Konkurentnost - Student 2 Srđan Đurić – SW63-2019

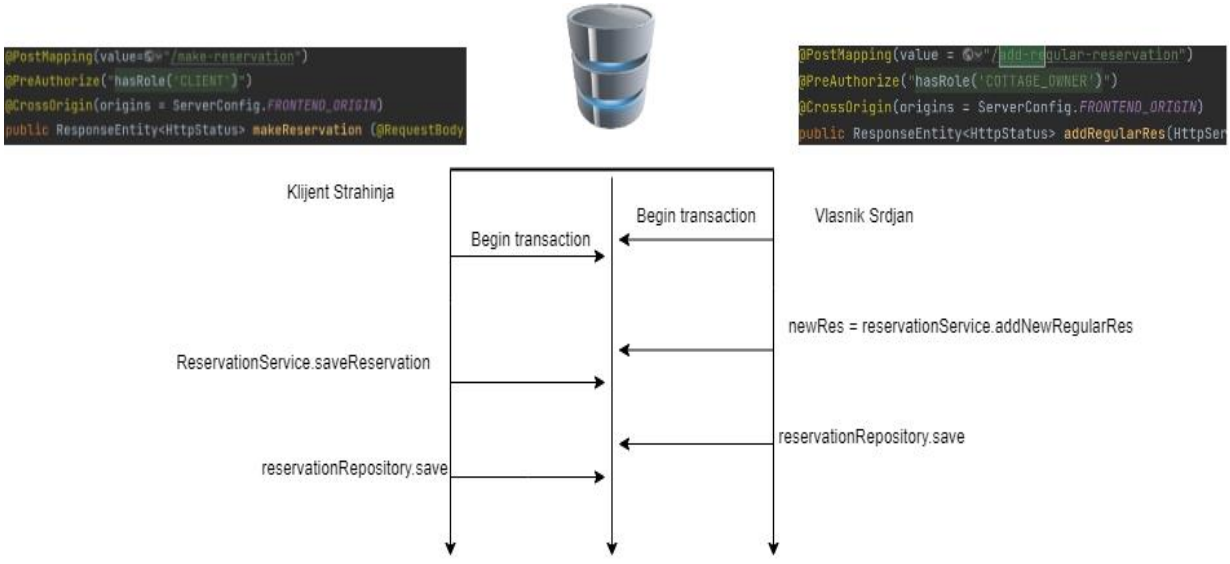
Rešavane konfliktne situacije:

1. Vlasnik vikendice/broda ne može da napravi rezervaciju u isto vreme kad i drugi klijent
2. Vlasnik vikendice/broda ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta
3. DODATNA SITUACIJA: Vlasnik vikendice/broda ne može da menja osnovne podatke entiteta(kao što su cena, opis, kapacitet..) u trenutku kada klijent rezerviše taj entitet

Rešenja:

1. **Vlasnik vikendice/broda ne može da napravi rezervaciju u isto vreme kad i drugi klijent**

   Scenario problema: Vlasnik nekog objekta za rentiranje i klijent u isto vreme pokušavaju da naprave rezervaciju za isti entitet. Obe vrste korisnika prvo izaberu datum za izabranu rezervaciju, međutim budući da rezervaciju vrše u isto vreme, doći će do konkuretne situacije. Vlasnik će napraviti novu rezervaciju za nekog klijenta i izmene sačuvati u bazi, ali je u istom tom trenutku i klijent napravio rezervaciju za isti period, samo nije svestan da je taj period zauzet jer ima staru verziju objekta svih rezervacija. Klijent će samim tim takođe uspeti da napravi rezervaciju za isti period i izmene sačuvati u bazi čime će biti narušena konzistentnost.



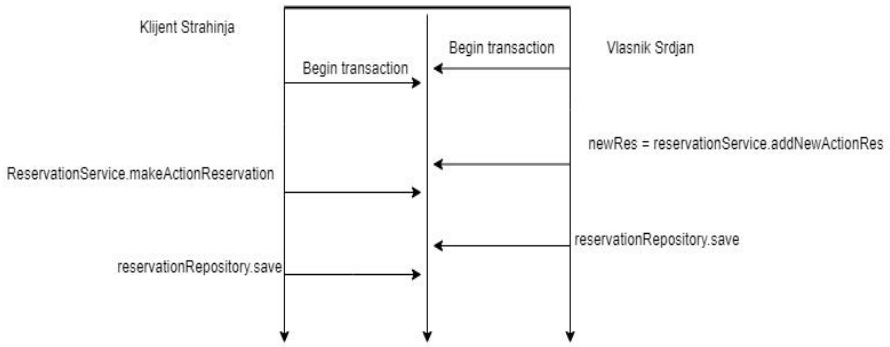
   Rešenje problema: *Objedinjeno sa rešenjem problema tačke 2.*

4. **Vlasnik vikendice/broda ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta**

   Scenario problema: Vlasnik nekog objekta za rentiranje i klijent u isto vreme pokušavaju da naprave rezervaciju za isti entitet. Obe vrste korisnika prvo izaberu datum za izabranu rezervaciju, međutim budući da rezervaciju vrše u isto vreme, doći će do konkuretne situacije. Vlasnik će napraviti novu akcijsku rezervaciju i izmene sačuvati u bazi, ali je u istom tom trenutku I klijent napravio rezervaciju za isti period, samo nije svestan da je taj period zauzet jer ima staru verziju objekta svih rezervacija. Klijent će samim tim takođe uspeti da napravi rezervaciju za isti period i izmene sačuvati u bazi čime će biti narušena konzistentnost.

```
@PostMapping(value = "/add-action-reservation")
@PreAuthorize("hasRole('COTTAGE_OWNER')")
@CrossOrigin(origins = ServerConfig.FRONTEND_ORIGIN)
public ResponseEntity<HttpStatus> addNewActionRes(HttpServle
```

```
@PutMapping(value="/make-action-reservation/{resId}")
@PreAuthorize("hasRole('CLIENT')")
@CrossOrigin(origins = ServerConfig.FRONTEND_ORIGIN)
public ResponseEntity<HttpStatus> makeActionReservation
```



Rešenje problema: Za rešenje problema korišćeno je optimistično zaključavanje. Dodata je kolona version u RentalService i boolean isChanged. Prilikom svakog dodavanja nove rezervacije vrši se prvo menjanje boolean vrednosti is changed i njegovo čuvanje u bazu. Svakim pozivom save metode vršiće se inkrementiranje version atributa. Ukoliko dva korisnika pokušaju da izvrše rezervaciju u isto vreme to neće biti moguće jer jedan neće imati odgovarajuću verziju version-a te neće biti izvršeno čuvanje druge transakcije. Tako da će se pri pokušaju ažuriranja rentala gde nije najnovija verzija desiti *ObjectOptimisticLockingFailureException*.

```
@Version
@Column(columnDefinition = "integer DEFAULT 0", nullable = false)
private Long version;

@Column(name = "is_changed",columnDefinition = "boolean DEFAULT false", nullable = false)
private boolean isChanged;
```

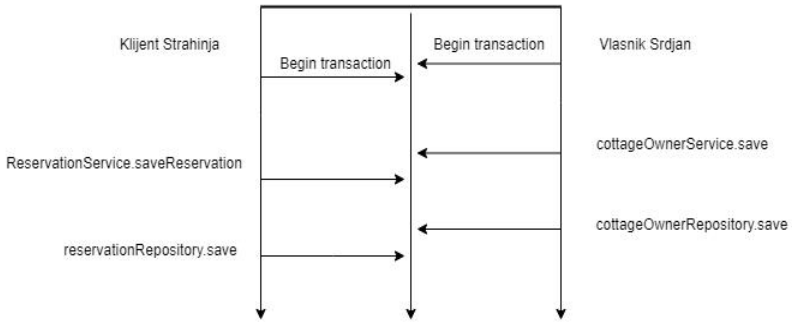
Takođe je dodata **@Transactional** anotacija nad onim metodama servisa koje rade sa kreiranjem rezervacija.

2. DODATNA SITUACIJA: Vlasnik vikendice/broda ne može da menja osnovne podatke entiteta(kao što su cena, opis, kapacitet..) u trenutku kada klijent rezerviše taj entitet

Scenario problema: Klijent želi da izvrši kreiranje rezervacije za određeni entitet. U istom trenutku vlasnik ažurira podatke za njegov objekat za rentiranje. Ukoliko vlasnik promeni cenu, opis, kapacitet ili neki ostali podatak u istom trenutku kada i klijent pravi rezervaciju, klijent će biti u prilici da kreira rezervaciju po staroj ceni, pri čemu takođe neće imati saznanje o potencijalnim izmenama atributa entiteta.

```
@PostMapping(value="/make-reservation")
@PreAuthorize("hasRole('CLIENT')")
@CrossOrigin(origins = ServerConfig.FRONTEND_ORIGIN)
public ResponseEntity<HttpStatus> makeReservation (@RequestBody
```

```
@PreAuthorize("hasRole('COTTAGE_OWNER')")
@CrossOrigin(origins = ServerConfig.FRONTEND_ORIGIN)
@PutMapping(consumes="application/json", value="/change-cottage-data")
public ResponseEntity<HttpStatus> updateCottageData(HttpServletRequest req
```



Rešenje problema: Za rešenje problema korišćeno je optimistično zaključavanje. Dodata je kolona version u RentalService i boolean isChanged. Prilikom svakog dodavanja nove rezervacije vrši se prvo menjanje boolean vrednosti is changed i njegovo čuvanje u bazu. Svakim pozivom save metode vršiće se inkrementiranje version atributa. Isto tako se priliko izmene podataka za svaki rental takođe vrši inkrementiraje atributa version. Tako da će se pri pokušaju ažuriranja rentala gde nije najnovija verzija desiti *ObjectOptimisticLockingFailureException*. Dodate su anotacije @Transactional nad metodama za izmenu rentala i kreiranje rezervacija.

```
@Version
@Column(columnDefinition = "integer DEFAULT 0", nullable = false)
private Long version;

@Column(name = "is_changed",columnDefinition = "boolean DEFAULT false", nullable = false)
private boolean isChanged;
```