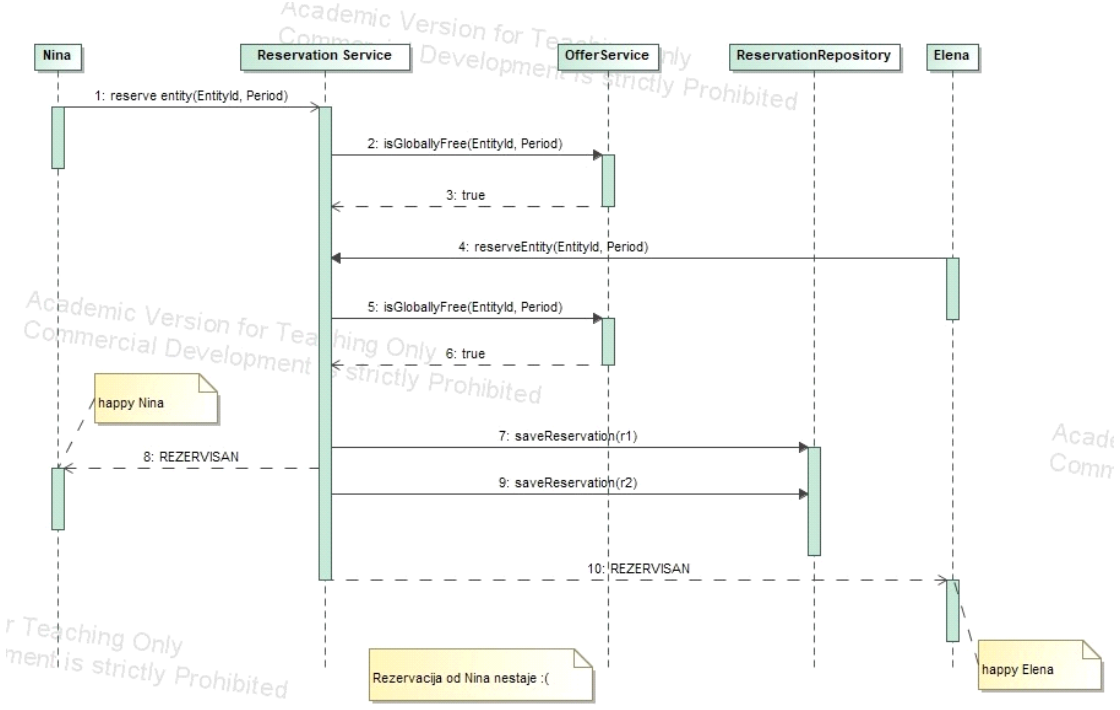
Dokumentacija rešenih konflikata

Napomena: Na slikama su prikazani tokovi interakcije bez detalja koji bi opterećivali sliku.

