# Konfliktne situacije - Student 2

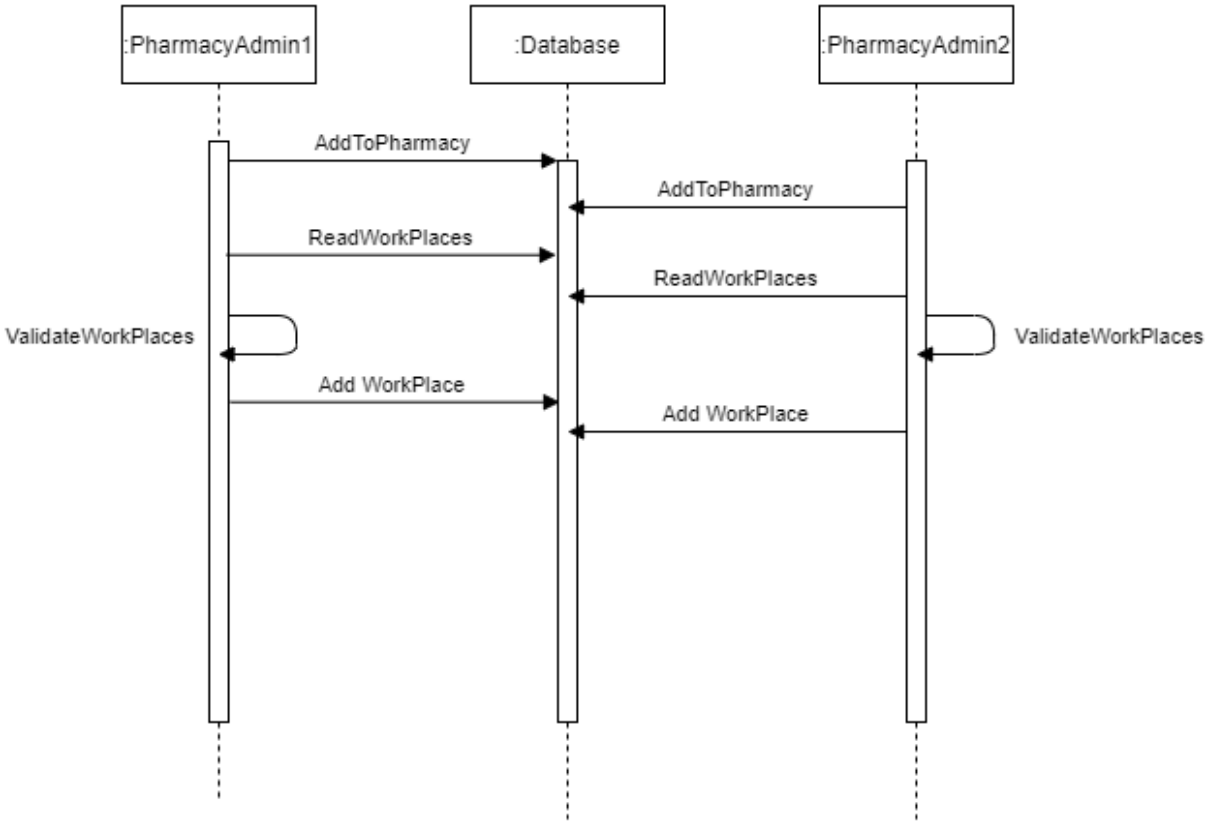
## Dodavanje dermatologa u apoteku od strane administratora apoteke:

Problem nastaje u slučaju da su dva administratora apoteke u isto vreme odlučila da dodaju dermatologa kao zaposlenog u svojoj apoteci. Kao što se može videti na slici, problem nastaje prilikom validacije radnih mesta dermatologa. Pošto su u isto vreme pročitana sva radna mesta, a potom rađena validacija, nije moguće detektovati grešku.

