Konkurentni pristup resursima u bazi

Student 2 | Đorđe Tovilovic IN-51/2018

Problem 1: Vlasnik vikendice/broda ili instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent.

U situaciji da vlasnik vikendice/broda ili instruktor napravi rezervaciju u isto vreme kad i drugi klijent može doći do problema zato što potencijalno mogu rezervisati isti entitet u periodu koji se preklapa što bi bio ozbiljan problem jer na primer vlasnik vikendice ne može da izda vikendicu u isto vreme različitim klijentima.

Rešenje:

Problem rešavamo tako što pesimistički zaključamo metodu za dobavljanje entiteta prilikom pravljenja nove rezervacije, tako da u jednom trenutku samo jedan klijent ili vlasnik ili instruktor može napraviti rezervaciju u jednom trenutku.

Metoda za kreiranje rezervacije za vikendicu:

```
@Transactional
public CottageReservationDto scheduleReservation(NewReservationDto newReservationDto, Long clientId) {
    try {
        Cottage cottage = cottageRepository.findByIdAndLock(newReservationDto.getEntityId());
        Client client = clientRepository.getById(clientId);

        clientRepository.save(client);

        CottageReservation cottageReservation = new CottageReservation();
        cottageReservation.setEntity(cottage);
        cottageReservation.setPrice(newReservationDto.getPrice());
        cottageReservation.setNumberOfGuests(newReservationDto.getNumberOfGuests());
        cottageReservation.setReservationStart(newReservationDto.getReservationStart());
        cottageReservation.setReservationEnd(newReservationDto.getReservationEnd());
        cottageReservation.setClient(clientRepository.getById(newReservationDto.getClientId()));
        if (newReservationDto.getActionId() != null)
            cottageActionRepository.deleteById(newReservationDto.getActionId());
        try {
            emailService.sendCottageReservationEmail(cottageReservation.getClient(), cottageReservation);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return modelMapper.map(cottageReservationRepository.save(cottageReservation), CottageReservationDto.class);

    } catch (PessimisticLockingFailureException e) {
        throw new LockingFailureException();
    }
}
```

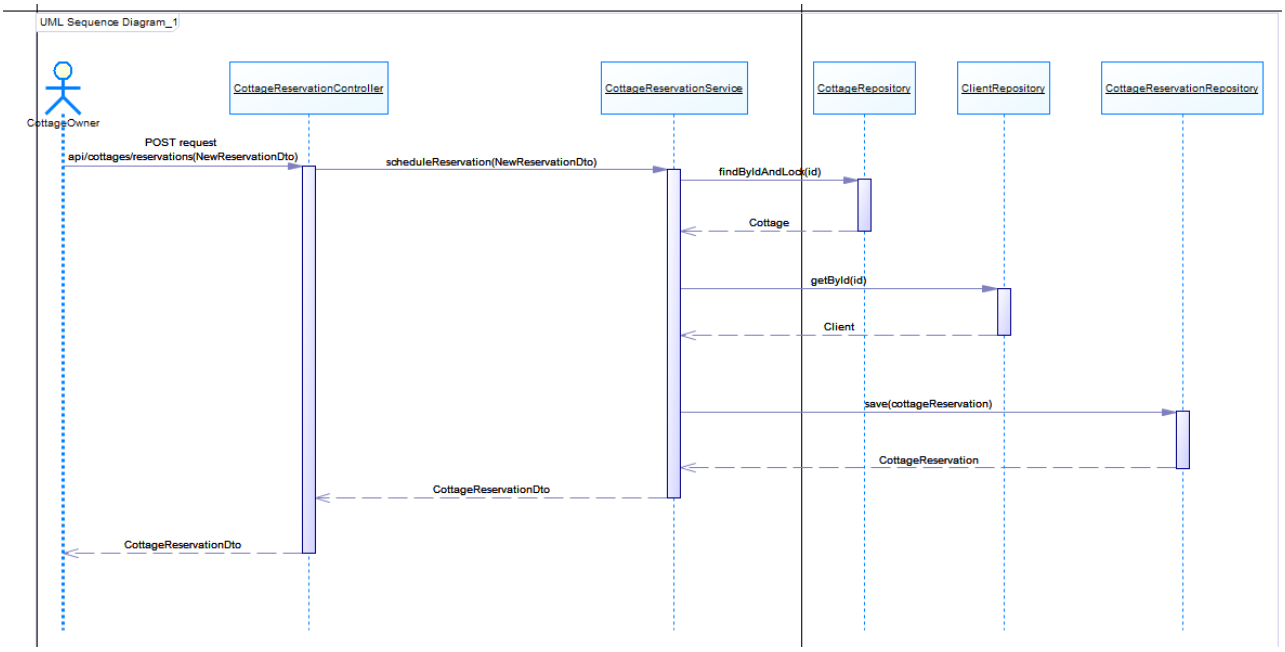
Kreiranje rezervacije za druga dva entiteta su identično urađena

