Konkurentan pristup resursima u bazi

Konflikt 1

Na jedan zahtev za brisanje naloga može da odgovori samo jedan administrator sistema

Do konfliktne situacije dolazi kada dva administratora u isto ili preklapajuće vreme pokušaju da odgovore na isti zahtev za brisanje. Zbog logike koja je implementirana ukoliko dođe do odobravanja brisanja sa jedne, a odbijanja sa druge strane, korisnik će dobiti dva obaveštenja i sam zahtev će biti i prihvaćen i odbijen čime se narušava konzistentnost sistema.

Solucija:

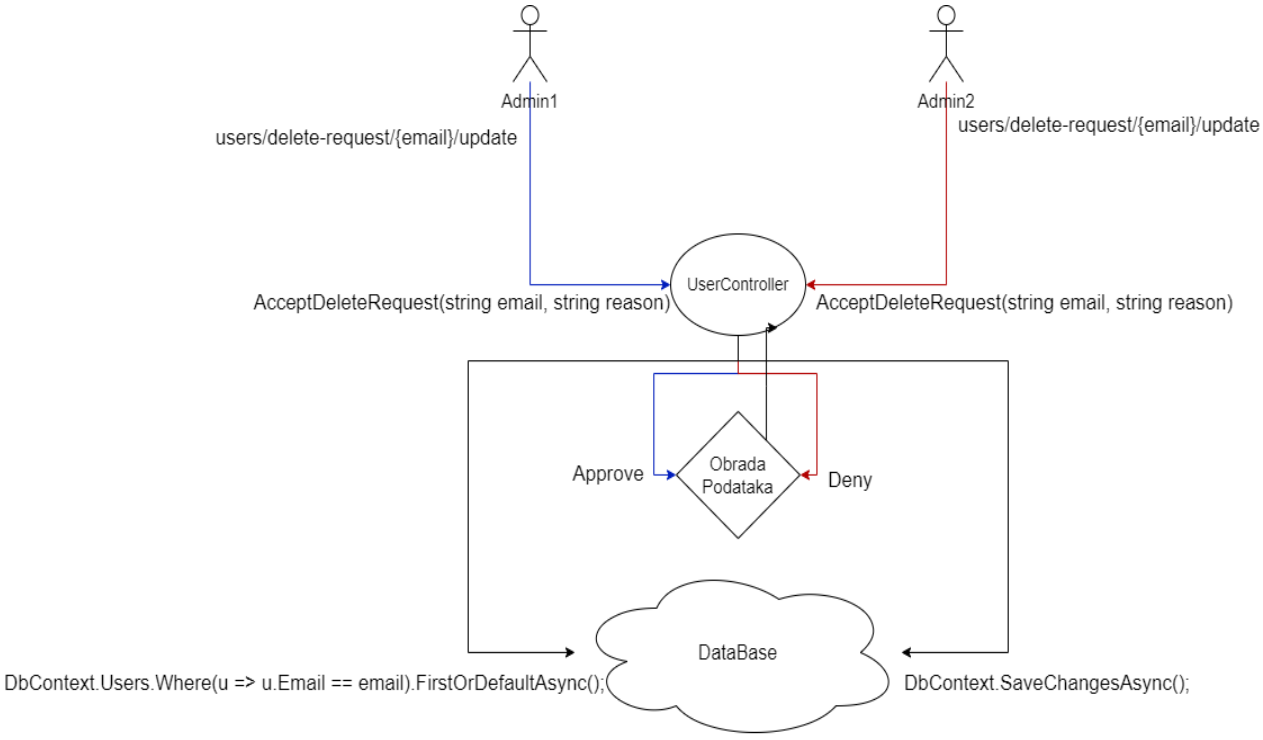
Problem rešavamo postavljanjem pesimističkog zaključavanja na metodu za brisanje naloga, sa nivoom izolacije *Serializable.* Prilikom pristupa svakom zahtevu za brisanje vrši se zaključavanje. Pesimističkim zaključavanjem, dok traje promena entiteta, tj dok traje transakcija, onemogućavaju se promene entiteta koje potiču van transakcije. Entitet se oslobađa tek nakon završetka transakcije. Ukoliko prilikom izmena nad zahtevom neki drugi admin pokuša da mu pristupi sa namerom da isti takođe izmeni, dobiće poruku o grešci.

```
// Conflicting Situation 3.1
[Authorize(Roles = Role.Admin)]
[HttpPost("delete-requests/{email}/update")]
0 references
public async Task<ActionResult> AcceptDeleteRequest([FromRoute] string email, [FromBody] string reason)
{
    await using var transaction = await _context.Database.BeginTransactionAsync(IsolationLevel.Serializable);
    try
    {
        var user = await _context.Users.Where(u => u.Email == email).FirstOrDefaultAsync();
```

```
bool success = await _context.SaveChangesAsync() > 0;
await transaction.CommitAsync();
```

```
        return Ok();
    }
    catch (Exception)
    {
        await transaction.RollbackAsync();
        return BadRequest("Could not update the request at this time. Please try again later.");
    }
}
```

