Rešavanje problema konkurentnog pristupa resursima u bazi podataka

1./2. Problem kreiranja rezervacije/rezervacije na akciji od strane vise istovremenih klijenata na istom entitetu u isto/preklapajuce vreme

Na pocetku procesa kreiranja obicne rezervacije klijent ima mogucnost da unese parametre pretrage medju kojima su za ovaj slucaj relevantni pocetni datum rezervacije I krajnji datum rezervacije.Klijentu ce se izlistati entiteti koji su slobodni u navedenom periodu, medjutim moze se desiti da vise klijenata uputi ovakav zahtev I dobije odgovor da su isti entiteti slobodni (jer nijedan od njih jos uvek nista nije rezervisao) pa moze nastati problem istovremene rezervacije na istom entitetu u isto/preklapajuce vreme.U samoj implementaciji rezervacija izvrsena je provera da li vec takva rezervacija postoji (metoda **isAlreadyReserved**), medjutim to nece pokriti slucaj kada 2 ili vise klijenata pokusa istovremeno da rezervise.

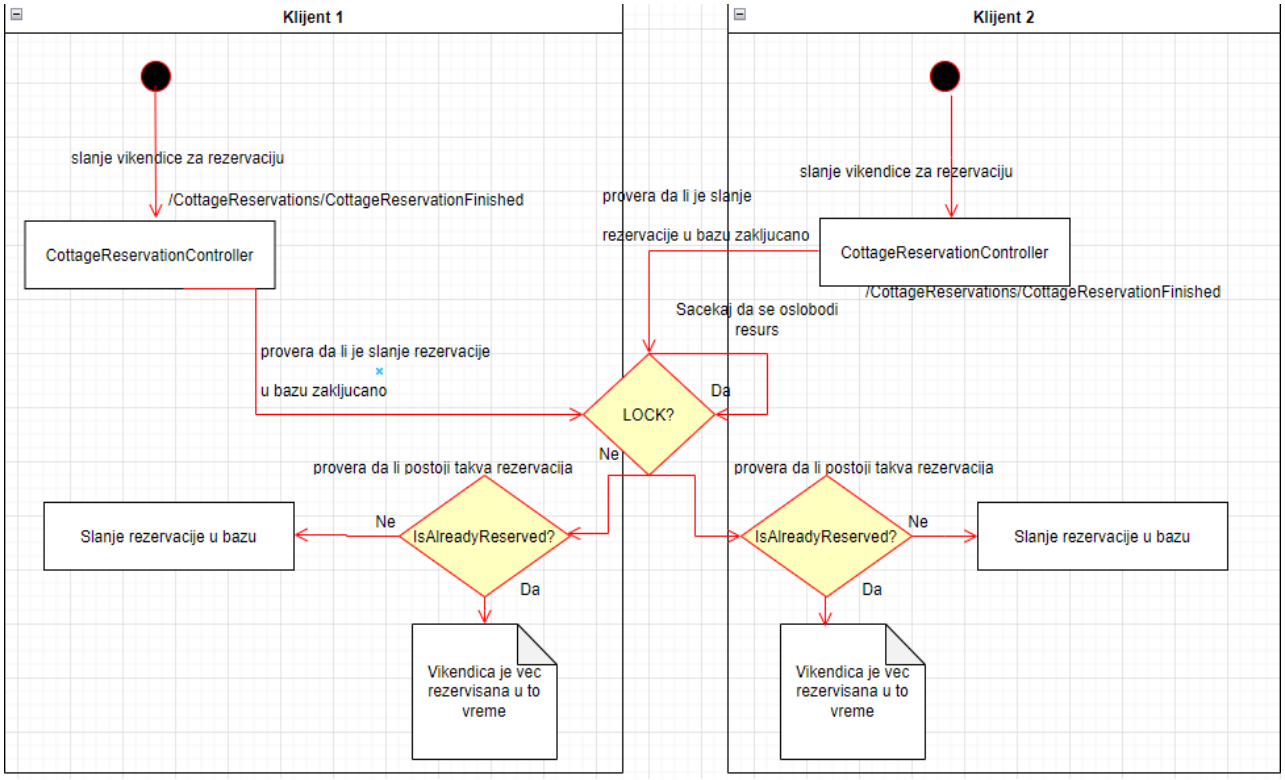
Ovaj problem bi se mogao resiti pesimistickim zakljucavanjem, medjutim ASP.NET s alatom Entity Framework ne podrzava pesimisticku konkurentnost, stoga ovaj problem je resen zakjucavanjem resursa.Nakon sto se rezervacija popuni odgovarajucim podacima, resurs se zakljucava I samo jedan klijent koji je bio najbrzi uci ce u deo metode gde se rezervacija salje u bazu podataka.Pre samog slanja rezervacije u bazu proverava se da li takva rezervacija vec postoji I ako postoji korisnik ce se obavestiti da je entitet vec rezervisan u to vreme.Dakle, prvi klijent ce dobiti mogucnost da upise rezervaciju u bazu podataka, zatim ce sledeci klijent koji je na cekanju kada dobije pristup delu metode koji se odnosi na slanje rezervacije u bazu proveriti da li u to I to vreme postoji rezervacija na tom I tom entitetu I ako da, ukoliko je scenario bio da su oba korisnika pokusala da rezervisu isti entitet, onaj kasniji ce dobiti poruku o gresci.
Gore navedeni mehanizam implementiram je metodom **lock** koju ASP.NET nudi a metode za rezervacije nalaze se u
CottageReservationsController/BoatReservationsController/AdventureReservati

onsController.

```
lock (LockObjectState)
{
    if (isAlreadyReserved(cottageReservation))
    {
        return RedirectToAction("CottageAlreadyReserved", "Home");
    }
    _context.Add(cottageReservation);
    _context.SaveChanges();
}
```

Sto se rezervacija na akciji tice, sve gore navedeno vazi I za njih. Ukoliko izuzmemo pretragu po datumima radi pronalazenja slobodnih entiteta (brze rezervacije imaju unapred definisane datume pocetka/kraja) sve ostalo je implementirano u istim kontrolerima/metodama.

Posto su implementacije identicne za svaki od entiteta, na dijagramu ce biti prikazan primer rezervacija vikendice.



3. Problem pretplate klijenta na odredjeni entitet ukoliko je u isto vreme entitet obrisan od strane administratora/vlasnika

Korisnik se moze pretplatiti na odredjeni entitet ulaskom na listu entiteta I klikom na link *Pretplati Se*.Konfliktna situacija nastaje ukoliko se korisnik pretplati na entitet koji u isto vreme biva obrisan od strane administratora/ vlasnika entiteta, korisnik bi prakticno bio pretplacen na nepostojeci entitet.
Ovaj problem je resen podizanjem **DbUpdateConcurrecyException-a** koji dolazi s okruzenjem ASP.NET I nalazi se u metodi **Create** u kontrolerima **CottageFavoritesController,BoatFavoritesController I AdventureFavoritesController**.
Ovaj exception ce ukoliko se desi brisanje entiteta na koji klijent pokusava da se pretplati dati do znanja klijentu da pretplata nije moguca jer je entitet obrisan.
Implementacija je identicna za sva 3 entiteta te ce na dijagramu biti prikazan slucaj pretplate na vikendicu.

```
try
{
    _context.Update(ctg);
    await _context.SaveChangesAsync();
    _context.Add(cottageFavorites);
    await _context.SaveChangesAsync();
}
catch (DbUpdateConcurrencyException)
{
    return RedirectToAction("ConcurrencyError", "Home");
}
```