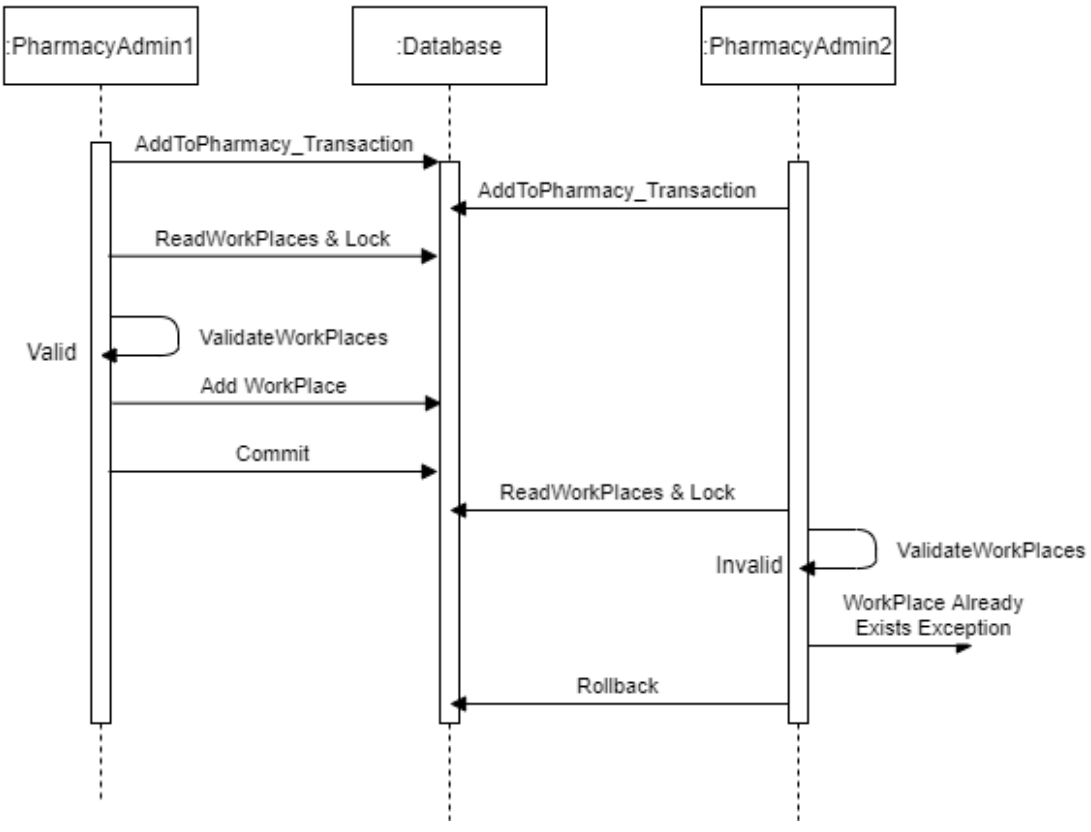
**Endpoint:**
POST /pharmacies/{pharmacyId}/dermatologists/{dermatologistId}
**Service method:**
IDermatologistService
Account AddToPharmacy(Guid pharmacyId, Guid dermatologistAccountId, WorkTime workTime);

# Add Dermatologist To Pharmacy

## Problem




## Solution



Rešenje jeste korišćenje transakcija i pesimističko zaključavanje resursa radnih mesta korišćenjem FOR UPDATE mehanizma unutar SELECT naredbe koji nam nudi MariaDB.

```
return _context.DermatologistWorkPlaces
    .FromSqlRaw(
        $"SELECT * FROM DermatologistWorkPlaces WHERE DermatologistId = \"{dermatologistId}\" AND Active = 1 FOR UPDATE;")
    .ToList();
```

```
return _context.DermatologistWorkPlaces
    .FromSqlRaw(
        $"SELECT * FROM DermatologistWorkPlaces WHERE DermatologistId = \"{dermatologistId}\" AND PharmacyId = \"{pharmacyId}\" AND Active = 1 FOR UPDATE")
    .FirstOrDefault();
```