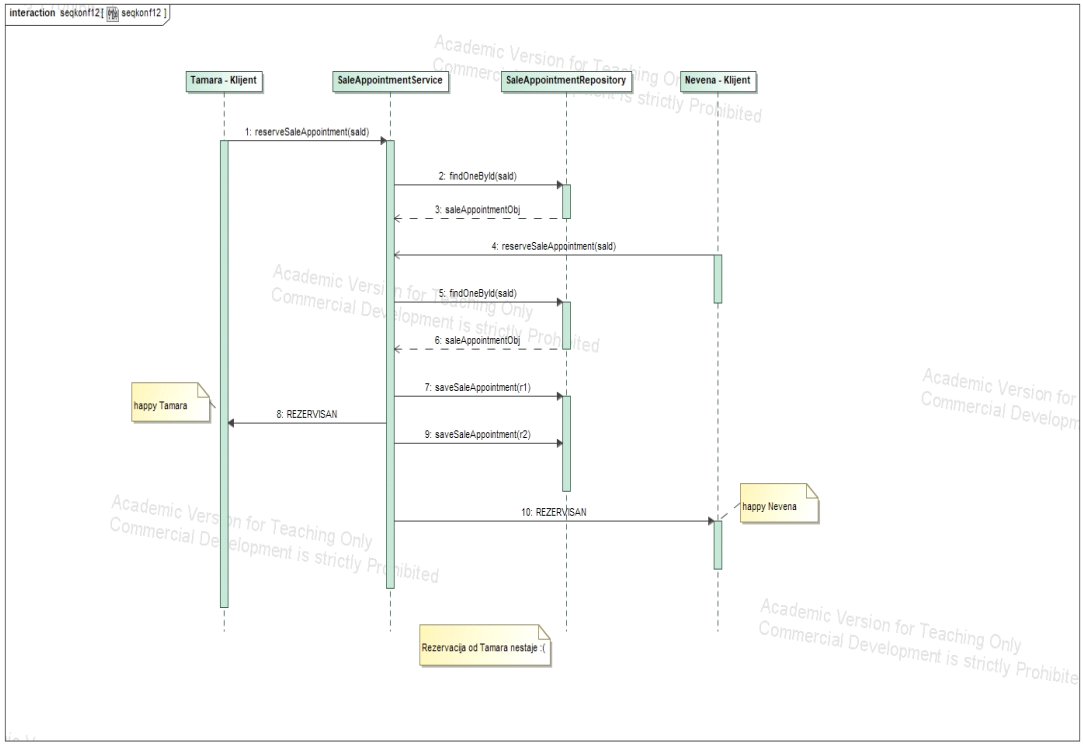
- ## Student 1
  - **Više istovremenih klijenata ne može da napravi rezervaciju istog entiteta u isto (ili preklapajuće) vreme**



Slika Prikaz konfliktne situacije gde Nina i Elena u isto vreme rezervišu isti entitet u preklapajućem vremenu

Za rešavanje konflikta je korišćena promena stepena izolacije tabela da bismo obezbedili konzistentnost podataka tokom cele transakcije i sprečili njihovo menjanje. Zahtev koji prvi stigne će rezervisati entitet. Ova opcija je izabrana jer smatramo da će korisnici češće pretraživati enitete/ponude, nego ih rezervisati. Pristupna tačka koja se gađa je **/offer/reserve**, metoda *public ResponseEntity<ReserveEntityResponseDTO> reserveEntity (@RequestBody ReserveEntityDTO reserveEntityDTO)* u ReservationController-u. Ova metoda poziva metodu ReservationService-a *public Reservation makeReservation (ReserveEntityDTO reserveEntityDTO)* sa anotacijom **@Transactional**, parametar isolation je setovan na **Isolation.REPEATABLE_READ**.

