# Konkurentni pristup podacima u bazi

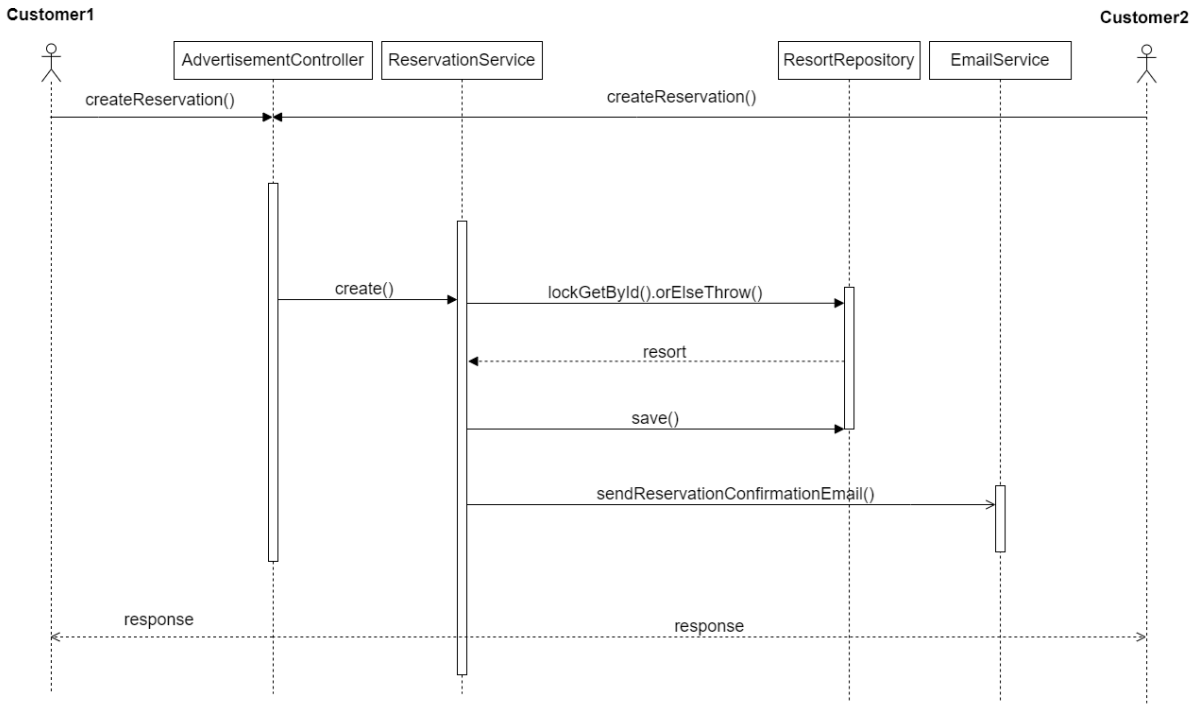
Student 1
Milan Sekulić SW54/2019

## 1. Dva klijenta za jedan entitet istovremeno prave rezervacije koje se poklapaju

***Problem:*** Klijent ima sposobnost da napravi rezervaciju za određeni entitet u sistemu odabirom datuma početka rezervacije, datuma završetka rezervacije i nekih drugih (za ovaj primjer manje bitnih) opcija. Naravno, klijentu su za izbor ponuđeni samo datumi koji su raspoloživi, što podrazumijeva da je entitet u tom periodu raspoloživ i da ne postoji druga rezervacija koja se odigrava tog datuma.

Problem nastaje kada dva klijenta istovremeno pokušaju da za isti entitet naprave rezervacije čiji se datumi dešavanja poklapaju u nekoj mjeri.

***Tok zahtjeva:*** Na slici 1 je prikazan dijagram sekvence za kreiranje rezervacije za vikendicu. Analogno tome se vrši kreiranje rezervacija za brodove i avanture, samo što se pozivaju metode iz repozitorijuma koji odgovaraju njima.




Slika 1 - dijagram sekvence za kreiranje rezervacije za vikendicu

***Rješenje:*** Ovaj problem je riješen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa POST metodom prilikom rezervacije od strane korisnika je */ads/{id}/reservations* kontrolera AdvertisementController.

