KONKURENTAN PRISTUP RESURSIMA U BAZI

KONFLIKT 1

Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta u isto (ili preklapajuće) vreme

Jedan od mogućih problema jeste situacija u kojoj više korisnika istovremeno pokuša da rezerviše isti entitet. Ukoliko bi oba korisnika za isto, ili barem preklapajuće vreme odradila proveru dostupnosti jednog entiteta, obe provere bi vratile validan rezultat, te bismo bili u **riziku dualne zauzetosti** jednog objekta, ili avanture. U idealnoj situaciji, ovaj problem bio bi regulisan time što bi samo jedan korisnik uspešno izvršio rezervaciju entiteta, dok bi drugi bili obavešteni o grešci.

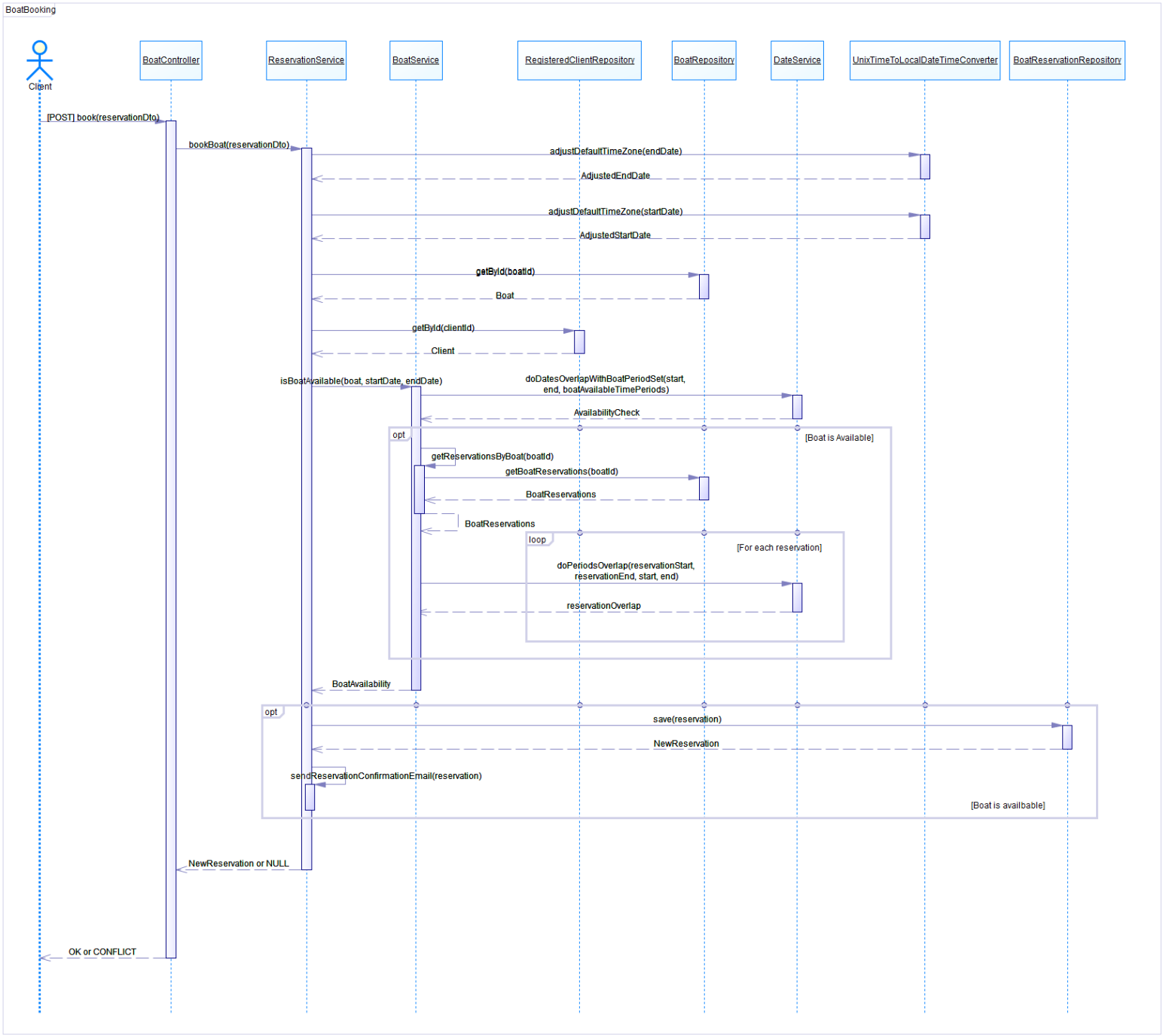
Kako se ovaj problem javlja na tri ekvivalentna mesta: prilikom rezervacije brodova, vikendica i avantura, **dijagram toka** prikazaćemo za prvu situaciju. Postupak se odvija na isti način, respektivno za druge tipove entiteta. Radi pojednostavljenja dijagrama i očuvanja konteksta problema, proces slanja mejla konfirmacije rezervacije je predstavljen kao *black box*. ***[Strana 2, slika 1]***