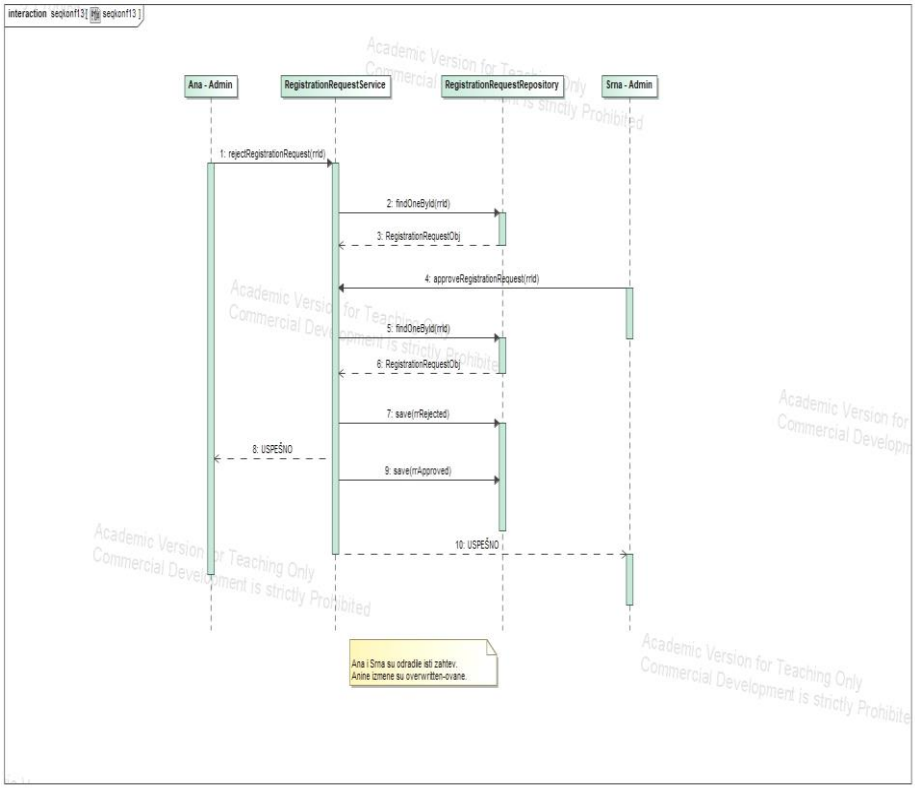
- **Više istovremenih klijenata ne mogu da naprave rezervaciju istog entiteta na akciji u isto vreme**



Slika Prikaz konflikte situacije gde Tamara i Nevena u isto vreme rezervišu istu brzu rezervaciju

Za rešavanje je korišćeno optimističko zaključavanje objekata. Na ovaj način obezbeđujemo da entitet ne bude višestruko rezervisan i da dozvolimo ostalima da čitaju sadržaj tabele. Pristupna tačka koja se gađa je **/sale/quick/reserve**, metoda *public ResponseEntity<SuccessResponseDTO> reserveSaleAppointment (@RequestBody ReserveSaleAppointmentRequestDTO dto)* u SaleAppointmentController-u. Ova metoda poziva metodu SaleAppointmentService-a *public void reserveSaleAppoitment(ReserveSaleAppointmentRequestDTO dto)* sa anotacijama **@Transactional**. U klasu SaleAppointment dodat je atribut version sa anotacijom **@Version**. Prilikom čuvanja je obezbeđena provera vrednosti atributa version u odnosu na onaj koji mi imamo u našem objektu. Ako se atribut ne poklapa onda se baca exception ObjectOptimisticLockingFailureException.

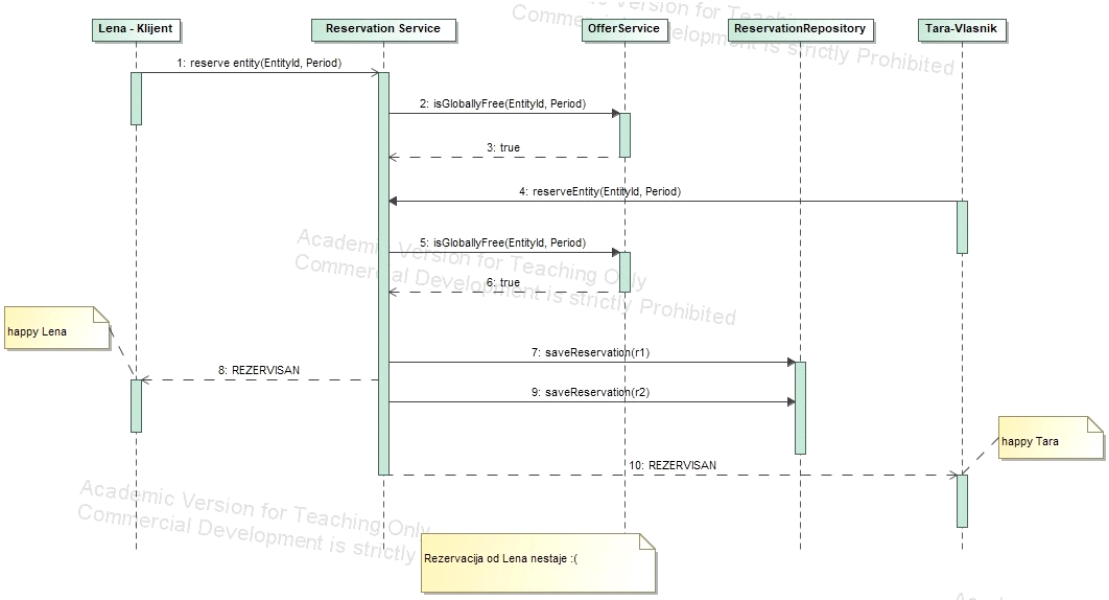
- **na jedan zahtev za registraciju može da odgovori samo jedan administrator sistema**



Slika Prikaz konflikte situacije gde su Ana i Srna obradile isti zahtev

Za rešavanje je korišćeno optimističko zaključavanje objekata. Na ovaj način obezbeđujemo da entitet ne bude višestruko obrađen i dozvoljavamo ostalima da čitaju sadržaj tabele. Pristupne tačke koje se gađaju su **request/registation/approve/{id}** i **request/registation/approve/{id}**, metode su *public ResponseEntity<TextDTO> approveRequest(@PathVariable Long id)* i *public ResponseEntity<TextDTO> rejectRequest(@PathVariable Long id, @RequestBody TextDTO text)* u RegistrationController-u. Ove metode pozivaju metode RegistrationRequestService-a. U klasu RegistrationRequest dodat je atribut version, koji je anotiran anotacijom **@Version** i prilikom čuvanja je obezbeđena provera vrednosti atributa version u odnosu na onaj koji mi imamo u našem objektu. Ako se atribut ne poklapa onda se baca exception ObjectOptimisticLockingFailureException.

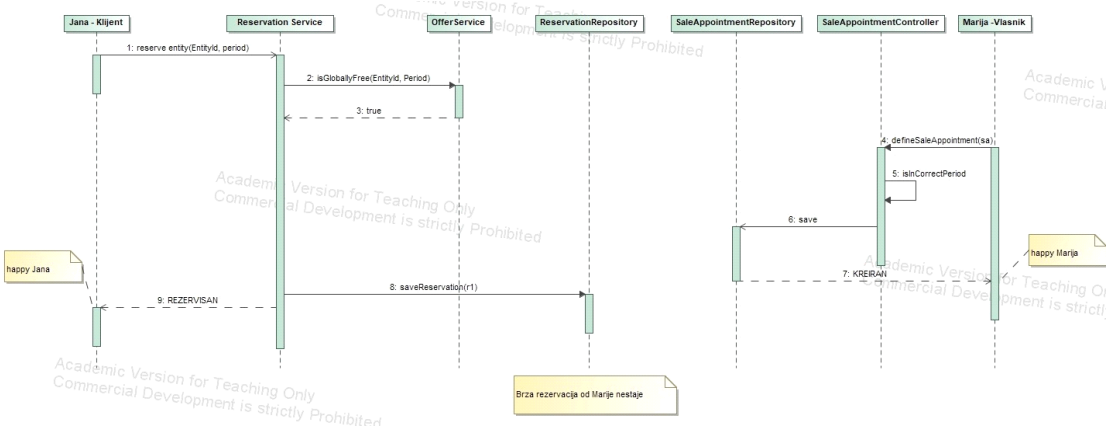
- **Student 2**
  - **vlasnik vikendice/broda ili instruktor ne može da napravi rezervaciju u isto vreme kad i drugi klijent**



Slika Prikaz konflikte situacije gde Lena i Tara u isto vreme rezervišu isti entitet u preklapajućem vremenu

Za rešavanje je korišćena promena stepena izolacije tabela da bismo obezbedili konzistentnost podataka tokom cele transakcije i sprečili njihovo menjanje. Zahtev koji prvi stigne će rezervisati entitet. Ova opcija je izabrana jer smatramo da će korisnici češće pretraživati enitete/ponude nego što će ih rezervisati. Pristupna tačka koja se gađa je **/offer/reserve**, metoda *public ResponseEntity<ReserveEntityResponseDTO> reserveEntity(@RequestBody ReserveEntityDTO reserveEntityDTO)* u ReservationController-u. Ova metoda poziva metodu *ReservationService-a public Reservation makeReservation(ReserveEntityDTO reserveEntityDTO)* sa anotacijom **@Transactional**, parametar isolation je setovan na **Isolation.REPEATABLE_READ**.

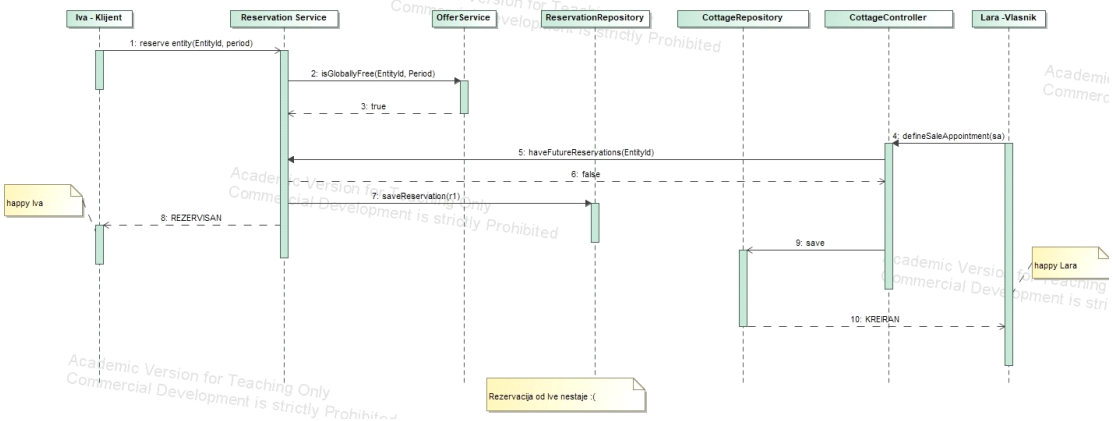
- **vlasnik vikendice/broda ili instruktor ne može da napravi akciju u isto vreme kad i drugi klijent vrši rezervaciju postojećeg entiteta**



Slika Prikaz konflikte situacije, gde Jana u isto vreme rezerviše entitet, kad vlasnik Marija pravi brzu rezervaciju

Za rešavanje je korišćen pristup: prilikom kreiranja brze rezervacije prvo se napravi rezervacija, za koju je id klijenta setovan na null i ako to uspešno prođe napravi se onda i brza rezervacija. Ova opcija je izabrana, jer smatramo da će korisnici češće pretraživati enitete/ponude, nego što će ih rezervisati. Pristupna tačka koja se gađa je **/sale/cottage/define/{id},** metoda *public ResponseEntity<TextDTO> defineSaleAppointmentCottage (@PathVariable Long id, @RequestBody SaleAppointmentDTO saleAppointmentDTO)* u SaleAppointmentController-u. U ovoj metodi se napravi objekat ReserveEntityDTO i prosledi metodi ReservationService-a *public Reservation makeReservation(ReserveEntityDTO reserveEntityDTO)* sa anotacijom **@Transactional**, parametar isolation je setovan na **Isolation.REPEATABLE_READ**. Prvo se poziva ta metoda za čuvanje rezervacije, a onda metoda SaleAppointmentService-a za čuvanje brze rezervacije.

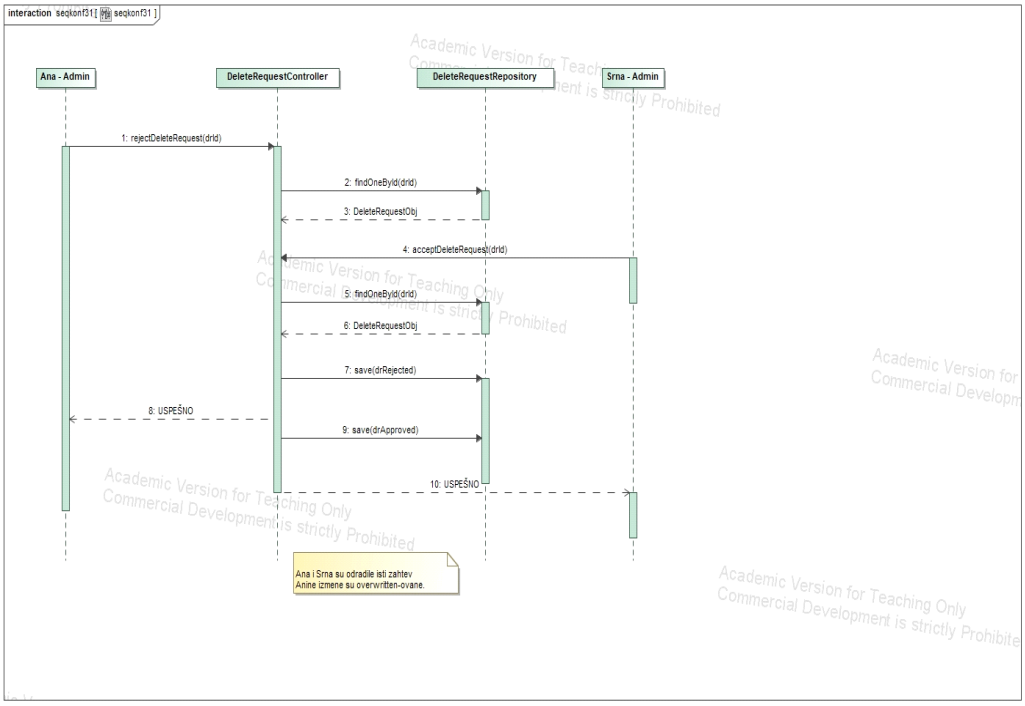
- **vlasnik vikendice/broda ili instruktor ne može da briše entitet u isto vreme kad i drugi klijent rezerviše**



Slika Prikaz konflikte situacije gde Iva u isto vreme rezerviše entitet kad vlasnik Lara briše isti entitet

Za rešavanje je korišćeno pesimističko zaključavanje. Funkcije koje hoće da brišu ili da rezervišu entitet međusobno se lock-uju. Ova opcija je izabrana da bismo sprečili istovremeno rezervisanje entiteta od strane klijenta i brisanje entiteta od strane vlasnika. Prilikom rezervacije gađa se pristupna tačka **/offer/reserve,** metoda *public ResponseEntity<ReserveEntityResponseDTO> reserveEntity(@RequestBody ReserveEntityDTO reserveEntityDTO)* u ReservationController-u. Ova metoda poziva metodu ReservationService-a *public Reservation makeReservation(ReserveEntityDTO reserveEntityDTO)*, u kojoj se poziva metoda *public void reserveEntity(ReserveEntityDTO reserveEntityDTO, Reservation r)*. Ova metoda pokušava da dobije objekat Offer tako što poziva metodu iz CottageService-a *public Cottage findOneTryOccupation(Long id)*, koja poziva metodu CottageRepository-a *Cottage findOneTryOccupation(Long id)* sa anotacijom @**Lock(LockModeType.PESSIMISTIC_WRITE)** i anotacijom **@QueryHints({@QueryHint(name = "javax.persistence.lock.timeout", value ="0")})**

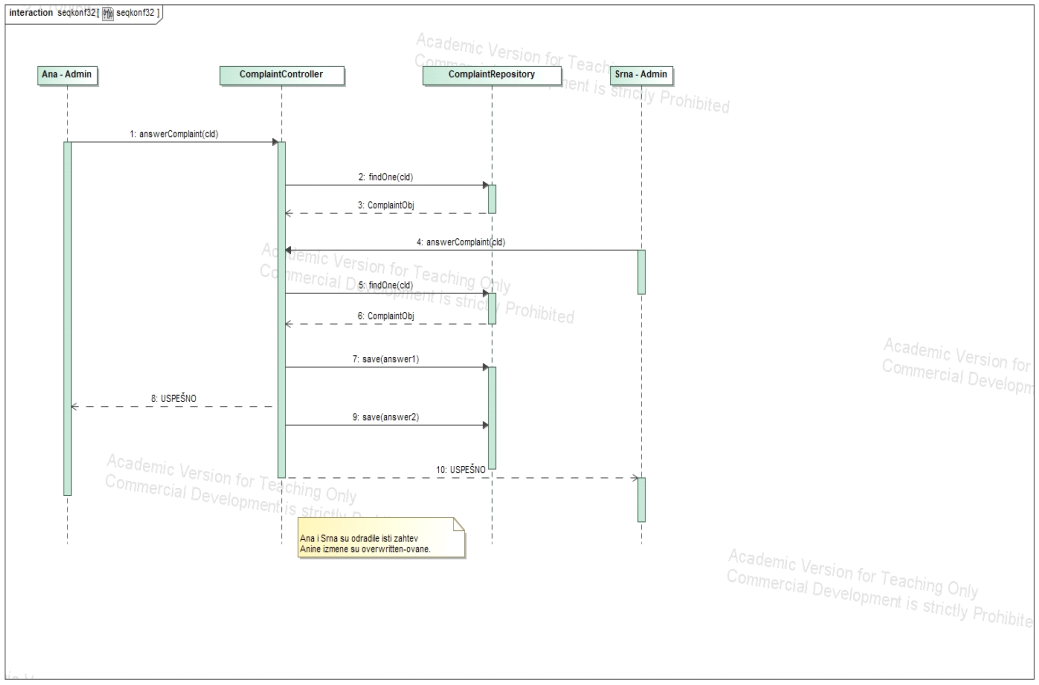
- **Student 3**
  - na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema



Slika Prikaz konflikte situacije gde su Ana i Srna obradile isti zahtev

Za rešavanje je korišćeno optimističko zaključavanje objekata. Na ovaj način obezbeđujemo da entitet ne bude višestruko obrađen i dozvoljavamo ostalima da čitaju sadržaj tabele. Pristupne tačke koje se gađaju su **account/delete/accept/{id}** i **account/delete/reject/{id}**, metode su *public ResponseEntity<TextDTO> acceptDeleteRequest(@PathVariable Long id, @RequestBody TextDTO text)* i *public ResponseEntity<TextDTO> rejectDeleteRequest(@PathVariable Long id, @RequestBody TextDTO text)* u DeleteRequestController-u. Ove metode pozivaju metode DeleteRequestService-a. U klasu DeleteRequest dodat je atribut version, koji je anotiran anotacijom **@Version** i prilikom čuvanja je obezbeđena provera vrednosti atributa version u odnosu na onaj koji mi imamo u našem objektu. Ako se atribut ne poklapa onda se baca exception ObjectOptimisticLockingFailureException.