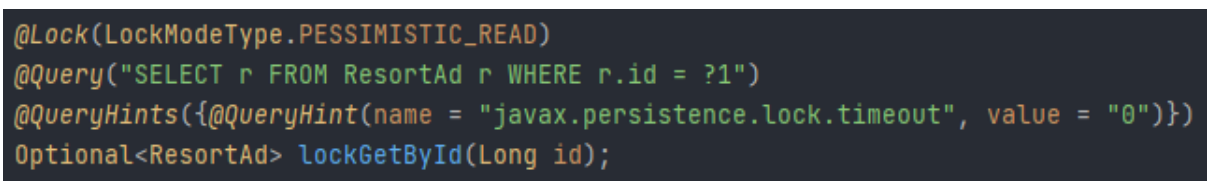

Slika 2 - Metoda kontrolera za prvi konflikni slučaj

Metoda *createReservation* ovog kontrolera, prikazana na slici 2, poziva metodu *create* servisa ReservationService.

Metoda *create* (Slika 3) ima anotaciju *@Transactional*. Njen posao je da zaključi konkretan tip oglasa, na osnovu čega poziva metodu *lockGetById* odgovarajućeg repozitorijuma. Repozitorijumi za vikendice, brodove i avanture sadrže ovu metodu, ali u konkretnom primjeru je prikazan primjer ove metode za vikendice. U slučaju da dva klijenta istovremeno pozivaju metodu za istu vikendicu, metoda *lockGetById* će baciti ***PessimisticLockingFailureException*** jednom od njih. Try/Catch će uhvatiti izuzetak i naknadno baciti *ReservationConflictException* sa adekvatnom porukom koja će se prikazati klijentu na frontu.


Slika 3 - Metoda servisa za prvi konkretan slučaj

Slika 4 prikazuje metodu *lockGetById* za *ResortAdRepository*. Koristi pesimističko zaključavanje za čitanje koje nam baca izuzetak u slučaju da su istovremeno dva korisnika pokušala da pristupe istoj vikendici.



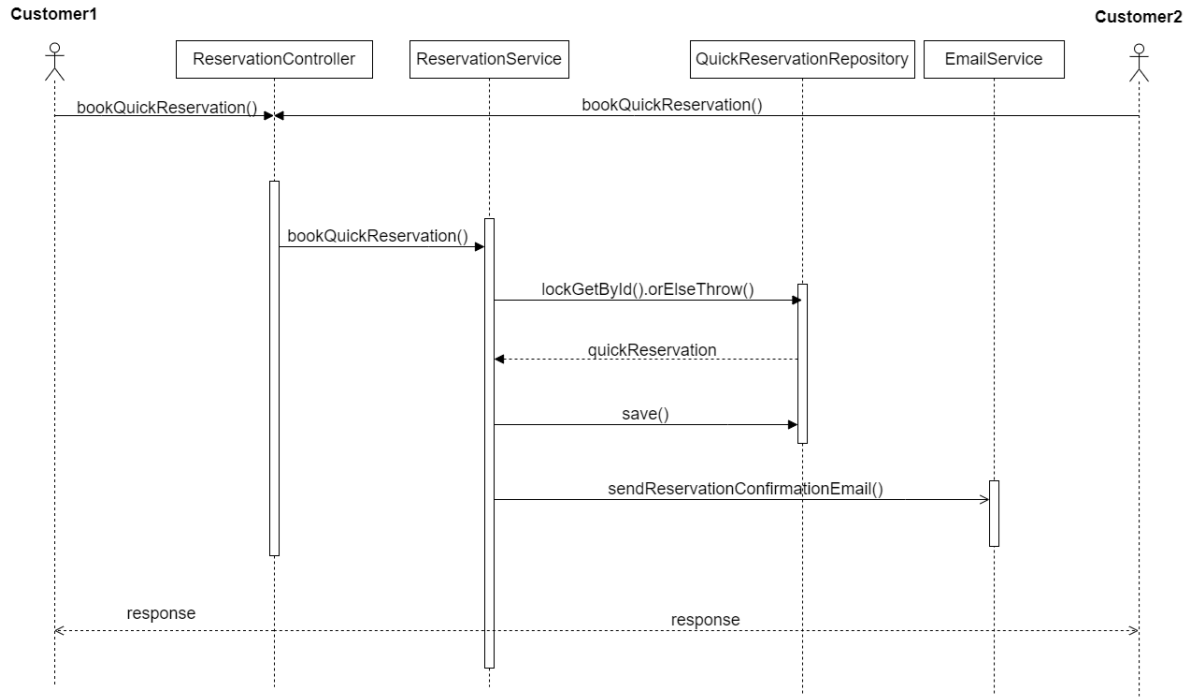

Slika 4 - Metoda repozitorijuma za prvi konfliktni slučaj

## 2. Dva klijenta istovremeno rezervišu istu brzu rezervaciju

***Problem:*** Klijent ima sposobnost da rezerviše brzu rezervaciju (predefinisanu rezervaciju na popustu) za određeni entitet u sistemu odabirom željene postojeće brze rezervacije i klikom na dugme 'Book'.

