Konkurentni pristup bazi (Teodora Maruna RA117-2018)

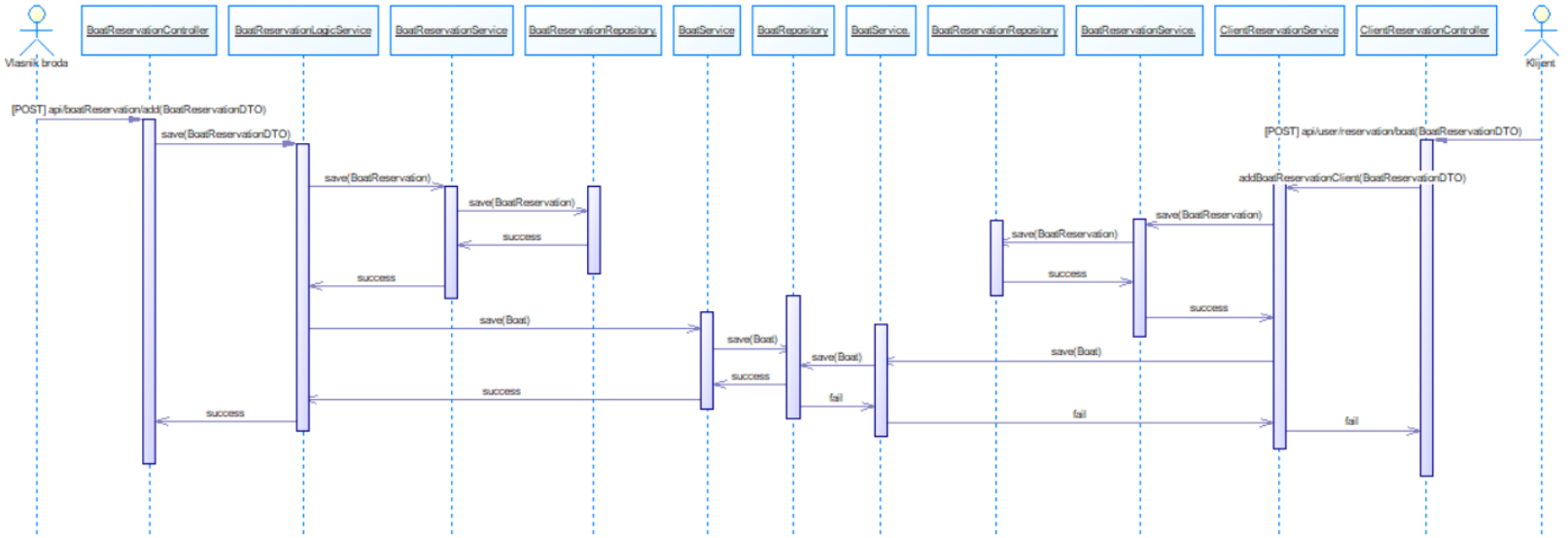
1. Postupak kreiranja rezervacije

Opis konfliktne situacije:

Vlasnici vikendica/brodova, instruktori i klijenti mogu da kreiraju rezervacije. Prilikom kreiranja rezervacije, moze doći do konfliktne situacije ukoliko vlasnik ili instruktor pokuša da napravi rezervaciju za klijenta u isto vreme kao i klijent. Problem se može rešiti korišćenjem optimističkog zaključavanja.

Crtež toka zahteva:

Crtež toka zahteva je prikazan za primer situacije kada vlasnik broda i klijent pokušaju da kreiraju rezervaciju u istom terminu. U aplikaciji je pokriven primer i za situaciju kada vlasnik vikendice i klijent pokušaju da kreiraju rezervaciju u istom terminu.



Opis načina na koji je problem rešen:

Ova konfliktna situacija je resena na osnovu optimističkog zaključavanja. Optimističko zaključavanje uvodi novo polje - version (verziju) u klasu Boat. Ovo polje se koristi za proveru stanja entiteta. Osoba čija transakcija je prva pokrenuta će uspešno izvršiti kreiranje rezervacije. Na gore ilustrovanom primeru, vlasnik broda će uspešno kreirati rezervaciju. Prilikom pokušaja kreiranje rezervacije u istom terminu od strane klijenta, vrši se poređenje početne vrednosti polja verzija sa trenutnom vrednosti polja verzije (broda) i s obzirom da se te vrednosti ne poklapaju, dolazi do izuzetka i zato se druga transakcija neće izvršiti.

BoatReservationLogicService (save) - transactional

```java
@Transactional
public boolean save(BoatReservationDTO dto) throws MessagingException {
    Boat boat = this.boatService.getBoatById(dto.getBoatId());
    MyUser owner = this.myUserService.findUserByBoatId(dto.getBoatId());

    if (isReservationFree(dto, boat)) return false;

    BoatReservation boatReservation = setFields(dto, boat);
    boatReservation =this.boatReservationService.save(boatReservation);
    setOwnerPoints(dto, owner);
    setAdditionalServices(dto, boatReservation);
    boat.addBoatReservation(boatReservation);
    this.boatService.save(boat);

    sendEmailIfClient(dto, boat, boatReservation);
    sendEmailIfAction(dto, boat);

    return true;
}
```

```java
@Entity
public class Boat {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Version
    @Column(nullable = false)
    private Integer version;
```

ClientReservationService (addBoatReservationClient) - transactional

```java
@Transactional
public boolean addBoatReservationClient(BoatReservationDTO dto){
    ReservationCheckDTO reservationCheckDTO = getReservationCheckDTO(dto);
    Boat boat = this.boatservice.getBoatById(dto.getBoatId());
    MyUser guest = this.myUserService.findUserById(dto.getGuestId());
    boolean isAvailable = this.boatReservationService.findBoatAvailability(reservationCheckDTO, boat.getId());
    if(isAvailable){
        Date startDate = new Date(dto.getMilisStartDate());
        Date endDate = new Date(dto.getMilisEndDate());
        BoatReservation boatReservation = new BoatReservation(dto.getId(), startDate, endDate, dto.getMaxGuests(),
                dto.getPrice(), dto.isAvailable(), boat);
        boatReservation.setAvailabilityPeriod(dto.isAvailabilityPeriod());
        boatReservation.setAction(dto.isAction());
        boatReservation.setGuest(guest);
        guest.setPoints(guest.getPoints() + this.companyService.getCompanyInfo((long) 1).getPointsPerReservationClient());
        this.checkUserCategory(guest);
        boatReservation = this.boatReservationService.save(boatReservation);
        addAdditionalServices(dto, boatReservation);
        sendEmailClient(guest,  houseName: "", boat.getName(),  adventureName: "");
    }
    return isAvailable;
}
```

