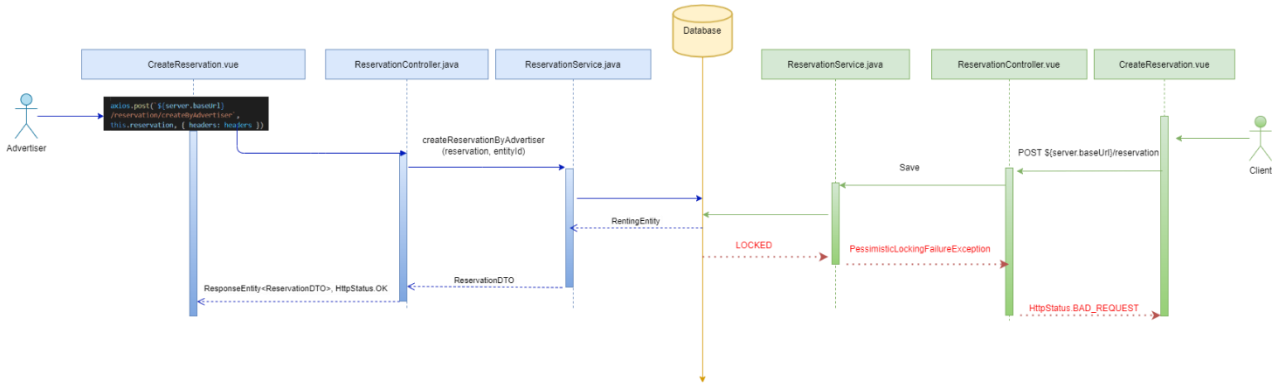
4.4 Konkurentni pristup resursima u bazi

Konfliktna situacija 1: Vlasnik vikendice/broda ili instruktor kreira rezervaciju u isto vrijeme kada i drugi klijent

Opis problema: Pored klijenta, vikendicu/brod ili avanturu može da rezerviše i vlasnik/instruktor za klijenta čija je rezervacija trenutno aktivna. Prilikom rezervacije, potebno je obezbjediti da klijent i vlasnik ne mogu u istom trenutku da izvrše rezervaciju istog entiteta, jer bi se mogla desiti situacija u kojoj se termini kreiranih rezervacija preklapaju.



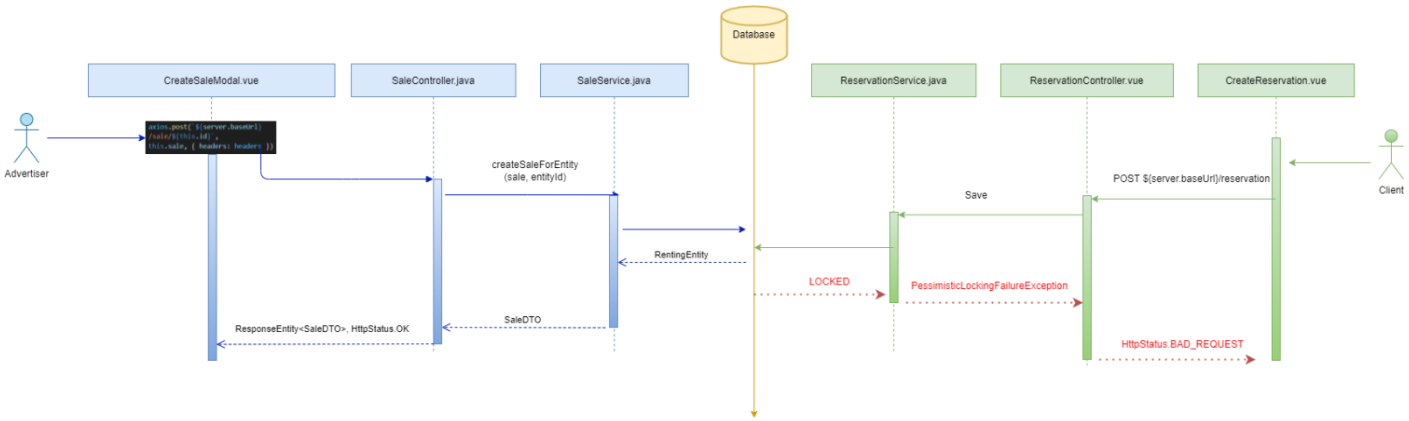
Slika 1: Konfliktna situacija 1 – dijagram sekvenci

Prijedlog rješenja: Rješenje konfliktne situacije nalazi se u klasama *ReservationController, ReservationService* i *EntityRepository*. U klasi *ReservationController*, odrađeno je rukovanje izuzetkom, kako bi klijent i vlasnik/instuktor bili obavješteni o (ne)uspješnosti kreirane rezervacije. Metode *Save(Reservation reservation)* i *saveReservationCreatedByAdvertiser (Reservation reservation, Integer entityId)* su označene kao transakcione i one pozivaju metodu *findLockedById(Integer id)* repozitorijuma *IEntityRepository*, u kome je izvršeno pesimističko zaključavanje navedene metode, ali samo na nivou jednog reda u tabeli koji predstavlja entitet koji se rezerviše. Kao tip zaključavanja, korišten je PESSIMISTIC_WRITE, čime je onemogućeno čitanje zaključanog reda.

Na ovaj način je riješena konfliktna situacija tako što je onemogućeno istovremeno zakazivanje istog entiteta. Loša strana ovakvog pristupa jeste to što će istovremeno zakazivanje istog entiteta biti onemogućeno čak i onda kada se termini rezervacija ne preklapaju i ne mogu dovesti do konflikta.

Konfliktna situacija 2: Vlasnik vikendice/broda ili instruktor kreira akciju u isto vrijeme kada i drugi klijent vrši rezervaciju postojećeg entiteta

Opis problema: Vlasnik vikendice/broda ili instruktor ima mogućnost kreiranja brze rezervacije/akcije, koju kasnije klijent rezerviše jednim klikom. Prilikom kreiranja akcije, može se desiti u istom trenutku klijent vrši standardnu rezervaciju istog entiteta, što bi značilo da bi se mogla desiti situacija u kojoj se termini kreiranih rezervacija preklapaju.



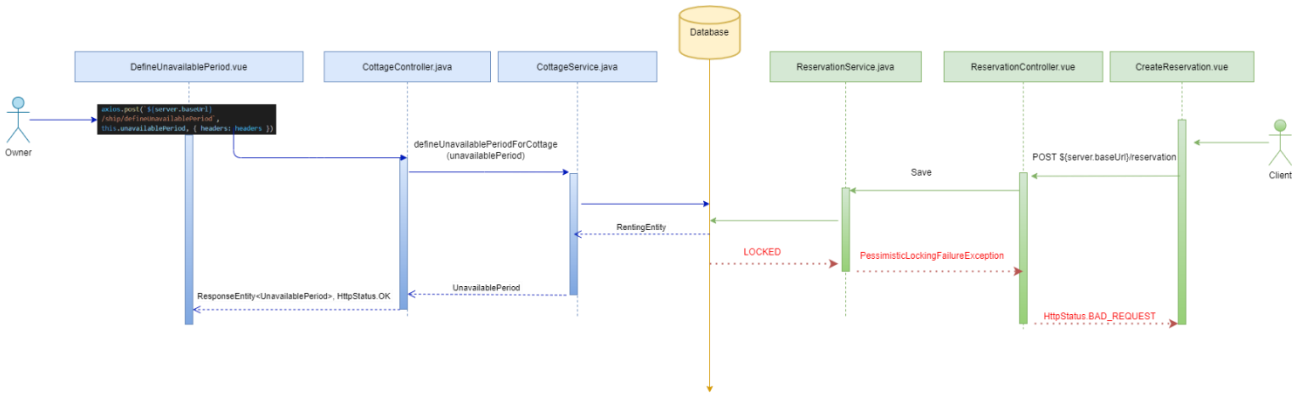
Slika 2: Konfliktna situacija 2 – dijagram sekvenci

Prijedlog rješenja: Rješenje konfliktne situacije nalazi se u klasama *ReservationController, SaleController, ReservationService, SaleService* i *EntityRepository*. U klasama *ReservationController* i *SaleController*, odrađeno je rukovanje izuzecima, kako bi klijent i vlasnik/instuktor bili obavješteni o (ne)uspješnosti kreirane rezervacije/akcije. Metode *Save(Reservation reservation)* u *ReservationService* i *createSaleForEntity(Sale sale, Integer entityId)* u *SaleService* su označene kao transakcione i one pozivaju metodu *findLockedById(Integer id)* repozitorijuma *IEntityRepository*, u kome je izvršeno pesimističko zaključavanje navedene metode, ali samo na nivou jednog reda u tabeli koji predstavlja entitet koji se rezerviše. Kao tip zaključavanja, korišten je PESSIMISTIC_WRITE, čime je onemogućeno čitanje zaključanog reda.

Analogno prehodnom primjeru, konfliktna situacija je riješena tako što je onemogućeno istovremeno zakazivanje istog entiteta. Loša strana ovakvog pristupa jeste to što će istovremeno zakazivanje istog entiteta biti onemogućeno čak i onda kada se termini rezervacija ne preklapaju i ne mogu dovesti do konflikta.

Konfliktna situacija 3: Vlasnik vikendice/broda definiše period nedostupnosti vikendice/broda u isto vrijeme kada i drugi klijent vrši rezervaciju istog entiteta

Opis problema: Vlasnik vikendice/broda ima mogućnost definisanja perioda nedostupnosti tokom kojih nisu moguće rezervacije. Prilikom kreiranja ovih perioda, može se desiti da u istom trenutku klijent vrši rezervaciju istog entiteta, što bi značilo da bi se mogla desiti situacija u kojoj se termin rezervacije preklapa sa period nedostupnosti koji je upravo definisan.



Slika 3: Konfliktna situacija 3 – dijagram sekvenci

Prijedlog rješenja: Rješenje konfliktne situacije nalazi se u klasama *ReservationController, CottageController, ShipController, ReservationService, CottageService, ShipService* i *EntityRepository*. U klasama *ReservationController, CottageController* i *ShipController*, odrađeno je rukovanje izuzecima, kako bi klijent i vlasnik bili obavješteni o (ne)uspješnosti kreirane rezervacije, odnosno perioda nedostupnosti. Metode *Save(Reservation reservation)* u *ReservationService, kao i defineUnavailablePeriodForCottage(UnavailablePeriodDTO, dto)* u *CottageService,* odnosno *defineUnavailablePeriodForShip(UnavailablePeriodDTO, dto)* u *ShipService* su označene kao transakcione i one pozivaju metodu *findLockedById(Integer id)* repozitorijuma *IEntityRepository*, u kome je izvršeno pesimističko zaključavanje navedene metode, ali samo na nivou jednog reda u tabeli koji predstavlja entitet koji se rezerviše. Kao tip zaključavanja, korišten je PESSIMISTIC_WRITE, čime je onemogućeno čitanje zaključanog reda.

Analogno prehodnim primjerima, konfliktna situacija je riješena tako što je onemogućeno istovremeno rezervisanje i definisanje perioda nedostupnosti istog entiteta. Loša strana ovakvog pristupa jeste to što će ove operacije biti onemogućene istovremeno, nad istim entitetom, čak i onda kada se njihovi termini ne preklapaju i ne mogu dovesti do konflikta.