Problem nastaje kada dva klijenta istovremeno pokušaju da za isti entitet rezervišu brzu rezervaciju.

***Tok zahtjeva:*** Na slici 5 je prikazan dijagram sekvence za rezervisanje brze rezervacije.




Slika 5 - Dijagram sekvence za rezervisanje brze rezervacije

***Rješenje:*** Ovaj problem je riješen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa POST metodom prilikom rezervacije od strane korisnika je */reservations/{id}/book-quick-reservation* kontrolera ReservationController.

```
@PostMapping(value = "/{id}/book-quick-reservation")
@PreAuthorize("hasRole('CUSTOMER')")
public ResponseOK bookQuickReservation(@PathVariable Long id, Authentication auth) {
    reservationService.bookQuickReservation(id, auth);
    return new ResponseOK("Quick reservation booked.");
}
```

Slika 6 - Metoda kontrolera za drugi konfliktni slučaj

Metoda *bookQuickReservation* ovog kontrolera, prikazana na slici 6, poziva metodu *bookQuickReservation* servisa ReservationService.

Metoda *bookQuickReservation* (Slika 7) ima anotaciju *@Transactional*. Ona poziva metodu *lockGetById* repozitorijuma za brze rezervacije *QuickReservationRepository*. U slučaju da dva klijenta istovremeno pozivaju metodu za istu brzu rezervaciju, metoda *lockGetById* će baciti ***PessimisticLockingFailureException*** jednom od njih. Try/Catch će uhvatiti izuzetak i naknadno baciti *ReservationConflictException* sa adekvatnom porukom koja će se prikazati klijentu na frontu.

```
@Transactional
public void bookQuickReservation(Long id, Authentication auth) {
    Customer customer = (Customer) auth.getPrincipal();
    QuickReservation quickReservation;
    try {
        quickReservation = quickReservationRepository.lockGetById(id).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new ReservationConflictException(
                "Another user is currently attempting to book this quick reservation. " +
                "Please try again in a few seconds.");
    }
```