Solucija:

Problem rešavamo postavljanjem anotacije *@Transactional* na metode klase ReservationService *bookBoat(), bookAdventure()* i *bookCottage().* Dodatno, koristili smo otpimističko zaključavanje dodavanjem polja version u superklasu rezervacije sa anotacijom *@Version*.

```java
@Getter
@Setter
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Reservation extends DatabaseEntity {

    @Version
    @Column(name = "version", nullable = false)
    private Long version = 1L;

    @Column(name = "reservationEnd", nullable = false)
    private LocalDateTime reservationEnd;
```

```java
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public CottageReservation bookCottage(ReservationDto reservationDto) {...}

@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public AdventureReservation bookAdventure(ReservationDto reservationDto) {...}

@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public BoatReservation bookBoat(ReservationDto reservationDto) {...}
```



Slika 1 - Rezervacija broda

KONFLIKT 2

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

Kako su akcijske ponude dostupne svim prijavljenim klijentima za brzu rezervaciju, lako je moguće da dva klijenta u isto, ili približno isto vreme pokušaju da rezervišu istu ponudu. Takođe je moguće da isti klijent pokuša da napravi iz dva različita prozora rezervaciju istog entiteta, ili 2 različita čija su vremena trajanja preklapajuća ili ista. Oba slučaja dovela bi do nekonzistentosti baze sa mogućnostima realnog sistema, te je u cilju iste izbeći.

