Student 2, Srđan Berić RA191-2018

4.4 Konkurentni pristup bazi podataka

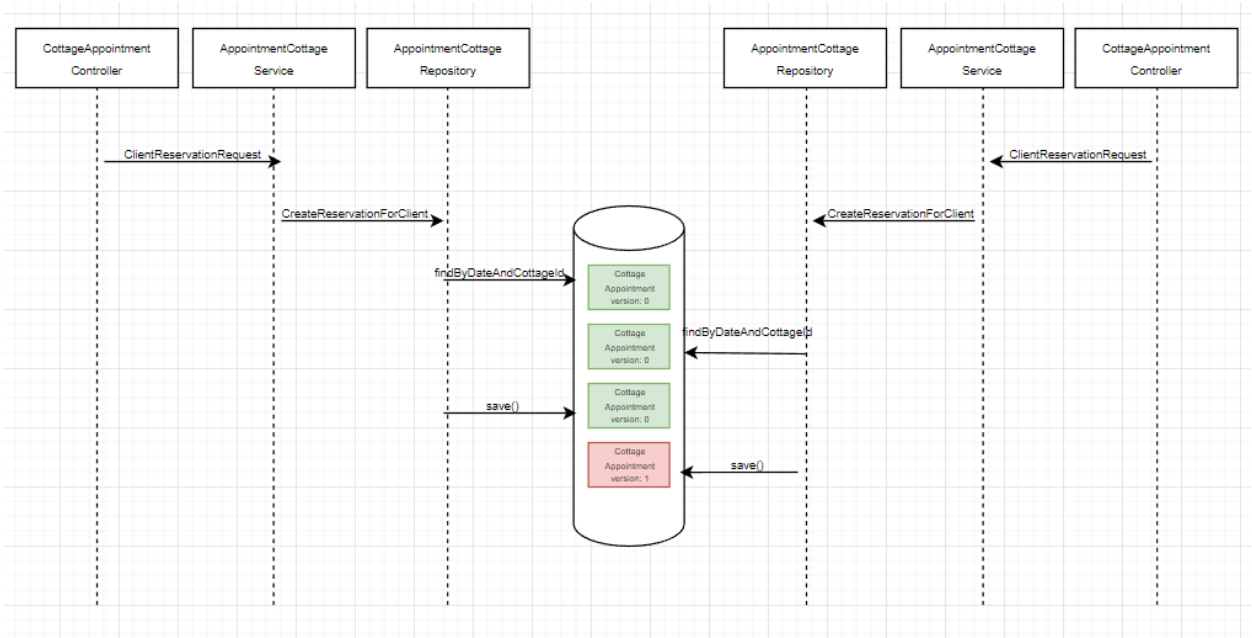
Uvod za scenario 1 i 2

Kada vlasnik vikendice definiše slobodne datume, kroz iteraciju se kreira lista termina (od početnog do krajnjeg datuma) koja sadrže datum, da li je termin na akciji, status (AVAILABLE, RESERVED) i ID klijenta. Kada klijent rezerviše termin, status prelazi iz Available u Reserved, polju client_id se dodeljuje vrednost.

	id [PK] bigint	date date	has_action boolean	price_per_day double precision	type character varying (255)	optlock integer	client_id bigint	cottage_id bigint
33	33	2022-08-18	false	10	RESERVED	0	2	1
34	34	2022-08-16	false	10	RESERVED	0	2	1
35	35	2022-08-19	false	10	AVAILABLE	0	[null]	1
36	36	2022-08-20	false	10	AVAILABLE	0	[null]	1
37	37	2022-08-21	false	10	AVAILABLE	0	[null]	1

Scenario 1: Više istovremenih klijenata ne može napraviti rezervaciju istog entiteta u isto vreme, ukoliko se datumi preklapaju.

Ovaj problem nastaje kada klijenti u isto vreme pokušavaju da rezervišu entitet u preklapajućim terminima. Problem takođe nastaje kada vlasnik rezerviše termin za klijenta u isto vreme kad i drugi klijenti pokušavaju da rezervišu preklapajući termin.