Metoda za dobavljanje i pesimističko zaključavanje vikendice:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Cottage c WHERE c.id = :id")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Cottage findByIdAndLock(Long id);
```

Pesimističko zaključavanje druga dva entiteta su identično urađena

Dijagram toka kreiranja nove rezervacije:



Vlasnik i klijent koriste isti endpoint prilikom pravljenja rezervacije, pa je tok isti u oba slučaja

Problem 2: Vlasnik vikendice/broda ili instruktor ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta.

U situaciji da vlasnik vikendice/broda ili instruktor napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta može doći do problema zato što potencijalno vlasnik ili instruktor može napraviti akciju za entitet u periodu koji se preklapa sa periodom u kom klijent vrši rezervaciju što bi značilo da bi postojala akcija za entitet u periodu kada on nije dostupan, što bi takođe moglo dovesti do toga da entitet bude rezervisan 2 puta u istom vremenskom periodu.

Rešenje:

Problem takođe rešavamo pesimističkim zaključavanjem metode za dobavljanje entiteta prilikom pravljenja nove akcije kao i kod pravljenja nove rezervacije , tako da u jednom trenutku samo jedan klijent ili vlasnik ili instruktor može napraviti akciju ili rezervaciju u jednom trenutku.

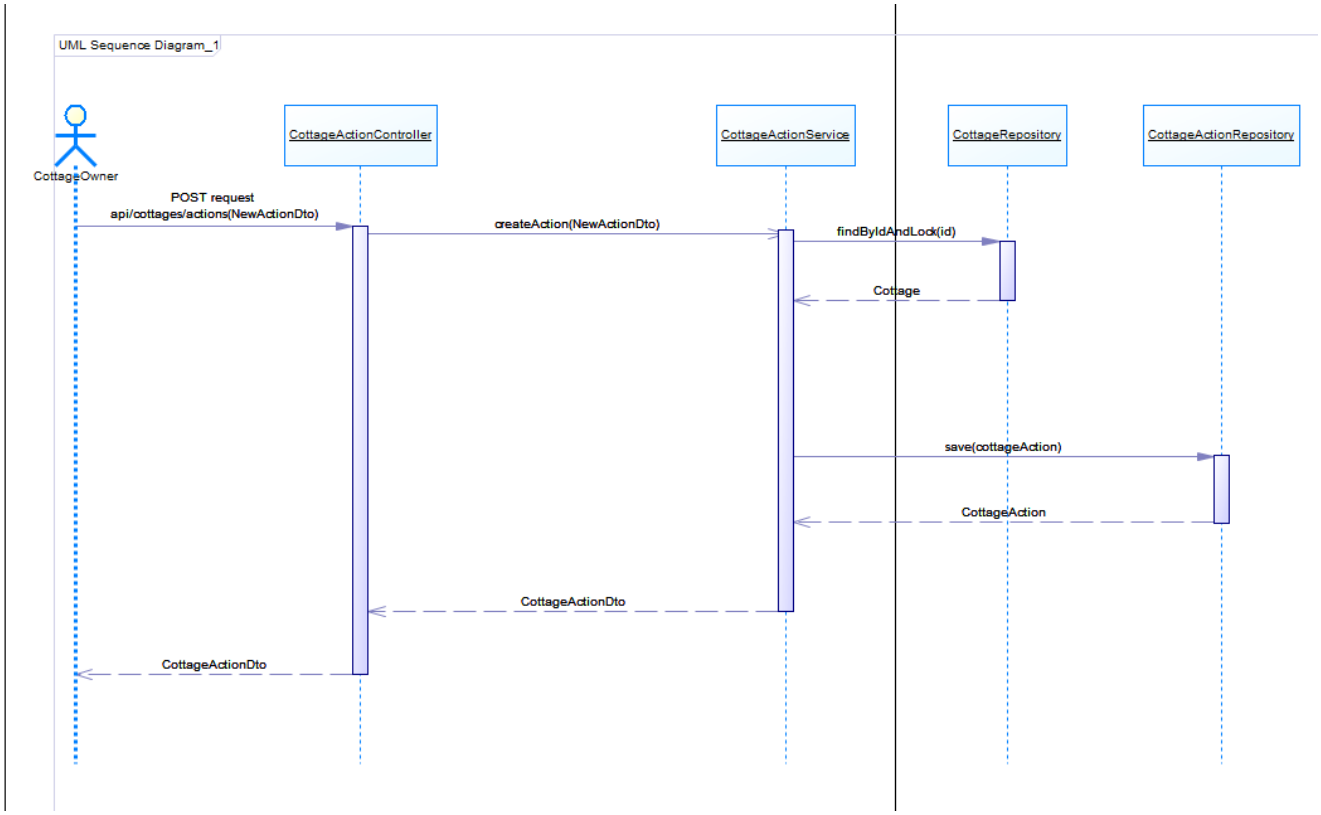
Metoda za kreiranje akcije za vikendicu:

```
@Transactional
public CottageActionDto createAction(NewActionDto newNewActionDto) {
    try {
        Cottage cottage = cottageRepository.findByIdAndLock(newNewActionDto.getEntityId());

        CottageAction cottageAction = new CottageAction();
        cottageAction.setEntity(cottage);
        cottageAction.setActionPrice(newNewActionDto.getActionPrice());
        cottageAction.setActualPrice(newNewActionDto.getActualPrice());
        cottageAction.setNumberOfGuests(newNewActionDto.getNumberOfGuests());
        cottageAction.setReservationStart(newNewActionDto.getReservationStart());
        cottageAction.setReservationEnd(newNewActionDto.getReservationEnd());
        cottageAction.setActionEnd(newNewActionDto.getActionEnd());
        CottageActionDto cottageActionDto = modelMapper.map(cottageActionRepository.save(cottageAction), CottageActionDto.class);

        clientSubscriptionRepository.findAllByEntityId(newNewActionDto.getEntityId());
        List<ClientSubscription> allSubs = clientSubscriptionRepository.findAllByEntityId(newNewActionDto.getEntityId());

        for (ClientSubscription sub : allSubs) {
            mailService.sendNewCottageActionEmail(sub.getClient(), cottageAction);
        }
        return cottageActionDto;

    } catch (PessimisticLockingFailureException e) {
        throw new LockingFailureException();
    }
}
```

Kreiranje akcije za druga dva entiteta su identično urađena

Koriste se iste metode za pesimističko zaključavanje i dobavljanje entiteta kao u prethodnom primeru.

Dijagram toka kreiranja nove akcije za vikendicu:



Problem 3: Vlasnik vikendice/broda ili instruktor ne može da izmeni period dostupnosti entiteta u isto vreme kad i drugi klijent vrši rezervaciju istog entiteta.

U situaciji da vlasnik vikendice/broda ili instruktor menja period dostupnosti entiteta u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta može doći do problema zato što potencijalno vlasnik ili instruktor može staviti dostupnost entiteta van perioda u kom klijent vrši rezervaciju što bi značilo da bi klijent rezervisao entitet u periodu kada on nije dostupan.

Rešenje:

Problem takođe rešavamo pesimističkim zaključavanjem metode za dobavljanje entiteta prilikom izmene perioda dostupnosti kao i kod pravljenja nove rezervacije , tako da u jednom trenutku samo jedan klijent ili vlasnik ili instruktor može menjati period dostupnosti ili praviti rezervaciju u jednom trenutku.

Metoda za ažuriranje vikendica preko koje se menja period dostupnosti:

```
@Transactional
public CottageDto update(CottageDto newCottage) {
    modelMapper.getConfiguration().setPropertyCondition(Conditions.isNotNull());
    Cottage cottage = cottageRepository.findByIdAndLock(newCottage.getId());
    modelMapper.map(newCottage, cottage);

    return modelMapper.map(cottageRepository.save(cottage), CottageDto.class);
}
```

Ažuriranje druga dva entiteta su identično urađena

Koriste se iste metode za pesimističko zaključavanje i dobavljanje entiteta kao u prethodnim primerima.

Dijagram toka ažuriranja perioda dostupnosti vikendice: