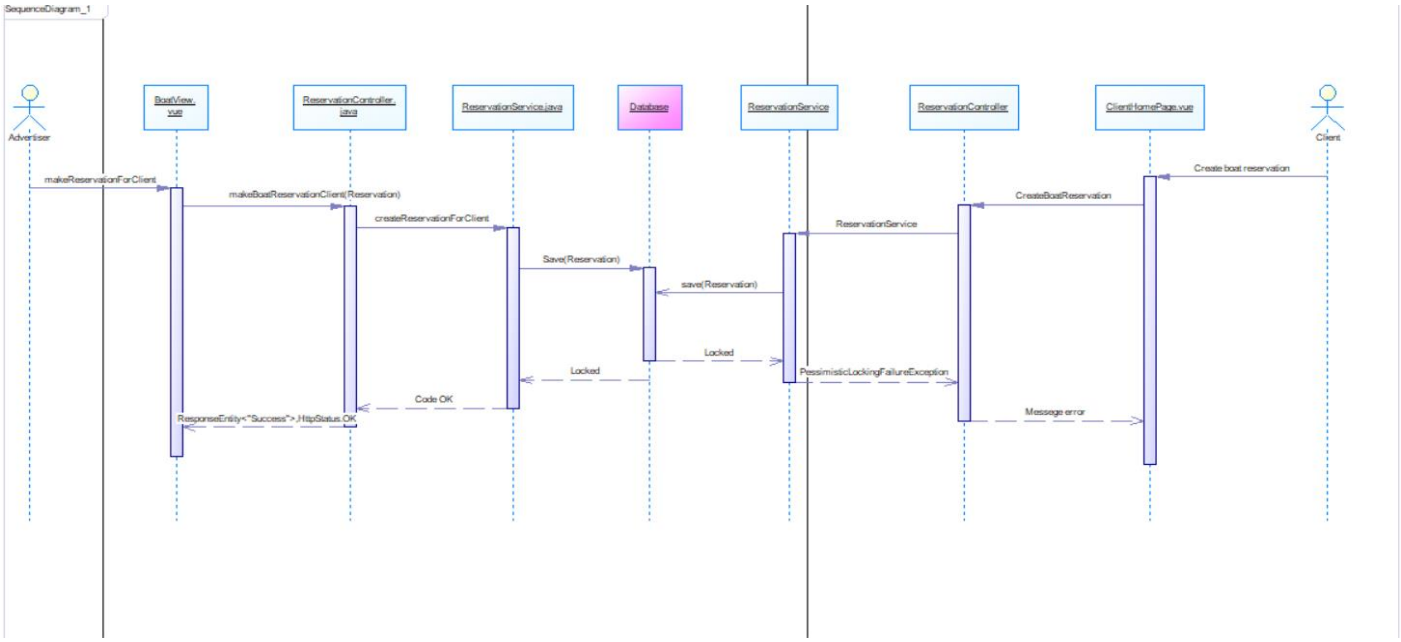
4.4 Konkurentni pristup resursima u bazi

1. Konfliktna situacija: Vlasnik vikendice/broda ili instruktor kreira rezervaciju u isto vreme kada i drugi klijent

Pored klijenta, vikendicu, brod I avanturu može da rezerviše i vlasnik ili instruktor za klijenta koji u toku ima aktivnu rezervaciju. Međutim, ako oni pristupe resursima u isto vreme može se desiti da i oglašivač (vlasnik ili instruktor) i klijent uspešno rezervišu isti entitet u preklapajuće vreme. Zbog toga je prilikom rezervacije neophodno obezbediti da klijent i oglašivač ne mogu u istom trenutku da izvrše rezervaciju istog entiteta.

```
makeReservationForClient(){
  axios.post( url: devServer.proxy+ "/makeBoatReservationClient",  data: {
```



Slika 1. dijagram sekvence prve konfliktne situacije

Rešenje konfliktne situacije: konfliktne situacije se nalazi u klasama *ReservationController, BoatReservationService* (*MansionReservationService, AdventureReservationService*), kao i u interfejsima za rad sa bazom *BoatRepository, MansionRepository* i *AdventureRepository*.

U interfejsima je odrađeno pesimističko zaključavanje *findLockerById(Long id*) metode na nivou jednog reda u tabeli koji predstavlja entitet. Korišten je *PESSIMISTIC_WRITE* kao tip zaključavanja. U servisima metode *CreateReservationForClien*t i *CreateReservation* pozivaju metodu *FindLockedById* iz interfejsa. U klasi *ReservationController* je dodato rukovanje izuzetkom, te oglašivač ili klijent dobija odgovarajuću poruku.

```
Boat entity = boatRepo.findLockedById(res.getEntityId());
if (entity == null)
    throw new PessimisticLockingFailureException("Some is already trying to reserve at the same time!");
```

Slika 2. Servis pozivanje metode findLockedById

```
catch (PessimisticLockingFailureException pe){
    System.out.println(pe);
    return new ResponseEntity<>( body: "Some is already trying to reserve at the same time!", HttpStatus.OK);
```

Slika 3. Vraćanje odgovarajuće poruke nakon neuspešne transakcije

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Boat findLockedById(Long id);
```

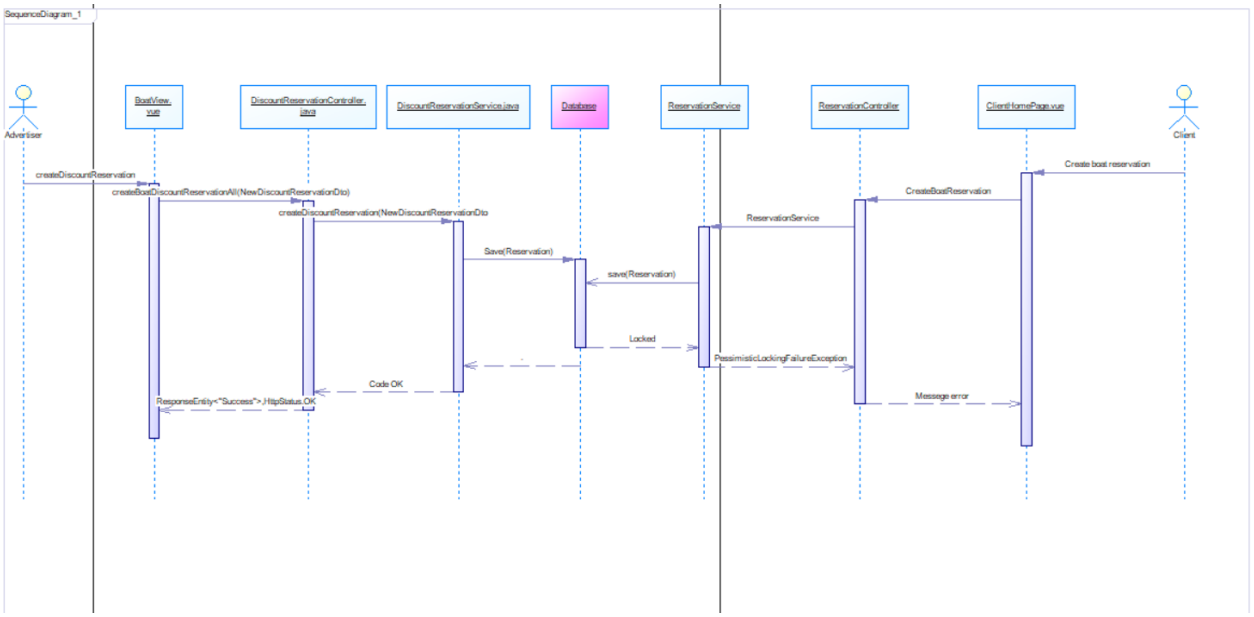
Slika 4. metoda u interfejsima

Pesimističkim zaključavanjem jednog reda u tabeli obezbedili smo nemogućnost rezervacije enetiteta u isto vreme od strane oglašivača i klijenta, međutim ovo će onemogućiti rezervaciju čak i u slučaju da se termini rezervacija ne poklapaju.

2. Konfliktna situacija: Vlasnik vikendice/broda ili ne može da napravi akciju u isto vreme kada I drugi klijent vrši rezervaciju

Oglašivačima je omogućeno da kreiraju brze rezervacije(akcije) koje klijenti mogu kasnije d arezervišu jednim klikom. Prilikom kreiranja brze rezervacije može se desiti da klijent vrši standardnu rezervaciju istog entiteta, ovo može da izazove konflikt I da obična rezervisana rezervacija I brza rezervacija imaju preklapajuće termine.

```
axios.post( url: devServer.proxy + "/createDiscountBoatReservation",  data: {
  "boatId" : this.boatToShow.id,
  "startDate" : this.startDateTimeQuick
```



Slika 5. dijagram sekvence druge konfliktne situacije

Rešenje konfliktne situacije: konfliktne situacije se nalazi u klasama *ReservationController, DiscountReservationController*, *BoatReservationService* (*MansionReservationService, AdventureReservationService*), *BoatDiscountReservationService (MansionDiscountReservationService, AdventureDiscountReservationService)* kao i u interfejsima za rad sa bazom *BoatRepository, MansionRepository* i *AdventureRepository*.

U interfejsima je odrađeno pesimističko zaključavanje *findLockerById(Long id*) metode na nivou jednog reda u tabeli koji predstavlja entitet. Korišten je *PESSIMISTIC_WRITE* kao tip zaključavanja. U servisima metode *createDiscountReservation* i *CreateReservation* pozivaju metodu *FindLockedById* iz interfejsa. U klasi *ReservationController* je dodato rukovanje izuzetkom, te oglašivač dobija odgovarajuću poruku.

```
@Override
@Transactional(readOnly=false,propagation= Propagation.REQUIRED,isolation= Isolation.SERIALIZABLE)
public int createDiscountReservation(NewDiscountReservationDto dto) {
    BoatDiscountReservation boatDiscountReservation = new BoatDiscountReservation();
    Boat entity = boatRepo.findLockedById(dto.boatId);
    if (entity == null)
        throw new PessimisticLockingFailureException("Some is already trying to reserve at the same time!");
```

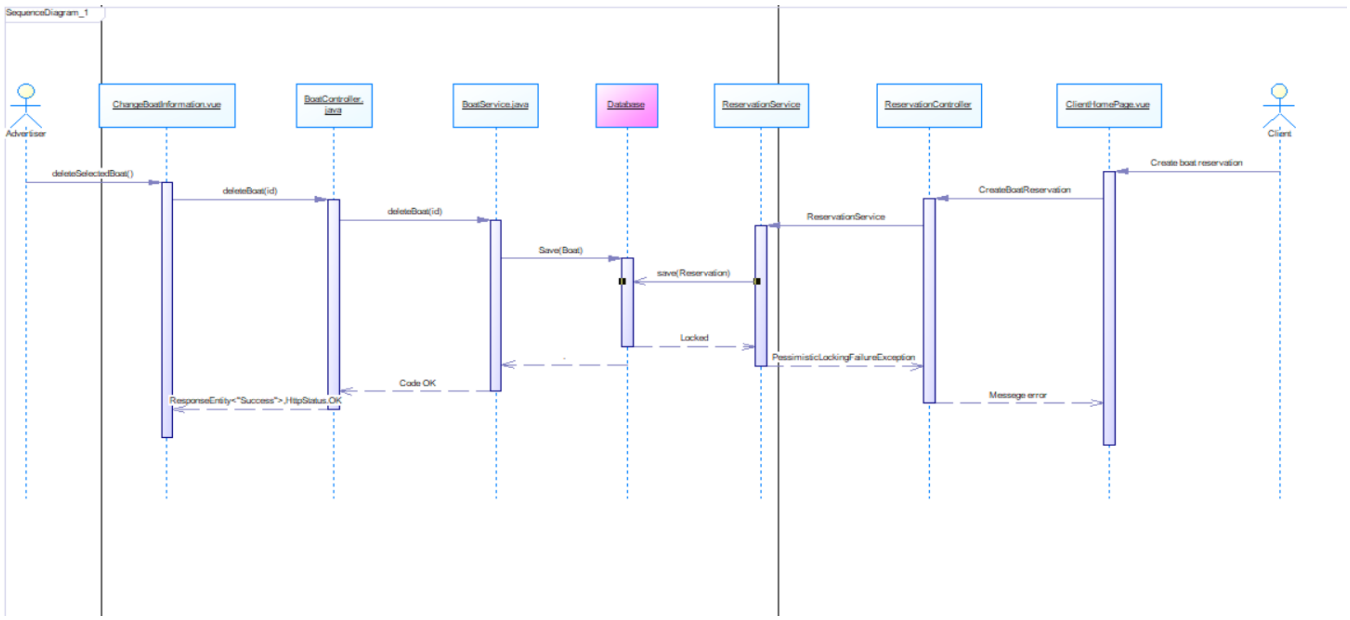
Slika 6. DiscountReservationService pozivanje metode findLockedById

Ista situacija kao i u prethodnoj konfliktnoj situaciji pesimističkim zaključavanjem jednog reda u tabeli obezbedili smo nemogućnost rezervacije enetiteta u isto vreme od strane klijenta i kreiranje akcije od strane oglašivača, međutim ovo će onemogućiti rezervaciju čak i u slučaju da se termini akcije i rezervacije ne poklapaju.

3. Konfliktna situacija: Vlasnik vikendice/broda ili ne može da obriše brod/vikendicu ili avanturu u isto vreme kada je klijent rezerviše

Oglašivači mogu da orišu svoje entitete ako u budućnosti ne postoji nijenda rezervacija za njih. U slučaju da postoji zahtev za brisanje će odmah biti odbijen, međutim ako ne postoji može se desiti da klijent pokušava da rezerviše u isto vreme kada oglašivač želi da obriše, pa će se desiti da klijent ima rezervaciju nad obrisanim entitetom jer je brisanje entiteta logičko. Ne postoji uslov da ne sme da postoji akcija koja nije rezervisana da bi entitet mogao da se obriše, pa je samim tim ovaj konflikt moguć u slučaju brze i obične rezervacije.

```
deleteSelectedBoat(){
  axios.post( url: devServer.proxy + "/deleteBoat", data: {
    "boatId" : this.selectedBoat.id
  },  config: {
```



Slika 7. dijagram sekvence druge konfliktne situacije

Rešenje konfliktne situacije:

Analogno prethodnim slučajevima, rešenje se radi pesimističkim zaključavanjem entiteta. Ovog puta nema loše strane ove metode, jer je bitno da se entitet zaključa.

Rešenje konfliktne situacije se nalazi u klasama *ReservationController, MansionController, BoatController, AdventureController, implementacijama ReservationService, DiscountReservationService* kao i u interfejsima za rad sa bazom *BoatRepository, MansionRepository* i *AdventureRepository.*

U interfejsima je odrađeno pesimističko zaključavanje *findLockerById(Long id*) metode na nivou jednog reda u tabeli koji predstavlja entitet. Korišten je *PESSIMISTIC_WRITE* kao tip zaključavanja. U servisima metoda *CreateReservation, DeleteBoat, DeleteMansion i DeleteAdventure* pozivaju metodu *FindLockedById* iz interfejsa. U klasi *ReservationController*, *BoatController, MansionController* i *AdvertiserController* je dodato rukovanje izuzetkom, te oglašivač dobija odgovarajuću poruku, kao što je prikazano na slici 8.

```
@PreAuthorize("hasRole('ROLE_BOAT_OWNER')")
@RequestMapping(method = RequestMethod.POST, value = "/deleteBoat",consumes = MediaType.APPLICATION_JSON_VALUE
public ResponseEntity<String> deleteBoat(@RequestBody LongIdDto boatId){
    if(service.isReserved(boatId.boatId)){
        return new ResponseEntity<>( body: "Boat is reserved, not possible to change or delete!", HttpStatus.OK);
    }
    System.out.println("I'm trying to delete boat id:"+ boatId.boatId);
    try {
        service.deleteBoat(boatId.boatId);
        return new ResponseEntity<String>( body: "Successfully deleted boat!", HttpStatus.OK);
    }
    catch (PessimisticLockingFailureException pe){
        return new ResponseEntity<>( body: "Client is reserving the entity!", HttpStatus.OK);
    }catch (Exception e) {
        return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Slika 8. poruka u slučaju konflikta u klasi BoatController

```
@Transactional(readOnly=false,propagation= Propagation.REQUIRED,isolation= Isolation.SERIALIZABLE)
public void deleteBoat(Long boatId) {
    Boat entity = boatsRepository.findLockedById(boatId);
    if (entity == null)
        throw new PessimisticLockingFailureException("Some is already trying to reserve!");
    Boat boat = boatsRepository.findById(boatId).get();
    boat.setDeleted(true);
    boatsRepository.save(boat);
}
```

```
@Override
@Transactional(readOnly=false,propagation= Propagation.REQUIRED,isolation= Isolation.SERIALIZABLE)
public int createDiscountReservation(NewDiscountReservationDto dto) {
    BoatDiscountReservation boatDiscountReservation = new BoatDiscountReservation();
    Boat entity = boatRepo.findLockedById(dto.boatId);
    if (entity == null)
        throw new PessimisticLockingFailureException("Some is already trying to reserve at the same time!");
```

Slika 9. pozivanje findLockedById iz interfejsa u sevisima

Napomena: Ilustrativno je sve prikazano na primeru Broda, međutim analogno je odrađeno I za ostale entitete (vikendica I avantura).