Slika 7 - Metoda servisa za drugi konfliktni slučaj

Slika 8 prikazuje metodu *lockGetById* za *QuickReservationRepository*. Koristi pesimističko zaključavanje za pisanje koje nam baca izuzetak u slučaju da su istovremeno dva korisnika pokušala da pristupe istoj brzor rezervaciji.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT qr FROM QuickReservation qr WHERE qr.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<QuickReservation> lockGetById(Long id);
```

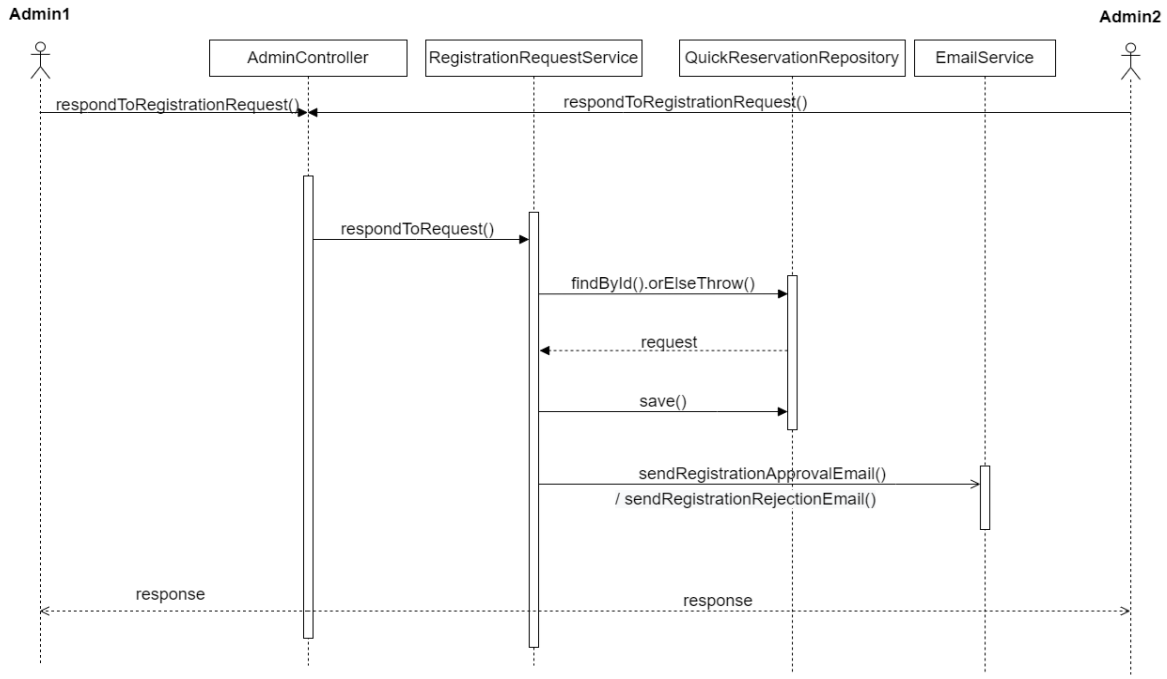
Slika 8 - Metoda repozitorijuma za drugi konfliktni slučaj

# 3. Dva administratora istovremeno odgovaraju na isti zahtjev za registraciju

***Problem:*** Pri registraciji oglašavača, zahtjev za registraciju se čuva i administratori sistema imaju opciju da ga prihvate ili odbiju uz odgovarajuće obrazloženje.

Problem nastaje kada dva administratora istovremeno pokušaju da prihvate/odbiju isti zahtjev za registraciju. Ovdje dolazi do još jednog očiglednog problema, a to je da se može desiti da iz nekog razloga jedan administrator prihvati zahtjev, a drugi ga odbije. Nakon toga bi se poslala dva različita mejla korisniku, što bi dovelo do dodatne zabune.

***Tok zahtjeva:*** Na slici 9 je prikazan dijagram sekvence za odgovaranje na jedan isti zahtjev za rezervaciju. Dva admina istovremeno šalju odgovor za isti zahtjev koji potencijalno dovodi do različitih odgovora istom potencijalnom korisniku.




Slika 9 - Dijagram sekvence za odgovaranje na zahtjev za registraciju

***Rješenje:*** Ovaj problem je riješen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa PATCH metodom prilikom adminovog odgovora na zahtjev za registraciju je */admin/registration-requests/{id}* kontrolera AdminController.

```
@PatchMapping(value = "/registration-requests/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasRole('ADMIN')")
public ResponseOK respondToRegistrationRequest(@PathVariable Long id, @Valid @RequestBody RegistrationRequestResponseDTO dto) {
    registrationRequestService.respondToRequest(id, dto);
    return new ResponseOK("Request resolved.");
}
```

Slika 10 - Metoda kontrolera za treći konfliktni slučaj

Metoda *respondToRegistrationRequest* ovog kontrolera, prikazana na slici 10, poziva metodu *respondToRequest* servisa RegistrationRequestService.

Metoda *respondToRequest* (Slika 11) ima anotaciju *@Transactional*. Ona poziva metodu *lockGetById* repozitorijuma *RegistrationRequestRepository*. U slučaju da dva admina istovremeno pozivaju metodu za isti zahtjev za registraciju, metoda *lockGetById* će baciti ***PessimisticLockingFailureException*** jednom od njih. Try/Catch će uhvatiti izuzetak i naknadno baciti *RegistrationRequestResponseConflictException* sa adekvatnom porukom koja će se prikazati adminu na frontu.

```
@Transactional
public void respondToRequest(Long id, RegistrationRequestResponseDTO dto) {
    RegistrationRequest request;
    try {
        request = registrationRequestRepository.lockGetById(id).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new RegistrationRequestResponseConflictException(
                "Another admin is currently attempting to respond to this request. Try again in a few seconds.");
    }
```

Slika 11 - Metoda servisa za treći konfliktni slučaj

Slika 12 prikazuje metodu *lockGetById* za *RegistrationRequestRepository*. Koristi pesimističko zaključavanje za pisanje koje nam baca izuzetak u slučaju da su istovremeno dva admina pokušala da promijene isti zahtjev za registraciju.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT r FROM RegistrationRequest r WHERE r.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<RegistrationRequest> lockGetById(Long id);
```

Slika 12 - Metoda repozitorijuma za treći konfliktni slučaj