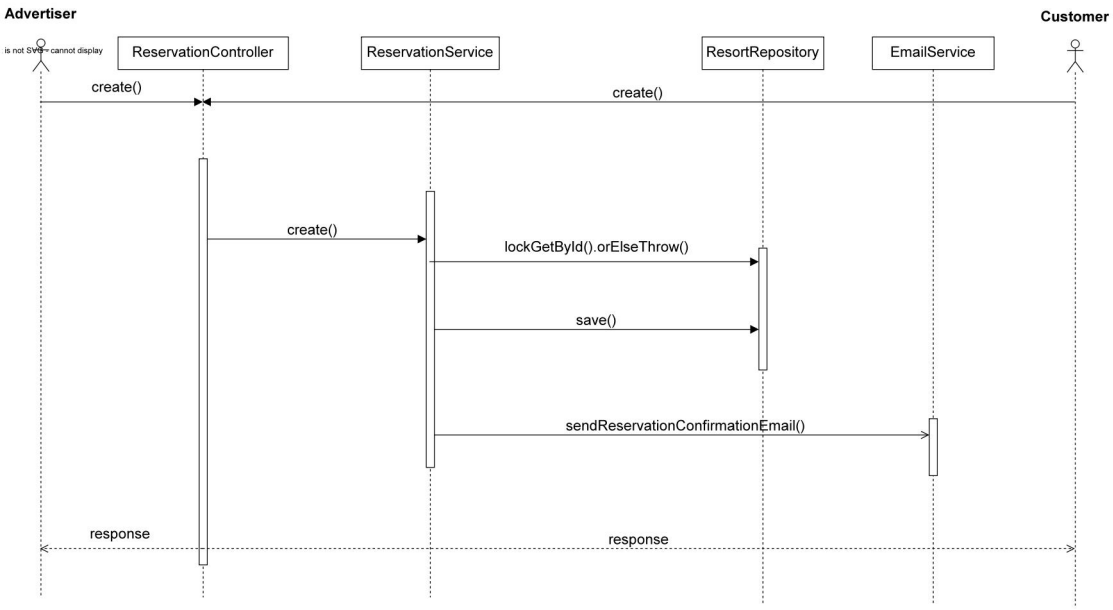
Konkurentni pristup podacima u bazi

Student 2 - Ana Anđelić SW14/2019

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

***Problem*:** Vlasnik vikendice/broda ili instruktor pecanja, u daljem tekstu oglašavač, može u dogovoru sa klijentom za kog trenutno traje rezervacija da kreira novu rezervaciju. Trenutno aktivna rezervacija se mora odnositi na oglas koji je postavio taj konkretan oglašavač. Prilikom kreiranja nove rezervacije bez konkurentnog pristupa se može desiti da se kreiraju dve rezervacije u istom ili preklapajućem vremenskom periodu, što naravno nije dozvoljeno.

***Tok zahteva*:** Na slici 1 je prikazan dijagram sekvence za kreiranje rezervacije za vikendice. Analogno ovom dijagramu, vrlo sličan bi bio dijagram kreiranja rezervacije za brodove i avanture. Jedina razlika je što bi se koristio repozitorijum za brod, odnosno avanturu.



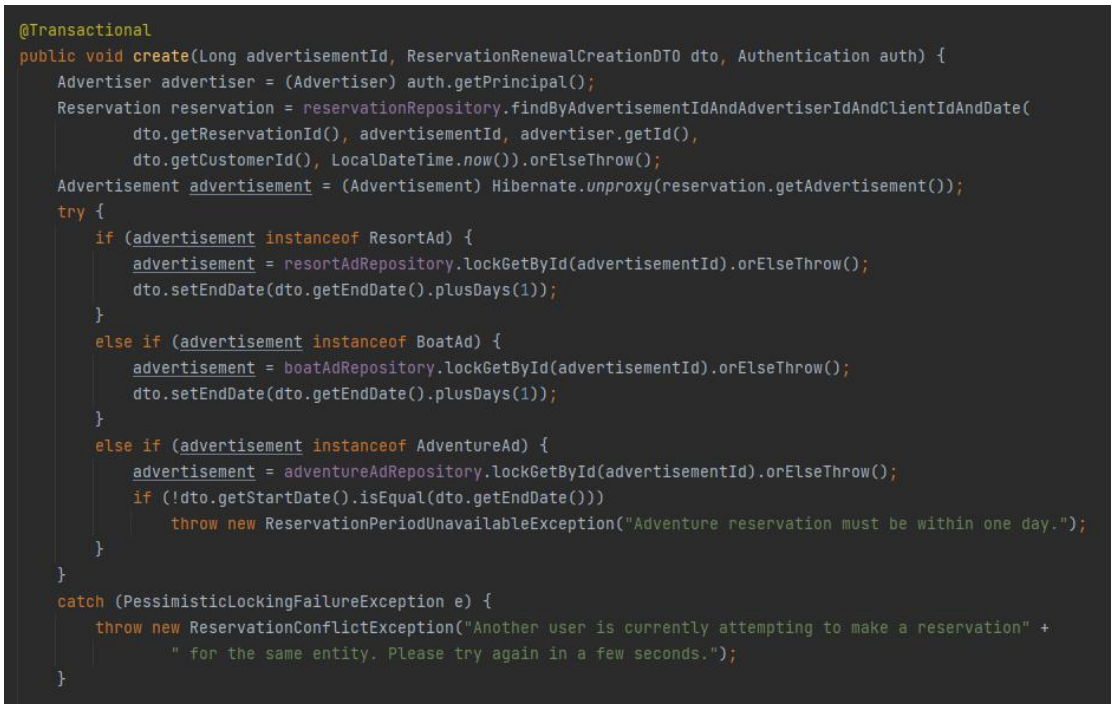
Slika 1 - Dijagram sekvence za kreiranje rezervacije za vikendice

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom obnove rezervacije od strane oglašavača je */reservations/{id}/renewal* kontrolera *ReservationController* i ova metoda je prikazana na slici 1.

```
@PostMapping(value = "/{id}/renewal")
@PreAuthorize("hasRole('ADVERTISER')")
public ResponseOK createRenewedReservation(@PathVariable Long id,
                                           @Valid @RequestBody ReservationRenewalCreationDTO dto,
                                           Authentication auth) {
    reservationService.create(id, dto, auth);
    return new ResponseOK("Reservation renewed.");
}
```

Slika 2 - Metoda kontrolera za prvi konfliktni slučaj

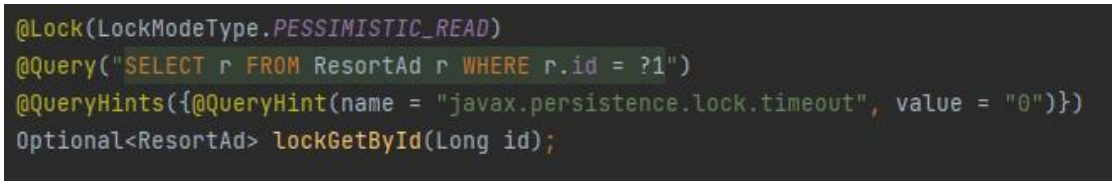
Metoda *createRenewedReservation* ovog kontrolera poziva metodu *create* servisa *ReservationService* i prikazana je na slici 2. Ova metoda je ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje vikendice, broda, odnosno avanture. Oglas ostaje zaključan do kraja metode. Ako u bilo kom trenutku korisnik pokuša da napravi rezervaciju, odnosno pokuša da zaključa isti ovaj resursa, doći će do greške pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen exception *ReservationConflictException* koji će korisniku vratiti odgovor sa porukom da neko drugi u ovom trenutku već pokušava da napravi rezervaciju za ovaj oglas.



```
@Transactional
public void create(Long advertisementId, ReservationRenewalCreationDTO dto, Authentication auth) {
    Advertiser advertiser = (Advertiser) auth.getPrincipal();
    Reservation reservation = reservationRepository.findByAdvertisementIdAndAdvertiserIdAndClientIdAndDate(
            dto.getReservationId(), advertisementId, advertiser.getId(),
            dto.getCustomerId(), LocalDateTime.now()).orElseThrow();
    Advertisement advertisement = (Advertisement) Hibernate.unproxy(reservation.getAdvertisement());
    try {
        if (advertisement instanceof ResortAd) {
            advertisement = resortAdRepository.lockGetById(advertisementId).orElseThrow();
            dto.setEndDate(dto.getEndDate().plusDays(1));
        }
        else if (advertisement instanceof BoatAd) {
            advertisement = boatAdRepository.lockGetById(advertisementId).orElseThrow();
            dto.setEndDate(dto.getEndDate().plusDays(1));
        }
        else if (advertisement instanceof AdventureAd) {
            advertisement = adventureAdRepository.lockGetById(advertisementId).orElseThrow();
            if (!dto.getStartDate().isEqual(dto.getEndDate()))
                throw new ReservationPeriodUnavailableException("Adventure reservation must be within one day.");
        }
    }
    catch (PessimisticLockingFailureException e) {
        throw new ReservationConflictException("Another user is currently attempting to make a reservation" +
                " for the same entity. Please try again in a few seconds.");
    }
```