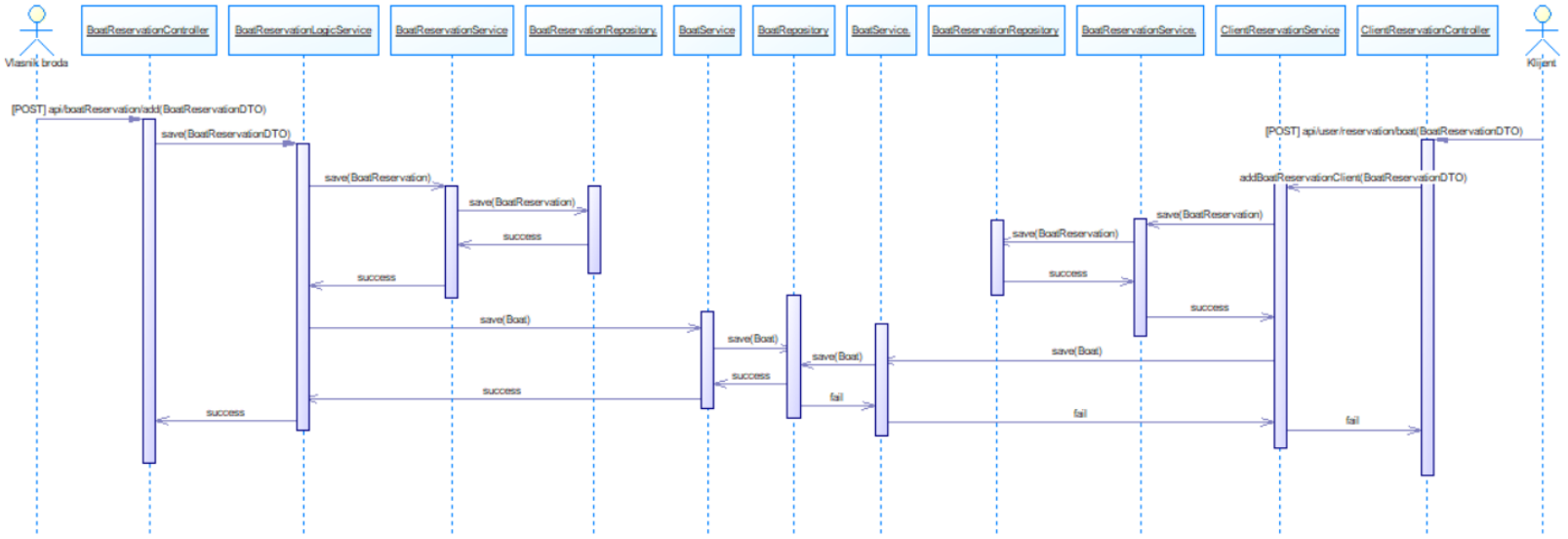
2. Postupak kreiranja akcije

Opis konfliktne situacije:

Prilikom kreiranja rezervacije, moze doći do konfliktne situacije ukoliko vlasnik ili instruktor pokuša da napravi akciju u isto vreme kao i klijent. Problem se može rešiti korišćenjem optimističkog zaključavanja.

Crtež toka zahteva:

Crtež toka zahteva je prikazan za primer situacije kada vlasnik broda pokuša da kreira akciju u određenom terminu a klijent pokuša da kreira rezervaciju u preklapajućem terminu.
U aplikaciji je pokriven i primer kada vlasnik vikendice pokuša da kreira akciju u nekom terminu a klijent pokuša da kreira rezervaciju u preklapajućem terminu.



Opis načina na koji je problem rešen:

Ova konfliktna situacija je resena isto kao i u prošlom primeru, s obzirom da se za kreiranje akcije koristi ista metoda kao i za kreiranje rezervacije. Konfliktnu situaciju rešavamo pomoću optimističkog zaključavanja. Optimističko zaključavanje uvodi novo polje - version (verziju) u klasu Boat. Ovo polje se koristi za proveru stanja entiteta.
Osoba čija transakcija je prva pokrenuta će uspešno izvršiti kreiranje rezervacije. Na gore ilustrovanom primeru, vlasnik broda će uspešno kreirati rezervaciju. Prilikom pokušaja kreiranje rezervacije u istom terminu od strane klijenta, vrši se poređenje početne vrednosti polja verzija sa trenutnom vrednosti polja verzije (broda) i s obzirom da se te vrednosti ne poklapaju, dolazi do izuzetka i zato se druga transakcija neće izvršiti.