Tok zahteva



Ilustracija 1- Zahtev za brisanje

Konflikt 2

Na jednu žalbu može da odgovori samo jedan administrator sistema

Do konfliktne situacije dolazi kada dva administratora u isto ili preklapajuće vreme pokušaju da odgovore na istu žalbu. Mogući ishod je da u slučaju aplikacije žalba bude dva puta označena kao obrađena, da se podnosiocu žalbe i osobi za koga se žalba odnosi dostave dva različita odgovora i da u jednom slučaju korisnik bude penalizovan a u drugom ne što narušava konzinstentnost sistema.

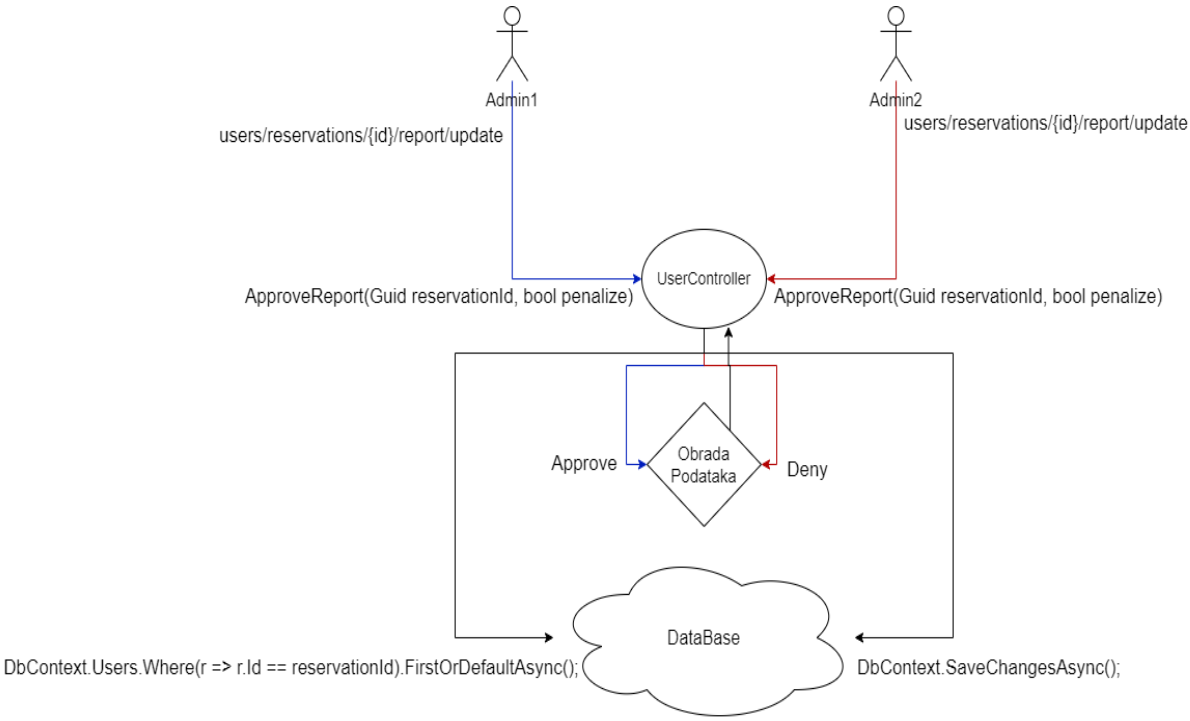
Solucija:

Problem rešavamo postavljanjem pesimističkog zaključavanja na metodu za obradu zahteva o žalbi, sa nivoom izolacije *Serializable.* Time će samo jedan administrator moći da pristupi zahtevu, dok će drugi dobiti informaciju da su izmene entiteta u toku i moraće da pokušaju ponovo. Pesimističnim zaključavanjem osiguravamo da dva admina ne mogu istovremeno da ažuriraru podatke o podnetoj žalbi.

```
// Conflicting Situation 3.2
[Authorize(Roles = Role.Admin)]
[HttpPost("reservations/{reservationId}/report/update")]
0 references
public async Task<ActionResult> ApproveReport([FromRoute] Guid reservationId, [FromQuery] bool penalize)
{
    await using var transaction = await _context.Database.BeginTransactionAsync(IsolationLevel.Serializable);

    try
    {
        await _context.SaveChangesAsync();
        await transaction.CommitAsync();
        return Ok();
    }
    catch (Exception)
    {
        await transaction.RollbackAsync();
        return BadRequest("Could not update the report at this time. Please try again later.");
    }
}
```