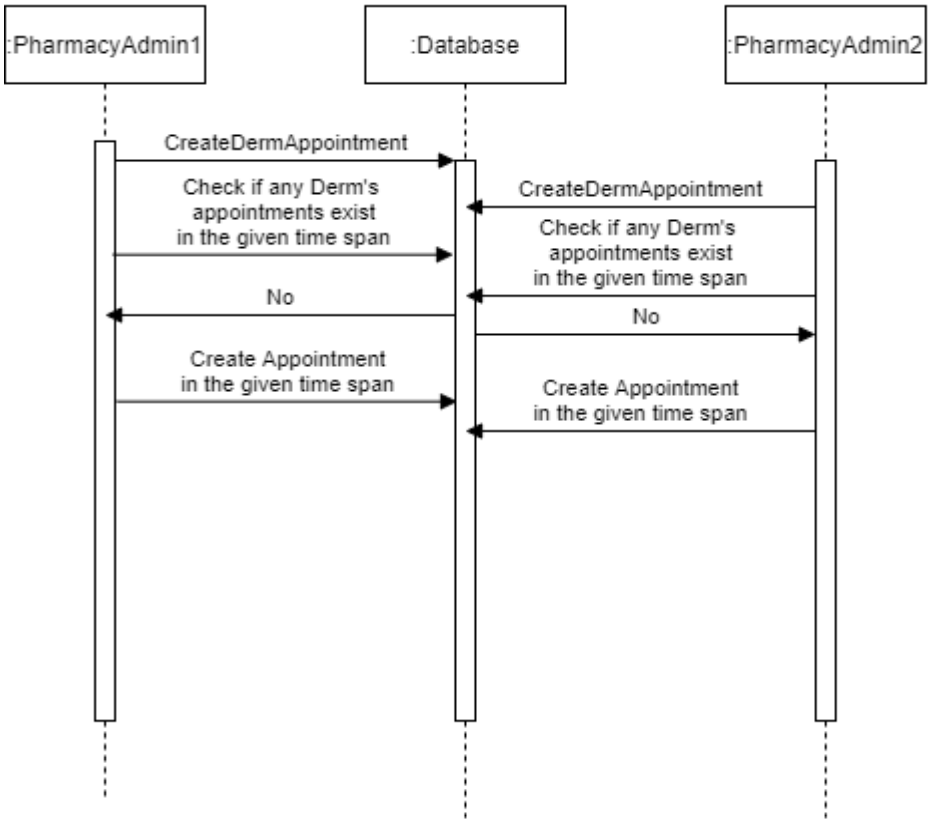
## Kreiranje slobodnih termina pregleda kod dermatologa u apoteci od strane administratora apoteke:

Problem nastaje u slučaju da su dva administratora apoteke u isto vreme odlučila da kreiraju slobodan termin za istog dermatologa u svojoj apoteci. Termini se mogu potencijalno preklapati, a da se tako nešto ne detektuje prilikom kreiranja. Slično kao u prethodnom primeru, sa slike se može videti da problem nastaje prilikom validacije slobodnih termina dermatologa. Pošto su u isto vreme pročitani svi termini, a potom rađena provera za preklapanje istih sa novim terminom, nije moguće detektovati grešku.

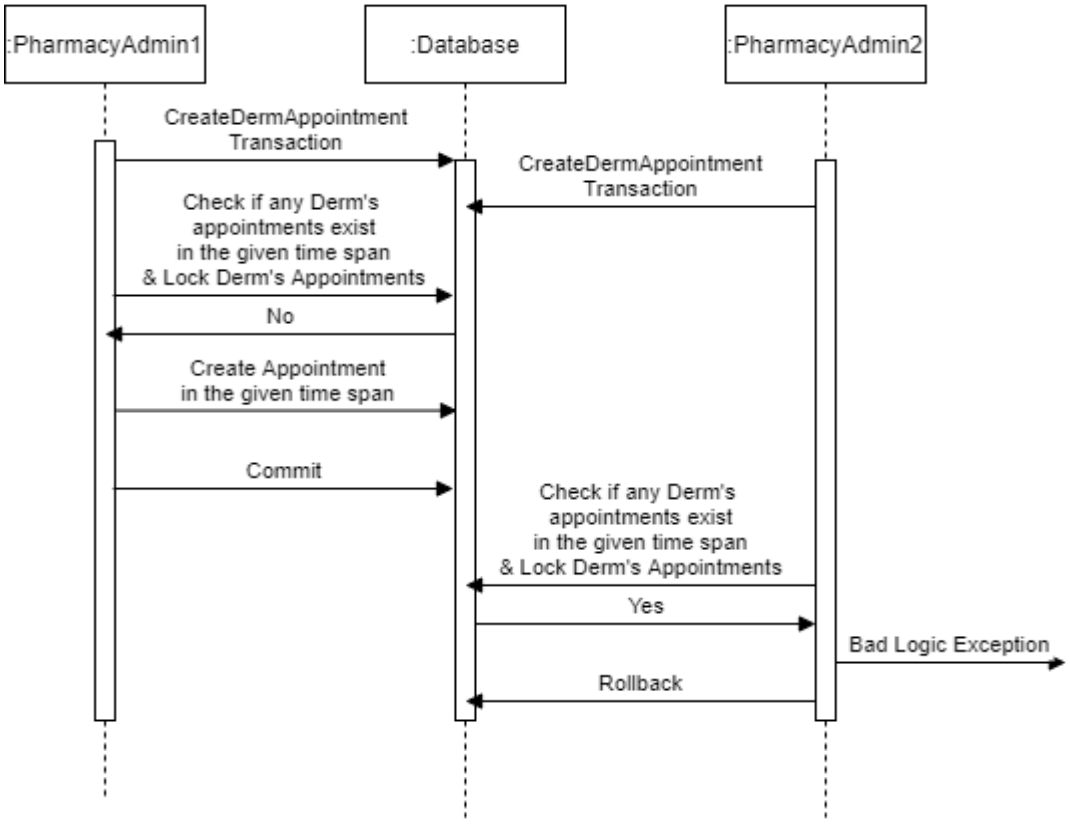
**Endpoint:**
POST /appointments/dermatologist
**Service method:**
IAppointmentService
Appointment CreateDermatologistAppointment(CreateAppointmentDTO appointmentDTO);

# Create Dermatologist Appointment

## Problem




## Solution



Rešenje jeste korišćenje transakcija i pesimističko zaključavanje resursa slobodnih termina dermatologa korišćenjem FOR UPDATE mehanizma unutar SELECT naredbe koji nam nudi MariaDB.

```
return _context.Appointments
    .FromSqlRaw($"SELECT * FROM Appointments WHERE MedicalStaffId = \"{medicalStaffId}\" AND Active = 1 FOR UPDATE;")
    .ToList();
```

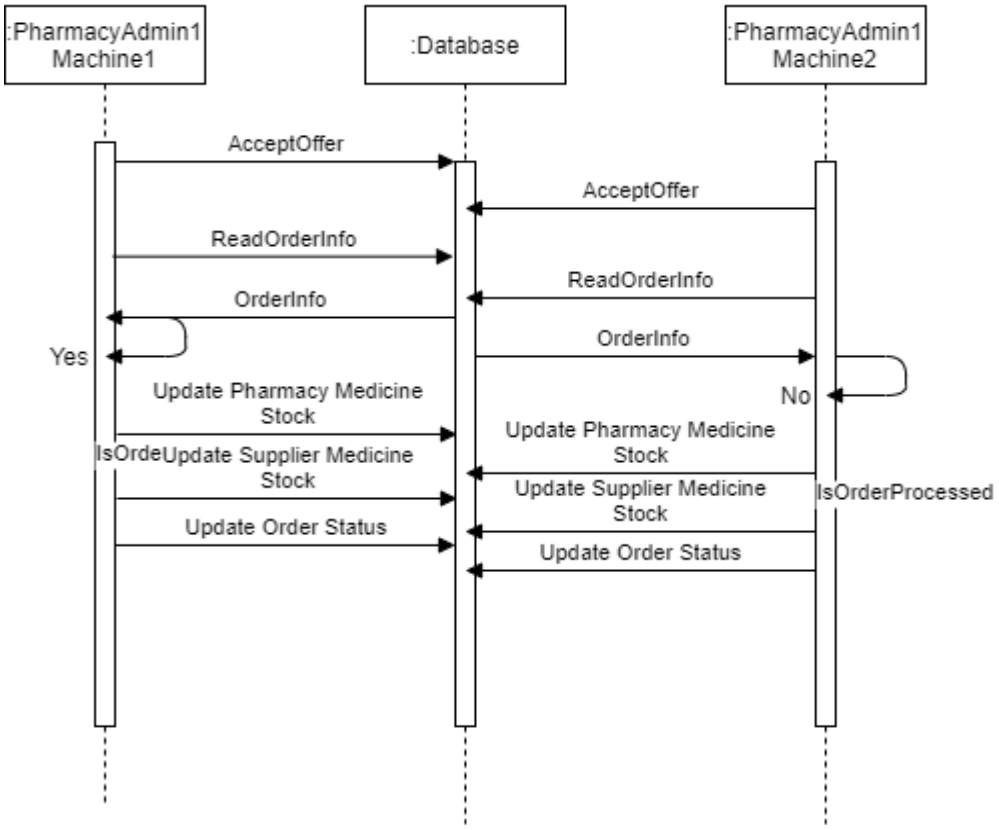
## Prihvatanje ponude za narudžbenicu od strane istog administratora apoteke sa više različitih mašina (sve je moguće):

Problem nastaje u slučaju da su jedan administrator apoteke odluči da pokuša da zgorča život finim ljudima koji kucaju softver, odnosno programerima. Dotični misli da je mnogo pametan i pokušava da prihvati istu ponudu sa dve različite mašine u isto vreme. Pošto su informacije o narudžbenici učitane u isto vreme, odnosno neposredno pred prihvatanje, oba objekta imaju vrednost atributa, koji ukazuje na to da li je narudžbenica obrađena ili ne, jednaku 0 odnosno “nije obrađena”. Tada nije moguće uraditi tačnu validaciju i ponuda za narudžbenicu će biti dva puta prihvaćena.
Ovo jeste veliki problem jer bi se u prethodno navedenom slučaju dva puta skinuli lekovi sa stanja dobavljača, odnosno uvećali u apoteci.

