Konfliktne situacije (student 1)

Po priloženoj specifikaciji zadatak je bio rešiti konfliktne situacije:

- Istovremena rezervacija istog entiteta od strane više klijenata
- Istovremena brza rezervacija istog entiteta od strane više klijenata

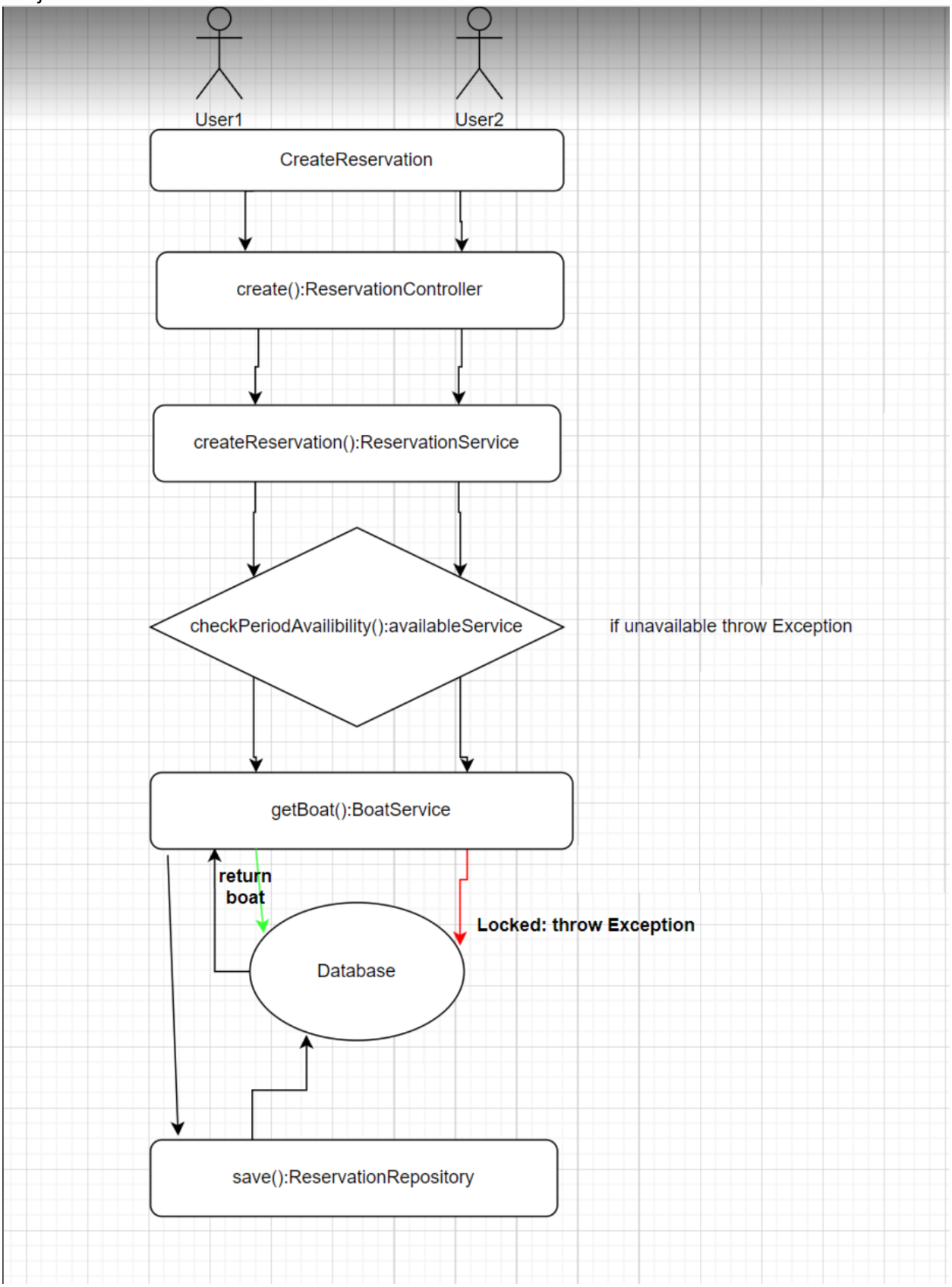
Dodatna uočena i rešena konfliktna situacija:

- Istovremeni pokušaj kreiranja profila klijenta sa istim mejlom od strane više klijenata

1. Istovremena rezervacija istog entiteta od strane više klijenata

Jedna od funkcionalnosti aplikacije je mogućnost klijenta da putem nje u slobodnom terminu kreira rezervaciju prilikom koje dobija potvrdu na mejl o potvdjivanju iste. Pri čemu se može javiti da više klijenata(u našem primeru 2) pokušaju da zakažu istovremeno u istom ili preklapajućem terminu što može bazu i ceo sistem dovesti u nekonzistentno stanje.
Od 2 potencijalna pristupa problemu zaključavanje baze pri dobaljvanju slobodnog termina i zaključavanja baze pri dobavljanju konkretnih entiteta vezanih za rezervaciju (brod,vikendica, instruktor) odabrali smo zaključavanje konretnog entiteta pri pokušaju rezervacije. Rešenje ćemo prezentovati na primeru rezervacije broda dok je za vikendice i instruktore respektivno tome. Dijagram toka odvijanja

akcije:


User1
User2
CreateReservation
create():ReservationController
createReservation():ReservationService
checkPeriodAvailibility():availableService
if unavailable throw Exception
getBoat():BoatService
return boat
Locked: throw Exception
Database
save():ReservationRepository

Početak implementacije rešenja konflikta nalazi se u BoatService-u u kom je definisana transakciona readonly metoda za dobavljanje broda

```
@Transactional(readOnly = true)
public Boat findOneById(Long id){
    Boat boat = boatRepository.findOneById(id);
    if(boat==null) throw new EntityNotFoundException(Boat.class.getSimpleName());
    return boat;
}
```

koja daljom propagacijom poziva zaključanu metodu repozitorijuma pri čijem pozivu prvostigli korisnik1 zauzima resurs i onemogućuje(baca Exception) korisnika2 da njime upravlja sprečavajući dalju koliziju i time dovodjenje sistema u nekozistentno stanje.

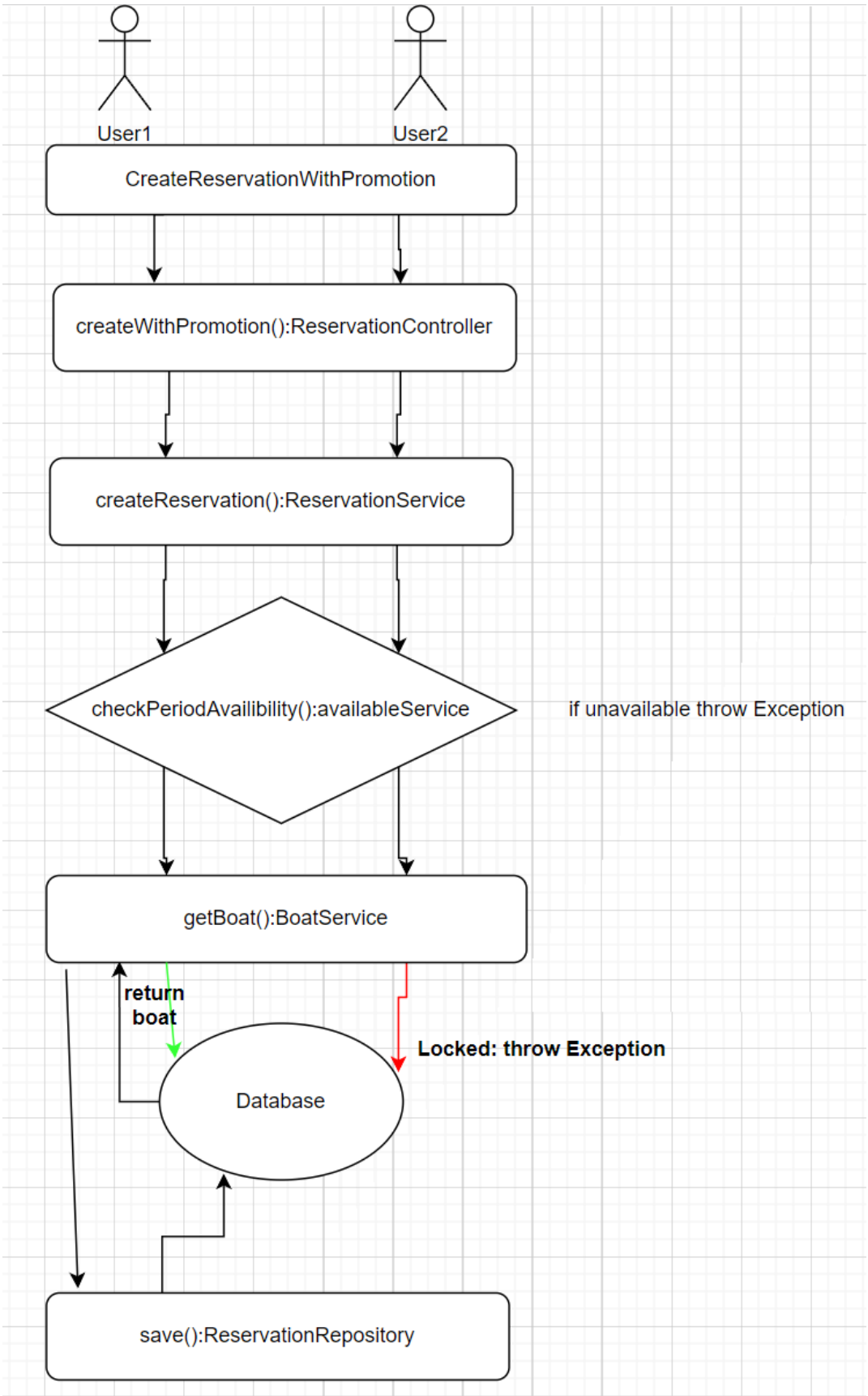
```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("select b from Boat b where b.id=:id")
@QueryHints({@QueryHint(name="javax.persistence.lock.timeout",value="0")})
public Boat findOneById(@Param("id")Long id);
```

Zaključavanje resursa vršimo u pesimističnom modu onemogućavajući klijenta i da čita i da piše zauzeti resurs.
Exception dalje hendlamo u ExceptionHandler klasi vraćajući klijentu HTTP.CONFLICT odgovor.

```
@ExceptionHandler(PessimisticLockingFailureException.class)
public ResponseEntity<ErrorMessage> dataAlreadyLocked(EntityAlreadyExistsException ex, WebRequest request) {
    ErrorMessage message = new ErrorMessage(
            ex.getMessage(),
            new Date());
    return new ResponseEntity<>(message, HttpStatus.CONFLICT);
}
```

2. Istovremena brza rezervacija istog entiteta od strane više klijenata

Jedna od funkcionalnosti aplikacije je mogućnost klijenta da putem nje u već određenim terminima po promotivnoj ceni kreira brzu rezervaciju prilikom koje dobija potvrdu na mejl o potvdjivanju iste. Pri čemu se može javiti da više klijenata(u našem primeru 2) pokušaju da zakažu istu promociju i time dovedu bazu pa I sam sistem u nekonzistentno stanje. Dijagram toka akcija kao I sama akcija su identične gorenavedenoj istovremenoj regularnoj rezervaciji te nećemo dalje širiti nego ćemo priložiti slike ovog puta brze rezervacije vikendice:

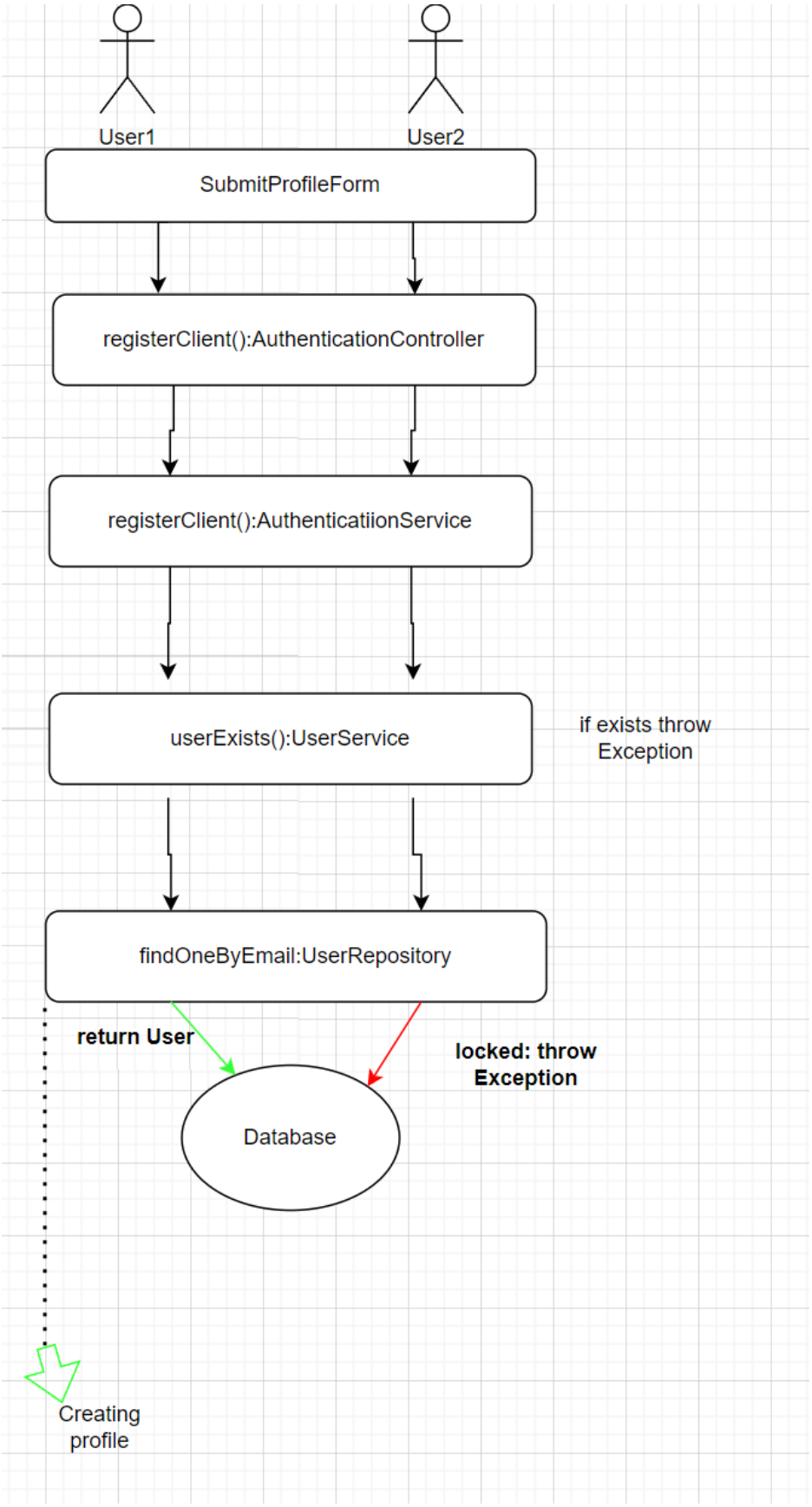

User1
User2
CreateReservationWithPromotion
createWithPromotion():ReservationController
createReservation():ReservationService
checkPeriodAvailibility():availableService
if unavailable throw Exception
getBoat():BoatService
return boat
Locked: throw Exception
Database
save():ReservationRepository

```java
@Transactional(readOnly = true)
public HolidayHome findOneById(Long id){
    HolidayHome home = holidayHomeRepository.findOneById(id);
    if(home == null) throw new EntityNotFoundException(HolidayHome.class.getSimpleName());
    return home;
}
```

```java
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("select h from HolidayHome h where h.id=:id")
@QueryHints({@QueryHint(name="javax.persistence.lock.timeout",value="0")})
public HolidayHome findOneById(@Param("id")Long id);
```

3. Istovremeni pokušaj kreiranja profila klijenta sa istim mejlom od strane više klijenata

Jedna od funkcionalnosti aplikacije je mogućnost neulogovanog korisnika da putem nje kreira profil kako bi mogao da u punom kapacitetu koristi naprednije funkcionalnosti sistema poput rezervacija i promocija. Prijava korisnika na sistem vrši se putem forme u kojoj se unose mejl i dodatni podaci neophodni za korisnički profil sa ograničenjem da mejl na nivou sistema mora biti jedinstven. Pri čemu se može javiti da više klijenata(u našem primeru 2) pokušaju da kreiraju profil sa istim mejlom i time dovedu bazu pa I sam sistem u nekonzistentno stanje. Dati problem rešili smo takođe pesimističnim zaključavanjem ovog puta korisnika koji se dobavlja iz baze pri proveri jedinstvenosti mejla čime smo ostale korisnike u našem slučaju korisnika2 onemogućili da nastavi svoju transakciju I time sprečili dalju koliziju I dovođenje Sistema u nekonzistentno stanje. Dijagram toka


User1
User2
SubmitProfileForm
registerClient():AuthenticationController
registerClient():AuthenticatiionService
userExists():UserService
if exists throw Exception
findOneByEmail:UserRepository
return User
locked: throw Exception
Database
Creating profile

Dijagram toka smo simplifikovali uklanjenjem daljih provera i kreiranjem samog profila te smo ga sveli samo na prikaz konkretnog konflikta i njegovog rešenja. Početak implementacije rešenja nalazi se u transakcionoj metodi AuthenticationService-a registerClient()

```
@Transactional(readOnly = false)
public Client registerClient(Client client) throws UnknownHostException {
    if(providerRegistrationService.registrationExists(client.getEmail()) || userService.userExists(client.getEmail()))
        throw new EntityAlreadyExistsException(client.getEmail());

    return clientRegistrationService.registerClient(client);
}
```

koja se dalje propagira na transakcionu readonly metodu UserService klase userExists()

```
@Transactional(readOnly = true,propagation = Propagation.REQUIRED)
public boolean userExists(String email) {
    return userRepository.findOneByEmail(email) != null;
}
```

te koja dalje poziva metodu UserRepository-a findOneByEmail.

```
@Lock(LockModeType.PESSIMISTIC_READ)
@Query("select b from User b where b.email=:email")
@QueryHints({@QueryHint(name="javax.persistence.lock.timeout",value="0")})
public User findOneByEmail(String email);
}
```