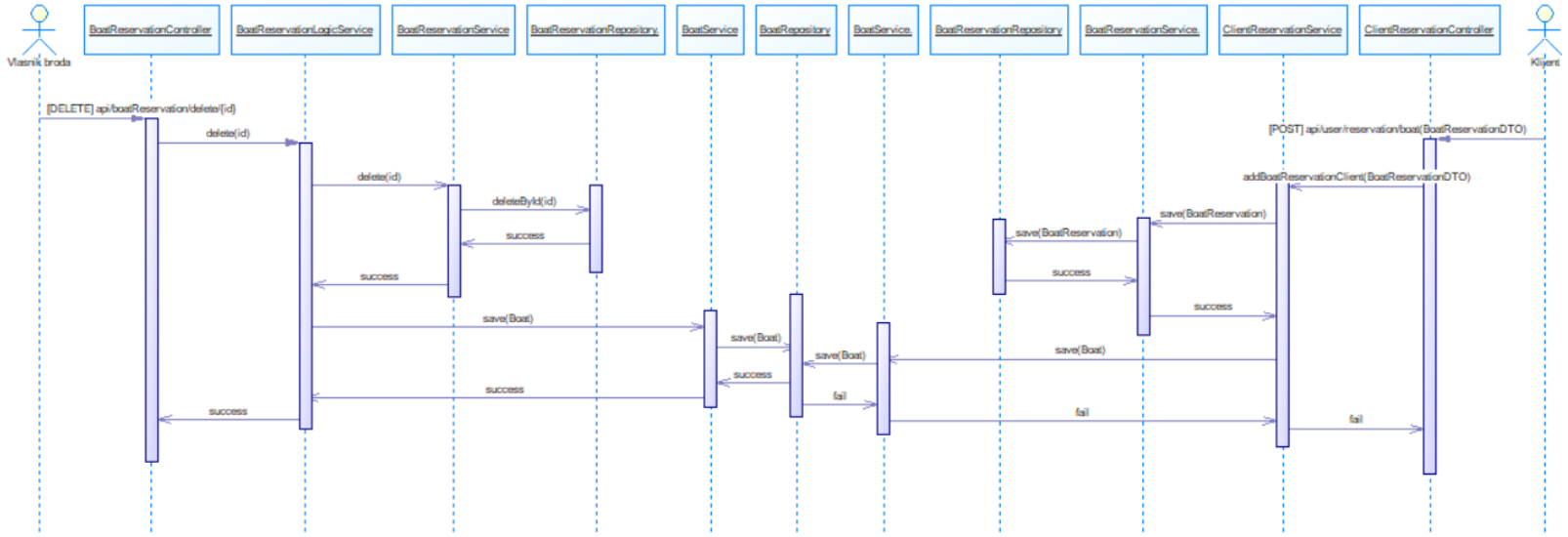
3. Postupak brisanja akcije vlasnika

Opis konfliktne situacije:

Moze doći do konfliktne situacije ukoliko vlasnik ili instruktor pokuša da izbriše akciju u isto vreme kada klijent pokuša da rezerviše tu akciju. Problem se može rešiti korišćenjem optimističkog zaključavanja.

Crtež toka zahteva:

Crtež toka zahteva je prikazan za primer situacije kada vlasnik broda pokuša da obriše akciju, a klijent pokuša da rezerviše baš tu akciju.



Opis načina na koji je problem rešen:

Ovu konfliktnu situaciju rešavamo pomoću optimističkog zaključavanja. Optimističko zaključavanje uvodi novo polje - version (verziju) u klasu Boat. Ovo polje se koristi za proveru stanja entiteta.
Osoba čija transakcija je prva pokrenuta će se uspešno izvršiti. Na gore ilustrovanom primeru, vlasnik broda će uspešno obrisati akciju (jer je ta transakcija prva pokrenuta). Prilikom pokušaja rezervisanja akcije, vrši se poređenje početne vrednosti polja verzija sa trenutnom vrednosti polja verzije i s obzirom da se te vrednosti ne poklapaju, dolazi do izuzetka i zato se druga transakcija neće izvršiti.

BoatReservationLogicService (delete) - transactional

```
@Transactional
public boolean delete(Long id) {
    BoatReservation boatReservation = this.boatReservationService.getBoatReservationById(id);

    removeAdditionalServices(boatReservation);
    boatReservation.setGuest(null);
    boatReservation = this.boatReservationService.save(boatReservation);
    this.boatReservationService.delete(boatReservation.getId());

    Boat boat = this.boatService.getBoatById(boatReservation.getBoat().getId());
    boat.removeBoatReservation(boatReservation);
    this.boatService.save(boat);

    return true;
}
```