Konkurentni pristup resursima u bazi

Student 1: Srđan Topić IN19/2018

Problem 1: Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta u isto vrijeme

Problem 2: Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vrijeme

Napomena: Problemi 1. i 2. koriste identične metode, te se rješenje odnosi na oba problema.

U situaciji da prvi klijent napravi rezervaciju u isto vrijeme kad i drugi klijent može doći do problema zato što potencijalno mogu rezervisati isti entitet u preklapajućem periodu što bi bio ozbiljan problem, jer se ne može entitet izdavati istovremeno dvoma ili više klijenata(u istom ili preklapajućem periodu).

Rješenje: Problem smo riješili tako što pesimistički zaključamo metodu za dobavljanje entiteta prilikom pravljenja nove rezervacije, tako da u jednom trenutku samo jedan klijent može napraviti rezervaciju.

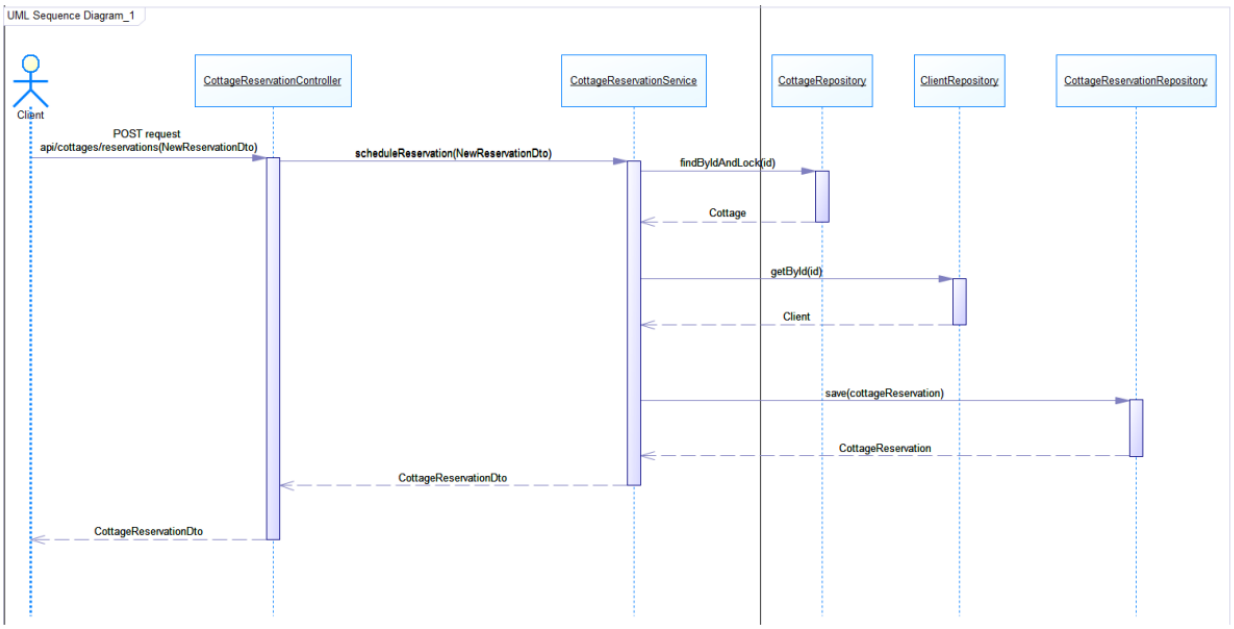
Metoda za pravljenje rezervacije vikendice(metode su identične za avanture i čamce/brodove):

```
@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();
    }
}
```

Metoda za dobavljanje i pesimističko zaključavanje vikendice(metode su identične za avanture i čamce/brodove):

```
@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);
```

Dijagram toka pravljenja nove rezervacije vikendice(metode su identične za rezervisanje preko akcija):



Problem 3: Ne može se istovremeno kreirati više naloga sa istim email-om ili username-om

U situaciji da dva korisnika iniciraju registraciju istovremeno sa istim email-om ili username-om može doći do većeg problema gdje kada se korisnik hoće ulogovati (gdje su potrebni username i password) može nastati problem da se uloguje na pogrešan nalog. Takođe, slanje email-a bi obavještavalo dva korisnika.

Rješenje: Problem smo riješili tako što pesimistički zaključamo metodu za sačuvavanje naloga (u servisu) prilikom registracije, tako da u jednom trenutku samo jedan korisnik može da napravi jedan nalog sa datim email-om ili username-om.

Metoda za registraciju klijenta:

```
@Transactional
public Client save(RegistrationDto registrationDTO) {
    Client c = new Client();
    c.setUsername(registrationDTO.getUsername());
    // pre nego sto postavimo lozinku u atribut hesiramo je
    c.setPassword(passwordEncoder.encode(registrationDTO.getPassword()));
    c.setFirstName(registrationDTO.getFirstName());
    c.setLastName(registrationDTO.getLastName());
    c.setEmail(registrationDTO.getEmail());
    c.setCountry(registrationDTO.getCountry());
    c.setCity(registrationDTO.getCity());
    c.setAddress(registrationDTO.getAddress());
    c.setPhoneNumber(registrationDTO.getPhoneNumber());
    c.setPenaltyCount(0);
    c.setLoyaltyPoints(0);
    c.setEnabled(false);

    List<Authority> auth = authService.findByName("ROLE_CLIENT");
    c.setAuthorities(auth);

    return this.clientRepository.save(c);
}
```

Metoda za dobavljanje i zaključavanje korisnika:

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
User findByUsername(String username);

@Lock(LockModeType.PESSIMISTIC_WRITE)
User findByEmail(String email);
```

Dijagram toka registracije klijenta: