Konflikta situacija: Istovremeno rezervisanje entiteta na akciji

Opis konflikte situacije:

Na stranici entiteta postoji link ka listi aktuelnih akcija koje klijenti mogu da rezervisu samo jednim klikom. Pogotovo u slucaju veceg broja klijenata moguc je scenario gdje 2 klijenta (ili vise) klikcu na istu akciju istovremo. Osim toga, za vrijeme dok jedan klijent razgleda, stanje akcija moze da se mijenja od strane ostalih. Drugi klijent moze da rezervise akciju, a moze i da tu istu akciju otkaze.

Rješenje konfliktne situacije:

Klasa DiscountReservation koja predstavlja akciju ima atribut ReservationStatus kojim se prati stanje akcije. Ono sto je bitno jeste da stanje akcije u trenutku kada klijent rezervise buse ACTIVE. To je ono sto se provjerava prilikom rezervisanja akcije i vraca feedback klijentu. Time je rijesen problem vise promjena do kojih moze doci dok je klijentu samo otvorena stranica sa rasplozivim akcijama. Problem istovremenog rezervisanje je rijesen upotrebom Version -a. Ako klijent pristupa verziji DiscountReservation objekta koja nije nulta, desice se `ObjectOptimisticLockingFailureException,` koji ce se uhvatiti I vratit odgovor klijentu.

```
@Override
public DiscountReservation makeReservationOnDiscount(long resId) throws OfferNotAvailableException,ObjectOptimisticLockingFailureException {

    MansionDiscountReservation res = reservationRepo.findByIdAndStatus(resId,ReservationStatus.ACTIVE);
    if(res == null) throw new OfferNotAvailableException();
    else {
    res.setStatus(ReservationStatus.RESERVED);
    res.setUser(authenticationService.getLoggedUser());
    return reservationRepo.save(res);
    }
}
```

Klasa MansionDiscountReservationService, ista logika primjenjena i na rezervisanje brzih akcija ostalih entiteta.

```
@PreAuthorize("hasRole('ROLE_CLIENT')")
@RequestMapping(method = RequestMethod.GET, value = "/makeDiscountMansionReservation",produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<String> makeDiscountMansionReservation(@RequestParam Long id){

    try {
    mansionReservationService.makeReservationOnDiscount(id);
    return new ResponseEntity<>("Reservation successfull!", HttpStatus.OK);
    }
    catch(ObjectOptimisticLockingFailureException | OfferNotAvailableException e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.OK);
    }
}
```

Klasa DiscountReservationsController.

Konflikta situacija: Rezervisanje entiteta u istom ili preklapajucem periodu

Opis konflikte situacije:

Prilikom rezervacije entiteta klijent popunjava parametre forme po koijma se vrsi pretraga entiteta i vraca lista potencijalnih rezervacija. Klijent bira datum pocetka rezervacije i broj dana (u slucaju broda, i broj sati). Slicno situaciji sa brzim rezervacijama, od trenutka kada se klijentu prikazu rezultati pretrage, odnosno slobodni entiteti za trazeni datum i vrijeme, do trenutka kada pritisne dugme rezervisi stanje slobodnih perioda za entitet moze da se promijeni od strane drugih klijenata. Bilo bi lose kada bi klijent mogao da napravi rezervaciju za period, ili dio perioda koji vise nije Slobodan i nastale bi dvije razlicite rezervacije za isti entitet u isto ili preklapajuce vrijeme.

Rješenje konfliktne situacije:

Jos jedan poziv koji provjerava da li je period jos uvijek dostupan.

```
@Override
@Transactional(readOnly=false,propagation=Propagation.REQUIRED,isolation= Isolation.SERIALIZABLE)
public MansionReservation createReservation(ReservationDto res)  throws PeriodNoLongerAvailableException, ParseException {

    ReservationStartEndDateFormatter formatter = new ReservationStartEndDateFormatter(res);
    Date startDate = formatter.startDate;
    Date endDate = formatter.endDate;

    MansionAvailablePeriod period = availablePeriodsRepo.getPeriodOfInterest(startDate, endDate,res.getEntityId());

    if(period == null) {
        throw new PeriodNoLongerAvailableException();}

    else {

         MansionReservation newMansionReservation = new MansionReservation(authenticationService.getLoggedUser(),
                 startDate, endDate, res.getNumberOfGuests(),res.getPrice(), period.getMansion());


            if(!period.getStartDate().equals(startDate)) {
                MansionAvailablePeriod periodBefore = new MansionAvailablePeriod(period.getStartDate(),startDate,period.getMansion());
                availablePeriodsRepo.save(periodBefore);
            }
            if(!period.getEndDate().equals(endDate)) {
                MansionAvailablePeriod periodAfter = new MansionAvailablePeriod(endDate,period.getEndDate(),period.getMansion());
                availablePeriodsRepo.save(periodAfter);
            }


            newMansionReservation.setAdditionalServices(addAdditionalServices(res.getAdditionalServices()));
            newMansionReservation.setTotalPrice(res.getPrice() + accountAdditionalServices(newMansionReservation.getAdditionalServices(),res));

            availablePeriodsRepo.delete(period);
            return mansionReservationRepo.save(newMansionReservation);
    }

}
```

