ISA-MRS KONKURENTI PRISTUP RESURSIMA

Spisak konfliktnih situacija koje su rešavane:

- Više istovremenih klijenata ne mogu da naprave rezervaciju u isto vreme.

-Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vreme.

-Na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator.

-Na jednu žalbu moze da odgovri samo jedan administrator.

-Samo jedan administrator može da obriše vikendicu u sistemu.

Prikaz kontroler metode koja se poziva prilikom brisanja entiteta:

```
@GetMapping(value = "/delete/{id}",produces = "application/json")
public CompletableFuture<ResponseEntity<Boolean>> delete(@PathVariable Long id) {
    return  cs.deleteCottage(id).thenApplyAsync(ResponseEntity::ok);
}
```

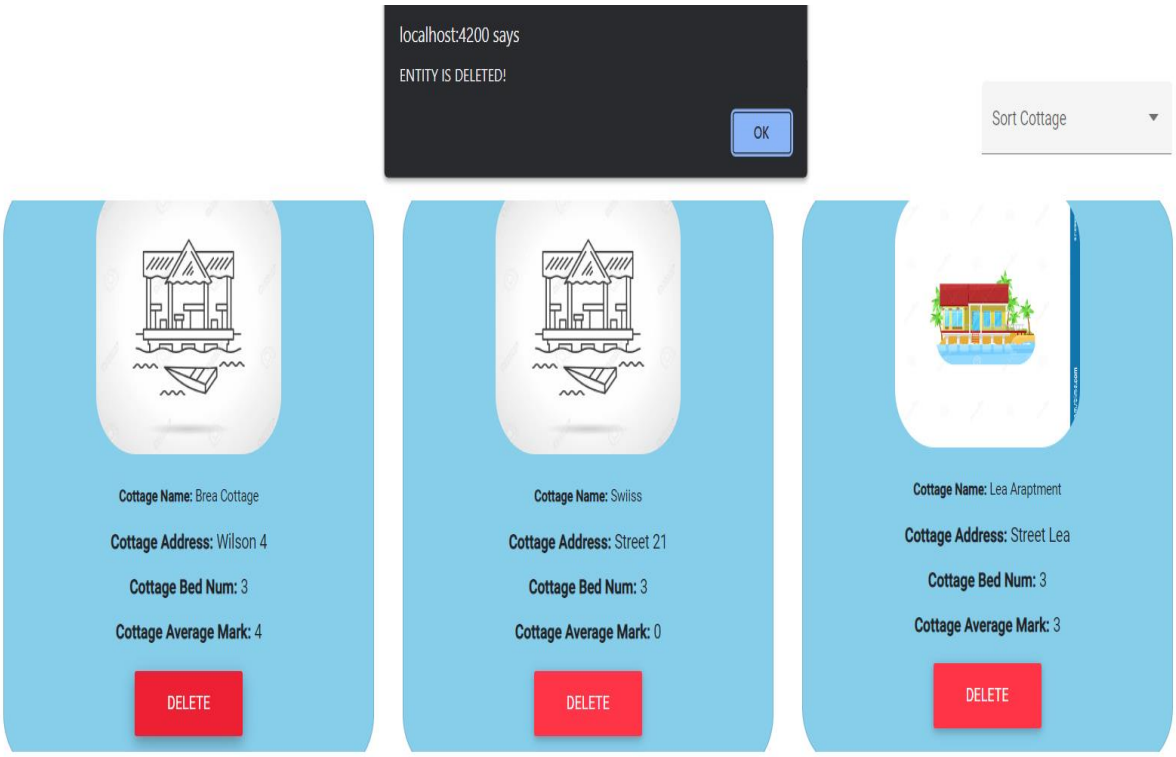
Slika 1.

Prikaz servisne metode koja se poziva prilikom birsanja entiteta:

```
@Async
public synchronized CompletableFuture<Boolean> deleteCottage(Long id) {

    log.info("DELETE COTTAGE WITH ID: " + id);

    Optional<Cottage> c = cr.findById(id);
    try {
        c.get().setDelete(true);
    }
    catch (Exception e){
        return CompletableFuture.completedFuture(false);
    }

    cr.save(c.get());

    return CompletableFuture.completedFuture(true);

}
```

Slika 2.

Prikaz obaveštenja adminu koji je pokušao da obriše prethodno obrisani entitet:



Slika 3.

• Admin 1	• Admin2
• `GET /home/cottage - Dobavlja sve vikendice.`	• `GET /home/cottage - Dobavlja sve vikendice.`
• `GET /home/cottage/delete/{id} - Briše odgovarajuću vikendicu(slika 2).`	• `GET /home/cottage/delete/{id} - Briše odgovarajuću vikendicu(slika 2).`
• Brisanje je uspešno I page se reload.	• Dobija poruku da je vikendica obrisana I nakon toga page reload gde vikendica više ne postoji u sistemu(Slika 3).

Tabela 1.

Grafički prikaz tabele 1. na Slici 4. gde Admin1 I Admin2 pokušavaju da obrišu isti entitet u isto vreme.

Slika 4.

Svih 5 konfliktnih situacija koje su rešavane u projektu su implementirane tehnikom optimističnog zaključavanja ali nije korišćen @Version atribut već Boolean atributi koji su već postojali u projektu a efekat koji se postiže je identičan. Prilikom svakog pristupa od strane korisnika on se po prijemu reqeust-a na Backendu odvaja u posebni Thread(U aplikaciji je kreiran ThreadPool čiji size je 6 jer toliko ima jezgara laptop na kome sam radio aplikaciju). Cilj je da odgovarajuće metode Servis klase koje treba da reše konfliknte situacije označim sa synchronized ključnom reči I na taj način izvršavanje metode je sekvencijalno. Thread koji prvi počne sa izvršavanjem metode izvršava čitavo telo.Tek nakon toga Drugi Thread

počinje izvršavanje. Na početku svake konfliktne metode postoji uslov koji treba da se zadovolji I to zadovoljava samo prvi Thread I metoda vraća “true” vrednost dok ostali ne zadovoljavaju taj uslov I metoda vraca “false” vrednost. Ukoliko je vracena vrednost “false” na frontu se ispisuje poruka da nije moguće izvršiti akciju nad odgovarajućim entitetom.

Nešto efikasniji način je korišćenjem lock objekta:

```
private final Object lock = new Object();
```

Na ovaj način bi u sinhronom bloku mogli da zaključamo samo deo metode kako Drugi Thread ne bi morao da čeka na početku metode.