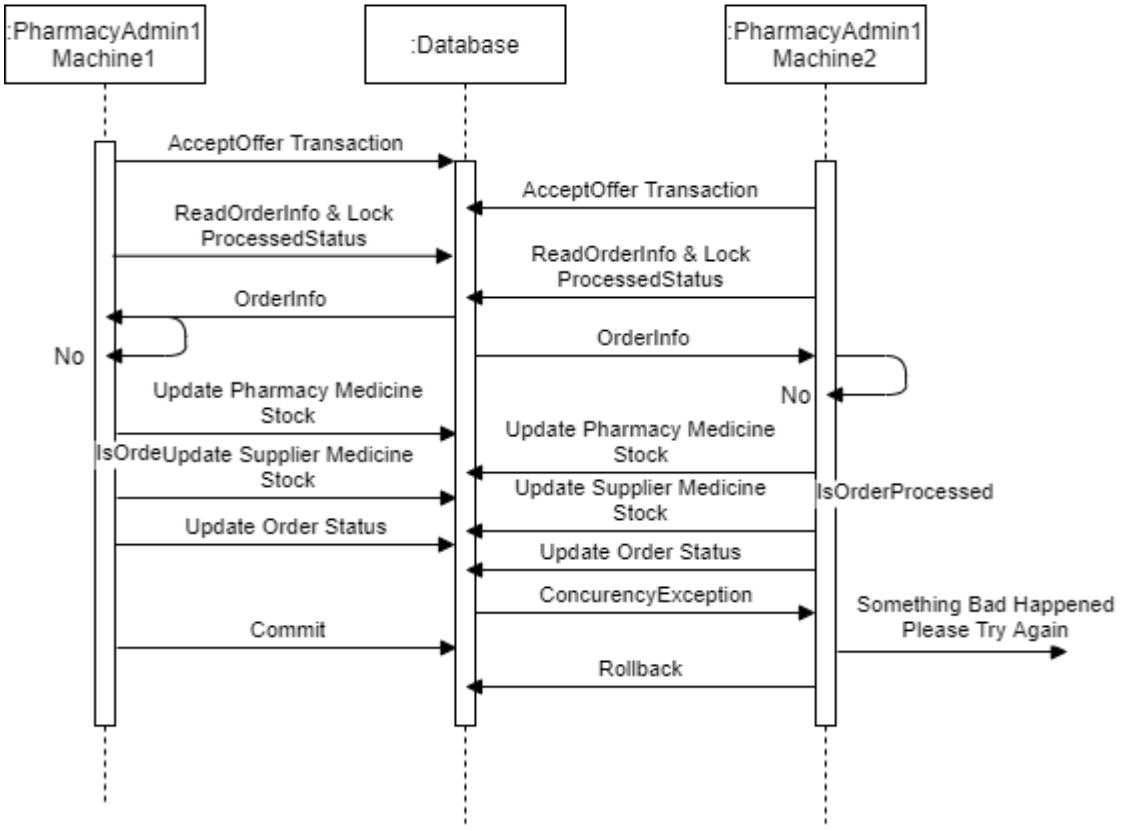
**Endpoint:**
POST /suppliers/offers/{supplierOfferId}
**Service method:**
IAppointmentService
SupplierOffer AcceptOffer(Guid offerId, Guid pharmacyAdminId);

# Accept Supplier Offer

## Problem




## Solution



Rešenje jeste korišćenje transakcija i optimistično zaključavanje polja IsProcessed resursa narudžbenice. Korišćen je mehanizam, odnosno atribut [ConcurrencyCheck] koju nam nudi korišćeni ORM Entity Framework Core. Ukoliko dođe do prethodno navedene situacije, jedan od zahteva neće biti uspešno obrađen jer će doći do DbUpdateConcurrencyException, nakon čega bi korisniku bilo stavljeno do znanja da je nešto pošlo po zlu i imaće opciju da ponovi istu operaciju.

## Korisni linkovi:

1. https://www.google.com/
2. https://stackoverflow.com/
3. https://mariadb.com/kb/en/lock-in-share-mode/
4. https://thinkrethink.net/2017/10/02/implement-pessimistic-concurrency-in-entity-frame work-core/
5. https://www.learnentityframeworkcore.com/concurrency
6. https://www.entityframeworktutorial.net/entityframework6/transaction-in-entity-framew ork.aspx