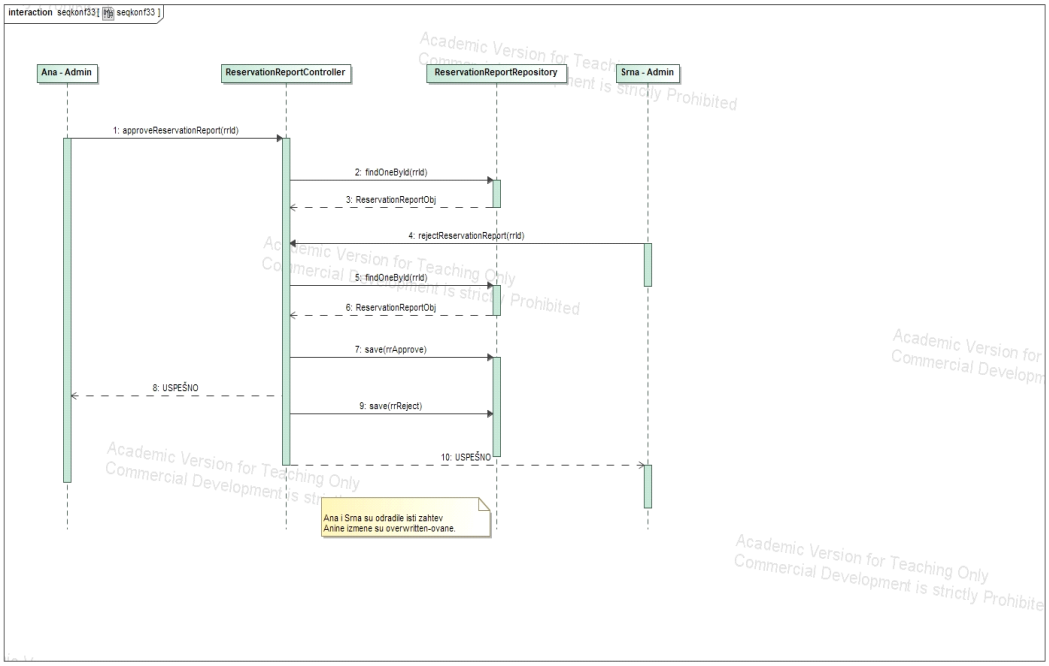
- na jednu žalbu može da odgovori samo jedan administrator sistema



Slika Prikaz konflikte situacije gde su Ana i Srna obradile istu žalbu

Za rešavanje je korišćeno optimističko zaključavanje objekata. Na ovaj način obezbeđujemo da entitet ne bude višestruko obrađen i dozvolimo ostalima da čitaju sadržaj tabele. Pristupna tačka koja se gađa je **complaint/answer/{id}**, metoda je *public ResponseEntity<TextDTO> answerComplaint(@PathVariable Long id, @RequestBody TextDTO text)* u ComplaintController-u. Ova metoda poziva metode ComplaintService-a. U klasu Complaint dodat je atribut version, koji je anotiran anotacijom **@Version** i prilikom čuvanja je obezbeđena provera vrednosti atributa version u odnosu na onaj koji mi imamo u našem objektu. Ako se atribut ne poklapa onda se baca exception ObjectOptimisticLockingFailureException.

- na jedan izveštaj o rezervaciji može da odgovori samo jedan administrator sistema



Slika Prikaz konflikte situacije, gde su Ana i Srna obradile istu reviziju

Za rešavanje je korišćeno optimističko zaključavanje objekata. Na ovaj način obezbeđujemo da entitet ne bude višestruko obrađen i dozvoljavamo ostalima da čitaju sadržaj tabele. Pristupne tačke koje se gađaju su **reservation/report/approve** i **reservation/report/reject**, metode su *public ResponseEntity<TextDTO> approveReservationReport(@RequestBody ReservationReportAdminDTO dto)* i *public ResponseEntity<TextDTO> rejectReservationReport(@RequestBody ReservationReportAdminDTO dto)* u ReservationReportController-u. Ove metode pozivaju metode ReservationReportService-a. U klasu ReservationReport dodat je atribut version, koji je anotiran anotacijom **@Version** i prilikom čuvanja je obezbeđena provera vrednosti atributa version u odnosu na onaj koji mi imamo u našem objektu. Ako se atribut ne poklapa onda se baca exception ObjectOptimisticLockingFailureException.