Tok zahteva



Ilustracija 2- Obrada zahteva žalbe

Konflikt 3

Jednu ocenu može da odobri samo jedan administrator sistema

Ovaj konflikt pokriva situaciju kada dva ili više administratora u istom ili preklapajućem vremenu obrade reviziju ocene vezane za neku rezervaciju. Paralelna obrada jednog entiteta ocene stvara konflikt u sistemu. Samo jedan od admina koji pokušavaju da prihvate ili odbiju ocenu mogu to i da ostvare.

Solucija:

Problem rešavamo postavljanjem pesimističkog zaključavanja na metodu za obradu zahteva o oceni entiteta, sa nivoom izolacije *Serializable.* Time će samo jedan administrator moći da pristupi oceni, dok će ostali dobiti informaciju da su izmene entiteta u toku i moraće da pokušaju ponovo. Pesimističnim zaključavanjem osiguravamo da dva admina ne mogu istovremeno da ažuriraru podatke o podnetoj oceni entiteta.

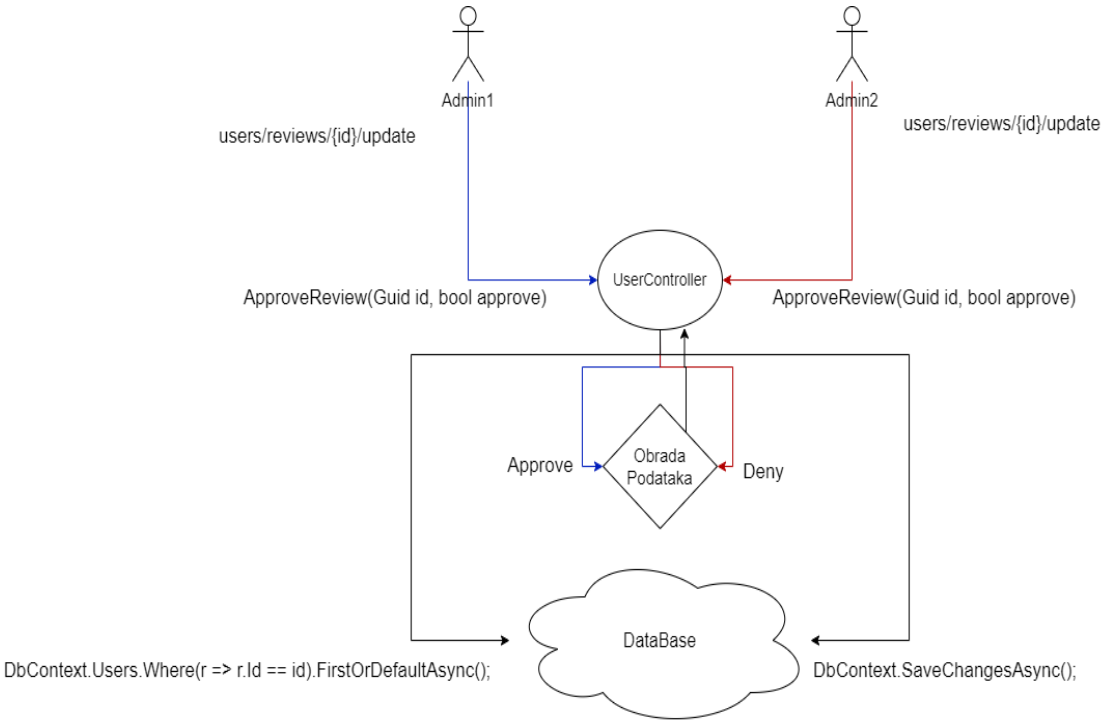
```
// Conflicting Situation 3.3
[Authorize(Roles = Role.Admin)]
[HttpPost("reviews/{id}/update")]
0 references
public async Task<ActionResult> ApproveReview([FromRoute] Guid id, [FromQuery] bool approve)
{
    await using var transaction = await _context.Database.BeginTransactionAsync(IsolationLevel.Serializable);

    try
    {
```

```
await _context.SaveChangesAsync();
await transaction.CommitAsync();
```

```
    }
    catch (Exception)
    {
        await transaction.RollbackAsync();
        return BadRequest("Could not update the review at this time. Please try again later.");
    }
}
```

Tok zahteva



Ilustracija 3- Odobravanje ocene