Konflikta situacija: Rezervisanje entiteta koji je u međuvremenu obrisan

Opis konflikte situacije:

Boat owner ili mansion owner imaju funkcionalnost brisanja svog entiteta. Uslov brisanje je da trenutno ne postoji nijedna rezervacija vezana za taj entitet, ali ne i da ne postoji definisan period dostupnosti. Ne postoji funkcionalost brisanja perioda dostupnosti. U prethodno opisanom postupku rezervacije, osim nove rezervacije moze se desiti i ovaj slucaj.Ovo brisanje u sistemu je logičko i klijent uspješno kreira nevažeću rezervaciju za obrisani entitet.

Rješenje konfliktne situacije:

U metodi kreiranje rezervacije je dodata i provjera da li je entitet logički obrisan i korisniku je u slučaju da jeste, klijent je o tome obavjesten. Kao i DiscountReservaiton, Reservation klasa ima svoj status koji se provjerava.

Ovaj slucaj je mogao biti rijesen i uvodjenjem transakcije kod brisanja entiteta i njegovih buducih perioda dostupnosti kod Studenta2.

```
@Override
@Transactional(readOnly=false,propagation=Propagation.REQUIRED,isolation= Isolation.SERIALIZABLE)
public MansionReservation createReservation(ReservationDto res)  throws PeriodNoLongerAvailableException, ParseException, EntityDeletedException {

    ReservationStartEndDateFormatter formatter = new ReservationStartEndDateFormatter(res);
    Date startDate = formatter.startDate;
    Date endDate = formatter.endDate;

    MansionAvailablePeriod period = availablePeriodsRepo.getPeriodOfInterest(startDate, endDate,res.getEntityId());
    Mansion mansion = mansionRepo.findByIdAndDeletedFalse(res.getEntityId());

    if(period == null) {
        throw new PeriodNoLongerAvailableException();
    }
    else if(mansion == null) {

        throw new EntityDeletedException();
    }
    else {

         MansionReservation newMansionReservation = new MansionReservation(authenticationService.getLoggedUser(),
                 startDate, endDate, res.getNumberOfGuests(),res.getPrice(), mansion);


            if(!period.getStartDate().equals(startDate)) {
                MansionAvailablePeriod periodBefore = new MansionAvailablePeriod(period.getStartDate(),startDate,period.getMansion());
                availablePeriodsRepo.save(periodBefore);
            }
            if(!period.getEndDate().equals(endDate)) {
                MansionAvailablePeriod periodAfter = new MansionAvailablePeriod(endDate,period.getEndDate(),period.getMansion());
                availablePeriodsRepo.save(periodAfter);
            }


            newMansionReservation.setAdditionalServices(addAdditionalServices(res.getAdditionalServices()));
            newMansionReservation.setTotalPrice(res.getPrice() + accountAdditionalServices(newMansionReservation.getAdditionalServices(),res));

            availablePeriodsRepo.delete(period);
            return mansionReservationRepo.save(newMansionReservation);
    }

}
```

```java
@PreAuthorize("hasRole('ROLE_CLIENT')")
@RequestMapping(method = RequestMethod.POST,value = "/reservations/createMansionReservation",
consumes = MediaType.APPLICATION_JSON_VALUE, produces = MediaType.APPLICATION_JSON_VALUE)
@CrossOrigin(origins = "*")
public ResponseEntity<String> createMansionReservation(@RequestBody ReservationDto res){

    try {
        MansionReservation newReservation = mansionResService.createReservation(res);
        try {
            mailService.sendMansionReservationConfirmationMail(newReservation);
        }catch (MessagingException e){
            return  new ResponseEntity<>("There is a problem with your mail!",HttpStatus.OK);
        }
        return new ResponseEntity<>("Reservation successfull!", HttpStatus.OK);
    } catch (ParseException e){
        return  new ResponseEntity<>("Check your date again!", HttpStatus.OK);
    } catch (PeriodNoLongerAvailableException e) {
        return  new ResponseEntity<>(e.getMessage(),HttpStatus.OK);
    } catch (Exception e) {
        return  new ResponseEntity<>(e.getMessage(),HttpStatus.OK);
    }
}
```

Detaljan opis i metode se nalaze na sequence dijagramima.