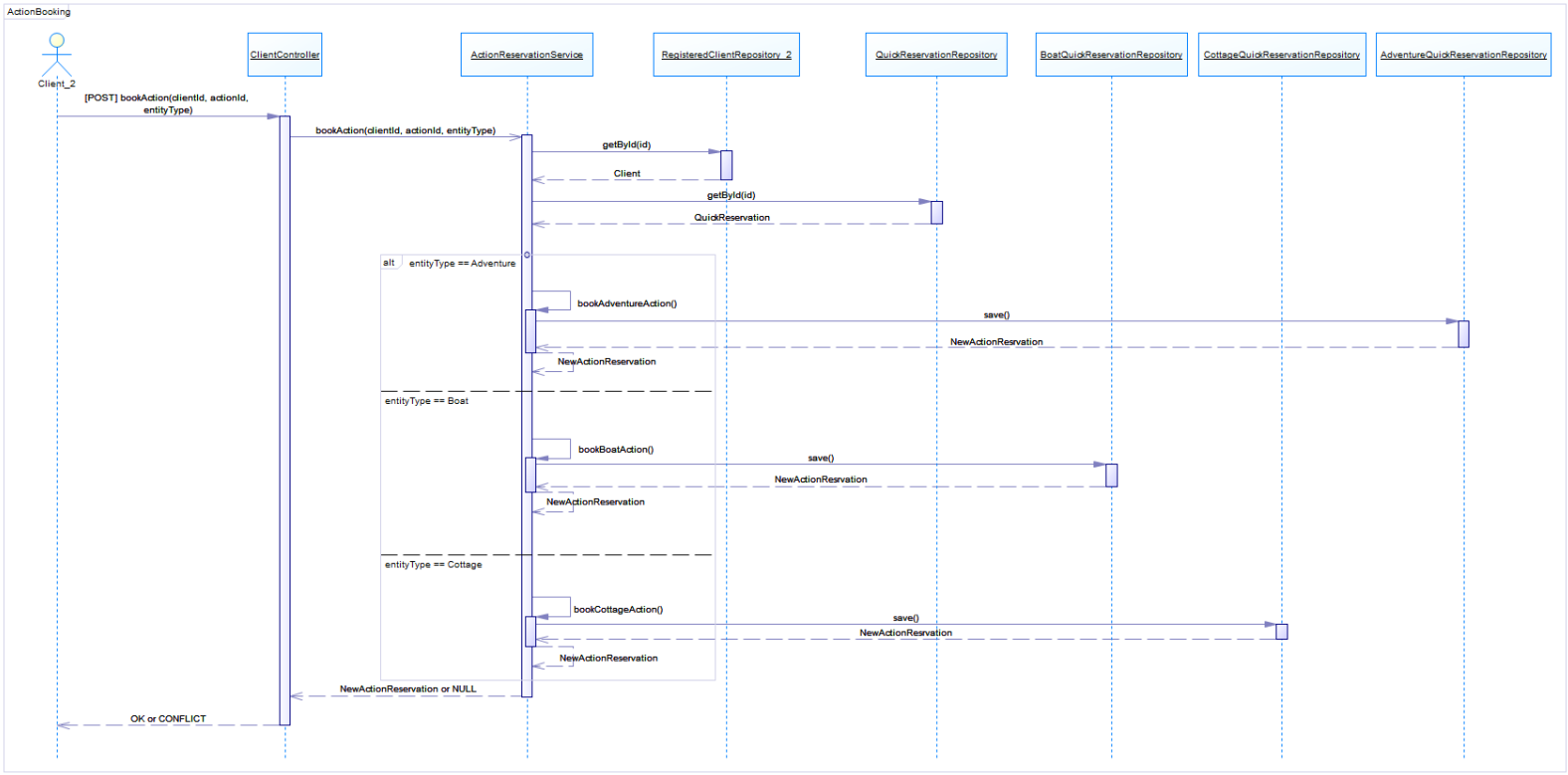
Na **dijagramu sekvence** prikazan je tok zahteva prilikom prijave na specijalnu akciju. U zavisnosti od uspeha poziva, klijentu će biti vraćen ili HTTP odgovor OK, ili CONFLICT. Princip kreiranja nove rezervacije je sličan procesu opisanom u prethodnom grafu, te su neki od poziva manjih funkcija izostavljeni, radi lepšeg prikaza grafa. ***[Strana 4, slika 2]***

Solucija:

Problem rešavamo postavljanjem anotacije *@Transactional* na metodu klase ActionReservationService *bookAction().* Dodatno, koristili smo otpimističko zaključavanje dodavanjem polja version u superklasu brze rezervacije sa anotacijom *@Version*.

```
@Override
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public Reservation bookAction(Long clientId, Long actionId, String type)
```

```
@Getter
@Setter
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class QuickReservation extends DatabaseEntity {

    @Version
    @Column(name = "version", nullable = false)
    private Long version = 1L;
```



Slika 2 – Rezervacija akcija

KONFLIKT 3

Više klijenata ne može da pošalje zahtev za registraciju u isto vreme sa istom mejl adresom

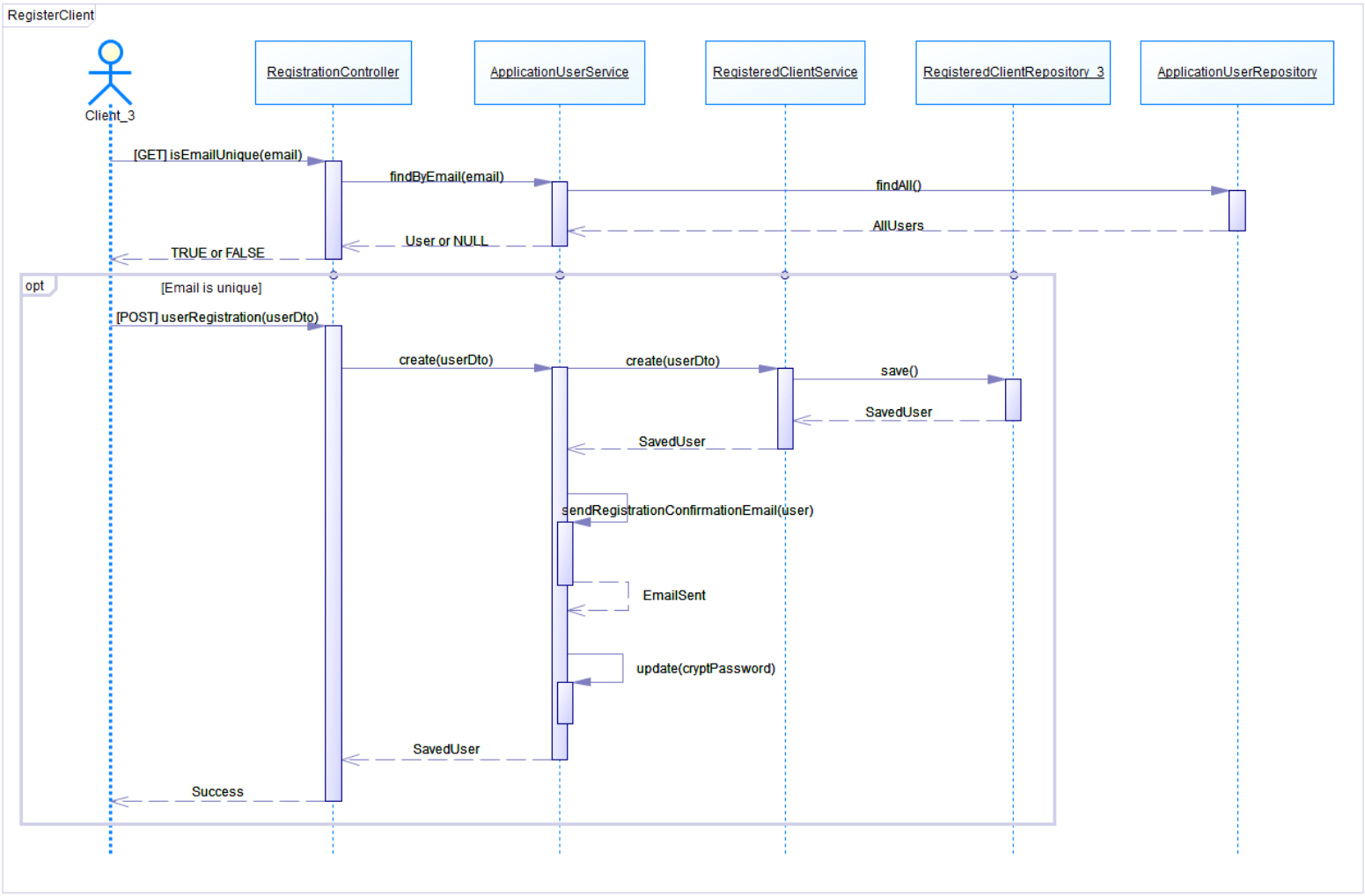
Kada korisnik pokuša da kreira nalog, najpre se izvršava provera unikatnosti unešene mejl adrese. U slučaju da je ista već zauzeta, korisnkiu se na stranici forme ispisuje poruka o grešci, i neophodno je promeniti mejl adresu. U situaciji da više korisnika istovremeno pokrene ovaj zahtev, postoji mogućnost da oba zahteva prođu do kreacije entiteta, što bi dovelo do brojnih greški u sistemu.

Dijagram toka prikazuje proces korisničkih zahteva od pokretanja validacije forme klikom na „SUBMIT“. Proces prati samo kreiranje korisnika tipa klijent, mada je proces približno identičan i za druge tipove korisnika. ***[Strana 5, slika 3]***

Solucija:

Problem rešavamo postavljanjem anotacije *@Transactional* na metode klase ApplicationUserService *create()* i *findByEmail().*

```
@Override
@Transactional
public ApplicationUser findByEmail(String email) {...}

@Override
@Transactional
public ApplicationUser create(ApplicationUserDto userDto) {...}
```



Slika 3 - Registracija klijenta

Sve slike u originalnim rezolucijama, kao i PowerDesigner dokumenti mogu se naći u folderu ***docs/Transactions – Studetn 1.***