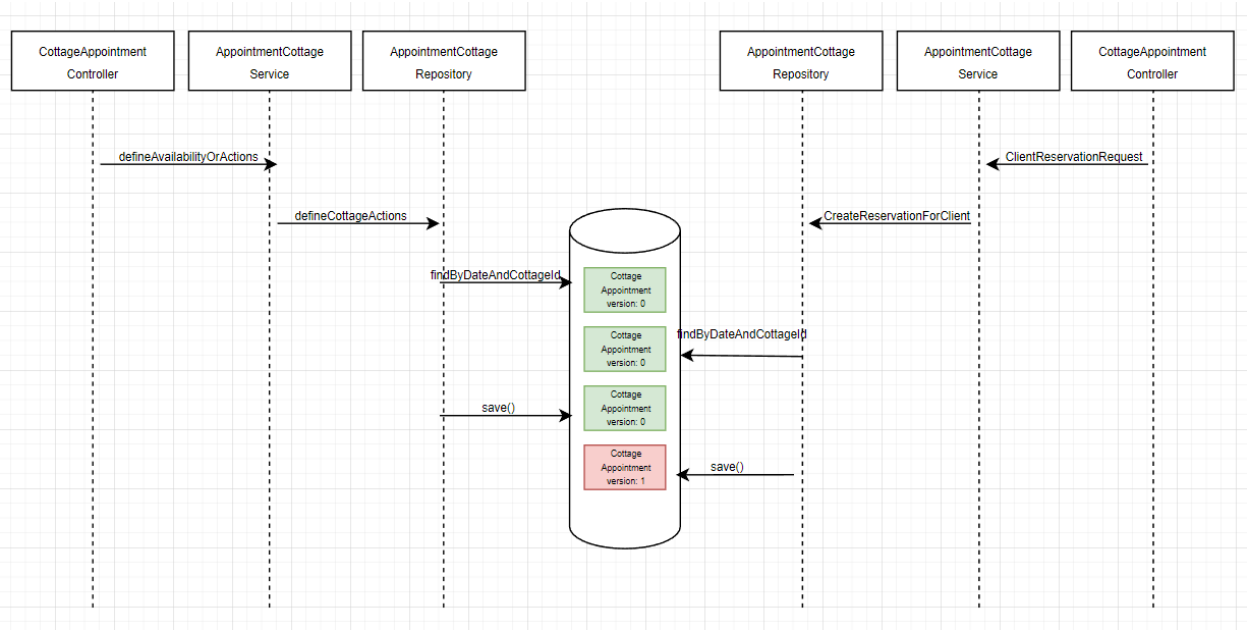
Rešenje: Ova situacija je rešena optimističkim zaključavanjem, odnosno verzioniranjem entiteta (CottageAppointment). Ukoliko jedan zahtev zauzme entitet i počne da ga menja, svaki sledeći zahtev biće odbijen.

```java
@Entity
public class AppointmentCottage extends Appointment {

    @Version
    @Column(name = "optlock", columnDefinition = "integer DEFAULT 0", nullable = false)
    private Long version = 0L;

    @ManyToOne(fetch = FetchType.EAGER)
}
```

```java
//Vlasnik vikendice kreira rezervaciju za klijenta
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public CottageReservationResponseDTO CreateReservationForClient (long clientId ,long cottageId,LocalDat
    //Ako pokusa da rezervise u terminima kada nije definisana dostupnos vikendice
    if(!existsByDateAndCottageId(startTime,cottageId)|| !existsByDateAndCottageId(endTime,cottageId)){
```

Scenario 2: Ukoliko vlasnik pravi akciju klijent ne može napraviti rezervaciju u istom terminu i obrnuto.

Ovaj problem nastaje kada klijent ili više klijenata pokušava da napravi rezervaciju u istom momentu kada i vlasnik pravi akciju za preklapajuće datume.

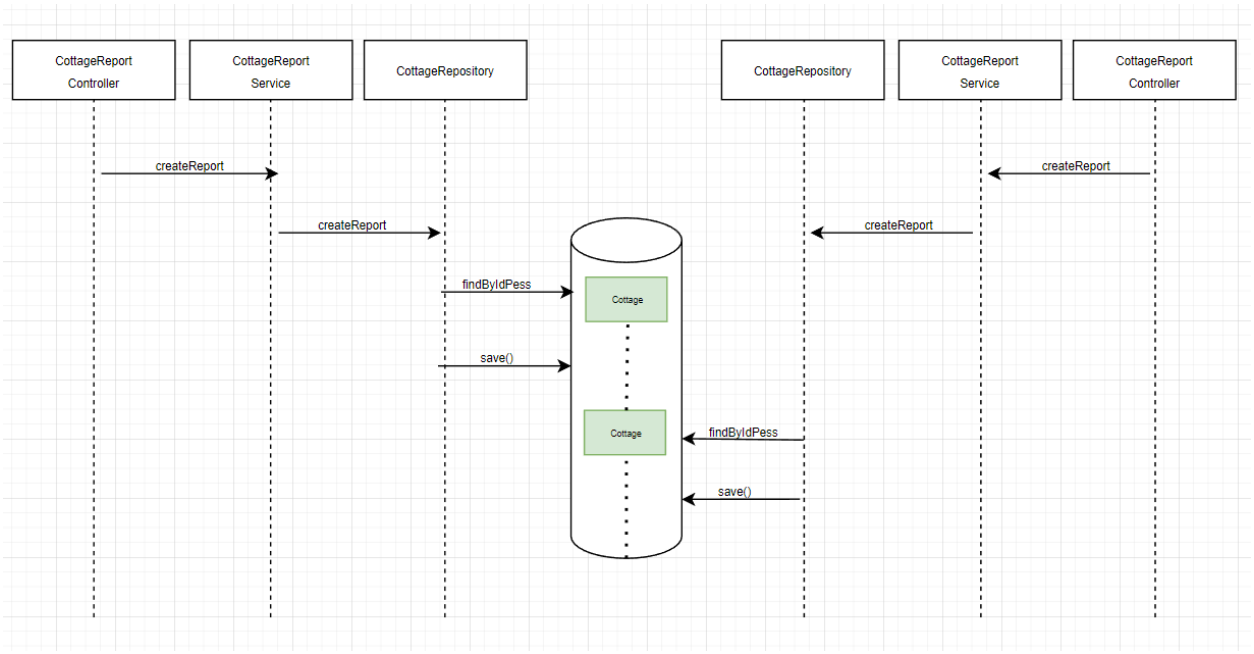


Rešenje: Ova situacija je rešena takođe optimistički, ovog puta zaključavanjem metode koja kreira akcije za određeni entitet. Svaki zahtev koji dospe tokom obavljanja prvog zahteva, biće odbijen.

```
}

@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public DefineCottageAvailabilityResponseDTO DefineCottageAction (DefineCottageAvailabilityRequestD
    Cottage cottage = cottageService.findById(request.getCottageId());
    ArrayList<AppointmentCottage> saveReservations = new ArrayList<>();
    for(LocalDate date = request.getStartDate(); date.isBefore(request.getEndDate().plusDays(1));
        //proveriti da li postoji slobodan/zauzet termin tog datuma, da ne bi doslo do dupliranja
```

Scenario 3: Vlasnik ne može u isto vreme da podnese više izveštaja za istu vikendicu

Ovaj situacija se može desiti ukoliko vlasnik u kratkom vremenskom razmaku inicira funkciju više puta, ili ukoliko sa različitih uređaja pokuša da podnese izveštaj.



Rešenje: Ova situacija je rešena pesimističkim zaključavanjem. Optimističkim zaključavanjem, odnosno verzioniranjem entiteta se ne može odstraniti problem, jer izveštaji u trenutku kreiranja još uvek ne postoje. Transakcija je urađena na način tako da se entitet zaključa kada je zahtev okupira, a otključa kada se pozove metoda save(). U repozitorijumu, na metodi za dobavljanje entiteta po ID-u postavljen je Lock, tako da će tokom transakcije samo jedan zahtev moći pristupiti tom entitetu.

```java
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select c from Cottage c where c.id=:id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})
public Cottage findByIdPess(@Param("id") long id);
```

```java
@Transactional(readOnly = false)
public CottageReport createReport (CottageReportDTO report) throws InterruptedException {
    //da li je klijent imao rezervaciju u toj vikendici i da li je ta rezervacija prosla (vraca broj takvih
    if(cottageReportRepository.clientHasReservedCottage(report.getClientId(), report.getCottageId())==0){
        return null;
    }
    Cottage cottage = cottageRepository.findByIdPess(report.getCottageId());
    CottageReport cottageReport = new CottageReport(cottageOwnerService.findById(cottage.getCottageOwner().
            report.getDescription(), LocalDateTime.now(), report.isReportClient(), report.isDidNotShowUp(),
```