Slika 3 - Metoda servisa za prvi konfliktni slučaj

Metode *lockGetById* repozitorijuma *ResortAdRepository*, *BoatAdRepository*, *AdventureAdRepository* služe za dobavljanje i zaključavanje vikendice, broda ili avanture i prikazane su redom na slikama 3, 4, 5.



```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT r FROM ResortAd r WHERE r.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<ResortAd> lockGetById(Long id);
```

Slika 4 - Metoda repozitorijuma za zaključavanje vikendice

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT b FROM BoatAd b WHERE b.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<BoatAd> lockGetById(Long id);
```

Slika 5 - Metoda repozitorijuma za zaključavanje broda

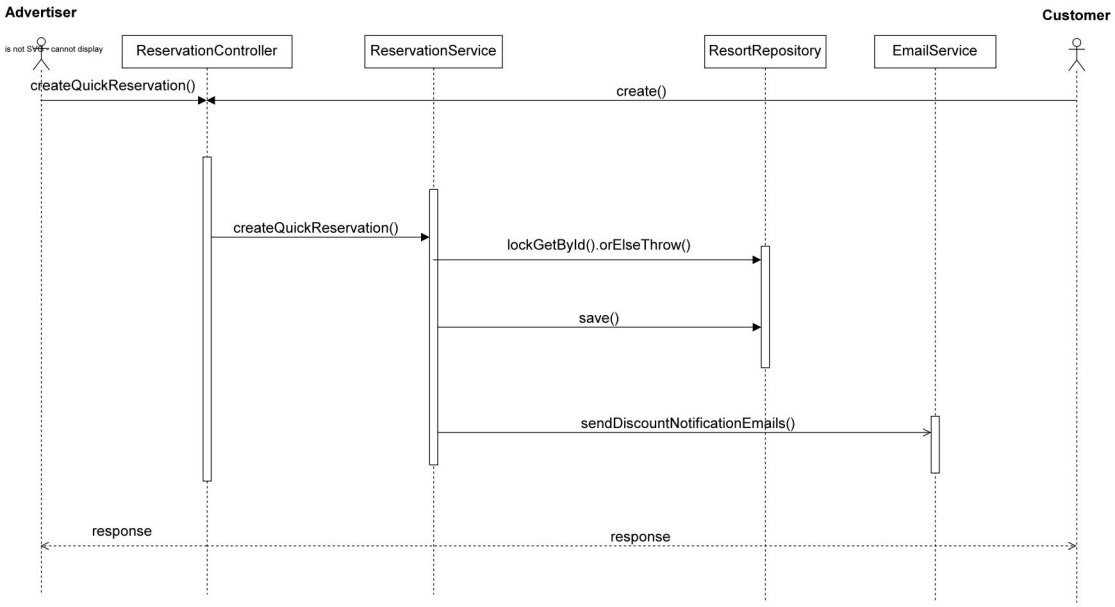
```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT a FROM AdventureAd a WHERE a.id = ?1")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<AdventureAd> lockGetById(Long id);
```

Slika 6 - Metoda repozitorijuma za zaključavanje avanture

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

***Problem*:** Vlasnik vikendice/broda ili instruktor pecanja, u daljem tekstu oglašavač, može da kreira akciju za odabrani oglas. Ovo podrazumeva odabir intervala buduće rezervacije tokom kog ne bi trebalo da postoji bilo kakvo preklapanje sa drugim rezervacijama za isti oglas. Prilikom kreiranja nove rezervacije bez konkurentnog pristupa se može desiti da se kreiraju rezervacija i akcija u istom ili preklapajućem vremenskom periodu, što naravno nije dozvoljeno.

Tok zahteva: Na slici 7 je prikazan dijagram sekvence za kreiranje akcije.



Slika 7 - Dijagram sekvence za kreiranje akcije za vikendice

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom obnove rezervacije od strane oglašavača je */reservations/quick-reservation* kontrolera *ReservationController* i ova metoda je prikazana na slici 8.

```
@PostMapping(value = "/quick-reservation")
@PreAuthorize("hasRole('ADVERTISER')")
public ResponseOK createQuickReservation(@Valid @RequestBody QuickReservationCreationDTO dto,
                                         Authentication auth) {
    reservationService.createQuickReservation(dto, auth);
    return new ResponseOK("Quick reservation created.");
}
```

Slika 8 - Metoda kontrolera za kreiranje brze rezervacije

Metoda *createRenewedReservation* ovog kontrolera poziva metodu *create* servisa *ReservationService* i prikazana je na slici 9. Ova metoda je ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje vikendice, broda, odnosno avanture. Ako je pre toga klijent već pristupio resursu i zaključao ga, doći će do greške pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen exception *ReservationConflictException* koji će korisniku vratiti odgovor sa porukom da neko od klijenata u ovom trenutku već pokušava da napravi rezervaciju za ovaj oglas.

```
@Transactional
public void createQuickReservation(QuickReservationCreationDTO dto, Authentication auth) {
    Advertiser advertiser = (Advertiser) auth.getPrincipal();
    Advertisement advertisement = advertisementRepository
            .findAdvertisementByIdAndAdvertiser(dto.getAdvertisementId(), advertiser).orElseThrow();

    try {
        if (advertisement instanceof ResortAd) {
            advertisement = resortAdRepository.lockGetById(dto.getAdvertisementId()).orElseThrow();
            dto.setEndDate(dto.getEndDate().plusDays(1));
        }
        else if (advertisement instanceof BoatAd) {
            advertisement = boatAdRepository.lockGetById(dto.getAdvertisementId()).orElseThrow();
            dto.setEndDate(dto.getEndDate().plusDays(1));
        }
        else if (advertisement instanceof AdventureAd) {
            advertisement = adventureAdRepository.lockGetById(dto.getAdvertisementId()).orElseThrow();
            if (!dto.getStartDate().isEqual(dto.getEndDate()))
                throw new ReservationPeriodUnavailableException("Adventure reservation must be within one day.");
        }
    }
    catch (PessimisticLockingFailureException e) {
        throw new ReservationConflictException("Customer is currently attempting to make a reservation" +
                " for the same entity. Please try again in a few seconds.");
    }
```

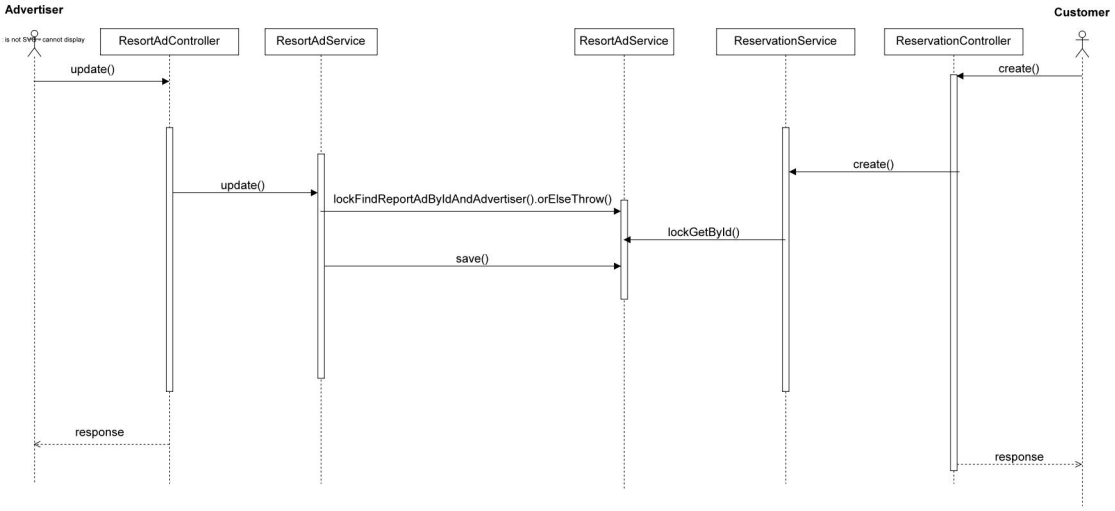
Slika 9 - Metoda servisa za drugi konfliktni slučaj

Metode *lockGetById* repozitorijuma *ResortAdRepository*, *BoatAdRepository*, *AdventureAdRepository* služe za dobavljanje i zaključavanje vikendice, broda ili avanture i prikazane su redom na slikama 3, 4, 5 u prvom poglavlju.

3. Vlasnik vikendice ne može da izmeni vikendicu u isto vreme kad i klijent vrši rezervaciju za istu vikendicu

***Problem*:** Vlasnik vikendice može da vrši izmenu svoje vikendice. Ovo obuhvata između ostalog i izmenu kapaciteta i opcija prilikom rezervacije. Zbog ovoga ne sme da se desi da klijent pokuša da napravi rezervaciju u istom trenutku kada se izvršava izmena.

Tok zahteva: Na slici 10 je prikazan dijagram sekvence za izmenu vikendice.



Slika 10 - Dijagram sekvence za izmenu vikendice

Rešenje: Ovaj problem je rešen pesimističkim zaključavanjem resursa u bazi podataka. Endpoint koji se gađa prilikom obnove rezervacije od strane oglašavača je */reservations/quick-reservation* kontrolera *ReservationController* i ova metoda je prikazana na slici 11.

```
@PutMapping(value = "/{id}", consumes = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasRole('RESORT_OWNER')")
public ResponseOK update(@PathVariable Long id, @Valid @RequestBody ResortAdUpdationDTO dto,
                         Authentication auth) {
    resortAdService.update(id, dto, auth);
    return new ResponseOK("Resort ad updated.");
}
```

Slika 11 - Metoda kontrolera za izmenu vikendice

Metoda update ovog kontrolera poziva metodu *update* servisa *ResortAdService* i prikazana je na slici 12. Ova metoda je ima anotaciju *@Transactional*. Prilikom poziva ove metode vrši se zaključavanje vikendice. Ako je pre toga klijent već pristupio resursu i zaključao ga, doći će do greške pod nazivom *PessimisticLockingFailureException*, koji će biti uhvaćen i umesto njega bačen exception *ReservationConflictException* koji će korisniku vratiti odgovor sa porukom da neko u ovom trenutku pokušava da napravi rezervaciju za ovaj oglas.

```java
@Transactional
public void update(Long id, ResortAdUpdationDTO dto, Authentication auth) {
    Advertiser advertiser = (Advertiser) auth.getPrincipal();
    ResortAd resortAd = resortAdRepository.findResortAdByIdAndAdvertiser(id, advertiser).orElseThrow();
    try {
        resortAd = resortAdRepository.lockFindResortAdByIdAndAdvertiser(id, advertiser).orElseThrow();
    }
    catch (PessimisticLockingFailureException e) {
        throw new ReservationConflictException("Customer is currently attempting to make a reservation" +
                " for this entity. Please try again in a few seconds.");
    }
    modelMapper.map(dto, resortAd);

    resortAd.verifyPhotosOwnership(advertiser);

    tagRepository.saveAll(resortAd.getTags());
    resortAdRepository.save(resortAd);
}
```

Slika 12 - Metoda servisa za treći konfliktni slučaj

Metoda pod nazivom *lockFindResortAdByIdAndAdvertiser* repozitorijuma *ResortAdRepository* služi za dobavljanje i zaključavanje vikendice i prikazana je na slici 13.

```java
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("SELECT r FROM ResortAd r WHERE r.id = ?1 AND r.advertiser = ?2")
@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value = "0")})
Optional<ResortAd> lockFindResortAdByIdAndAdvertiser(Long id, Advertiser advertiser);
```