Sve pristigle zahteve za brisanje mogu da vide svi administratori. U slučaju kada bi dva administratora odgovorila na zahtev korisnika za brisanje naloga, došlo bi do preplitanja. Uzmimo za primer da su u isto vreme dva različita administratora obradila zahtev za brisanje naloga, jedan potvrdno a drugi odrično. Posto se korisniku na mejl šalje informacija o tome da li mu je zahtev za brisanje naloga prihvaćen ili nije, korisniku bi bila istovremeno poslata dva mejla. U jednom bi pisalo da je zahtev prihvaćen, a u drugom da nije.

Tok zahteva:




Rešenje konfliktne situacije: Pomenuta konfliktna situacija rešena je tako što je jedan resurs zaključavan svaki put kada bi neki administrator odobrio ili odbio zahtev za brisanje naloga nekog korisnika. U tom slučaju, drugi administrator ne bi mogao da utiče na to da li će zahtev biti prihvaćen ili odbijen, a samim tim ne bi došlo do kolizije i višestrukog slanja mejla korisniku koji je poslao zahtev za brisanje naloga.

Prikaz u kodu:

```
@Transactional(readOnly = false)
public Boolean acceptRequest(AccountDeleteRequest accountDeleteRequest) {
    if(!concurentWatcherRepository.findByTableName("AnswerToDeclineRequest").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("AnswerToDeclineRequest");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

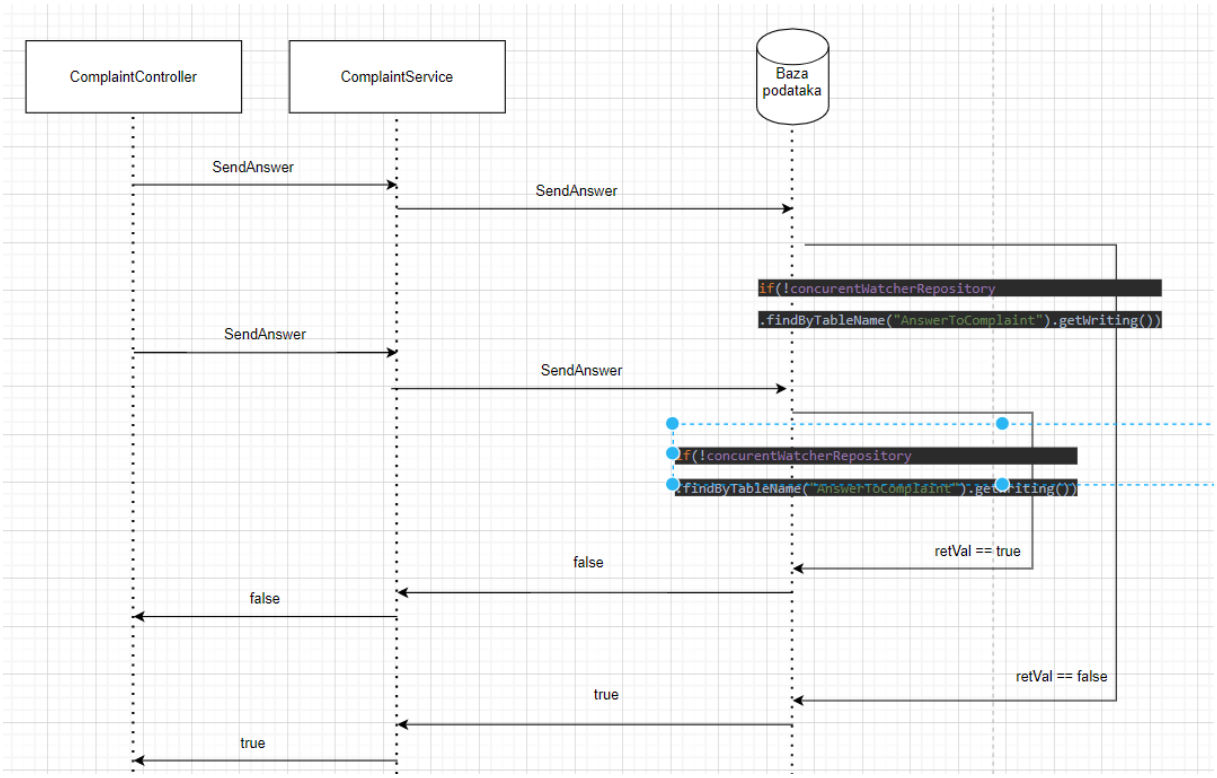
```
@Transactional(readOnly = false)
public Boolean declineRequest(AccountDeleteRequest accountDeleteRequest) {
    if(!concurentWatcherRepository.findByTableName("AnswerToDeclineRequest").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("AnswerToDeclineRequest");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

Kao što vidimo na slikama, resurs AnswerToDeclineRequest se zaključava svaki put kada jedan administrator prihavti/odbije zahtev, nakon čega se šalje mejl korisniku kako bi se on informisao da li je njegov zahtev prihvaćen ili odbijen. Takođe, metode su anotirane sa @Transactional.

4. Administrator može da odgovora na žalbe

-Opis situacije: Korisnici mogu da upute žalbe administratorima sistema na neki entitet ili nekog vlasnika entiteta. Svi administratori imaju uvid u žalbe, kao i mogućnost da na iste odgovore. U situaciji kada bi dva administratora istovremeno odgovorila na žalbu, korisniku koji je poslao žalbu bi se istovremeno poslala dva odgovora, ali različita. Opet možemo da uzmemo za primer da jedan administrator odgovori korisniku da je žalba uvažena, dok drugi administrator može da odgovori da je žalba odbijena iz nekog razloga, čime bi se stvorila kolizija sa korisnikove strane.

Tok zahteva:




Rešenje konfliktne situacije: Pomenuta konfliktna situacija rešena je tako što je jedan resurs zaključavan svaki put kada bi neki administrator odlučio da pošalje odgovor na žalbu korisnika. U tom slučaju, drugi administrator ne bi mogao da šalje odgovor na istu žalbu korisniku jer bi resurs bio zaključan.

Prikaz u kodu:

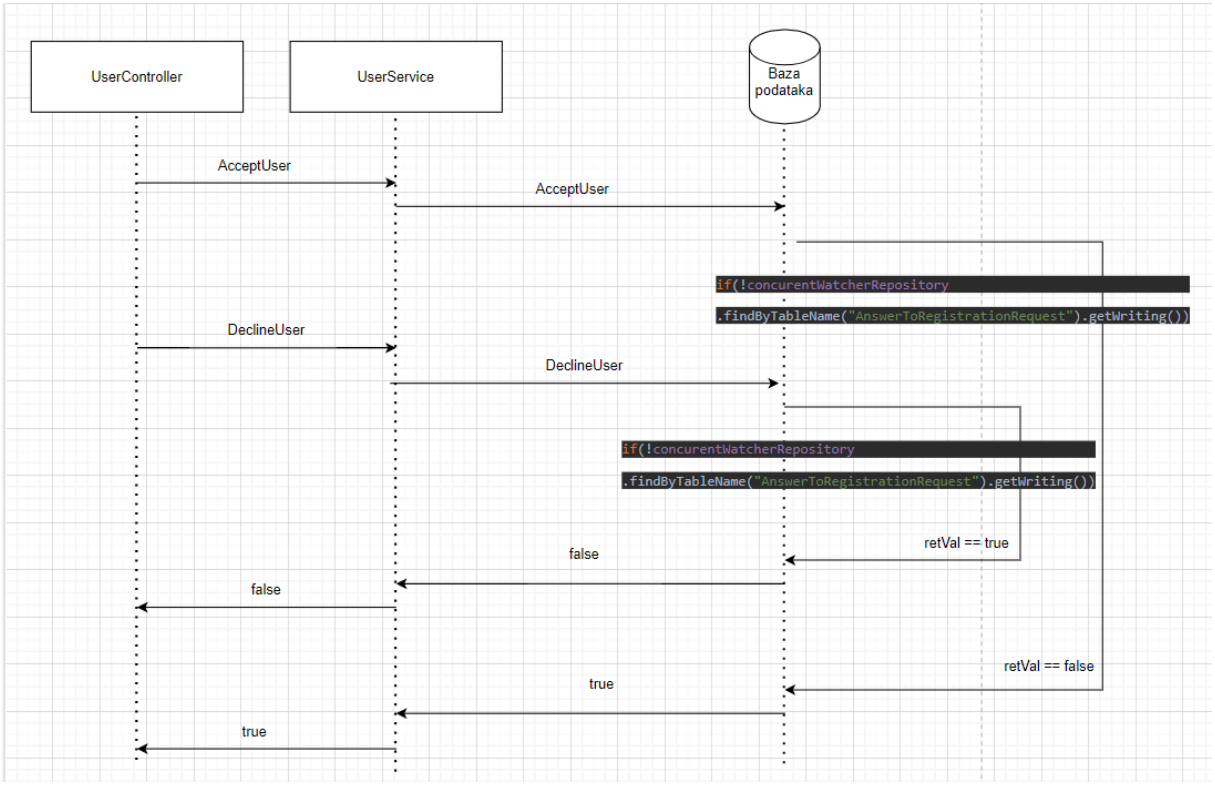
```
@Transactional(readOnly = false)
public Boolean sendAnswer(Complaint complaint) {
    if(!concurentWatcherRepository.findByTableName("AnswerToComplaint").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("AnswerToComplaint");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

Kao što vidimo na slici, konfliktna situacija je rešena zaključavanjem resursa AnswerToComplaint čime je omogućeno da samo jedan administrator odgovori na određenu žalbu korisnika.

5. Administrator može da prihvati ili odbije zahtev za registraciju korisnika

Opis situacije: Svaki neregistrovani klijent ima mogućnost da se registruje na sajt putem forme za registraciju. Unutar te forme bira da li želi da se registruje kao klijent, vlasnik vikendice, vlasnik broda ili instruktor. Svi zahtevi za registrovanje su vidljivi svim administratorima sistema, na koje oni mogu da utiču. Ukoliko se zahtev prihvati, šalje se email o odobravanju zahteva za registraciju, dok sa druge strane ukoliko se odbije zahtev za registraciju, šalje se email sa obrazloženjem zašto je zahtev odbijen. Takođe možemo uzeti za primer da dva administratora različito odgovore na zahtev za registraciju, jedan potvrdno a drugi odrično. Korisniku bi stigla dva mejla, u jednom bi mu aktivacija naloga bila omogućena, dok bi u drugom dobio obrazloženje zašto je njegov zahtev odbijen.

Tok zahteva:




Rešenje konfliktne situacije: Pomenuta konfliktna situacija rešena je tako što je jedan resurs zaključavan svaki put kada bi neki administrator odlučio da prihvati ili odbije zahtev za registraciju. U slučaju da odobri zahtev za registraciju, korisniku bi se poslao mejl sa potvrdnim odgovorom na njegov zahtev za registraciju, dok bi u slučaju odričnog odgovora dobio mejl sa razlogom zašto je njegov zahtev odbijen. Tek nakon što bi administrator obradio zahtev, resurs bi se otključao i administratori bi opet mogli da obrađuju preostale zahteve za registraciju.

Prikaz u kodu:

```
@Transactional(readOnly = false)
public Boolean acceptUser(long id){
    if(!concurentWatcherRepository.findByTableName("AnswerToRegistrationRequest").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("AnswerToRegistrationRequest");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

```
@Transactional(readOnly = false)
public Boolean declineUser(String text) {
    if(!concurentWatcherRepository.findByTableName("AnswerToRegistrationRequest").getWriting()) {
        ConcurentWatcher cw = concurentWatcherRepository.findByTableName("AnswerToRegistrationRequest");
        cw.setWriting(true);
        concurentWatcherRepository.save(cw);
```

Kao što vidimo na slici, resurs koji se zaključava jeste AnswerToRegistrationRequest, čime je onemogućen konkurentni pristup od strane više administratora.