Rešavanje konfliktnih situacija

Aleksa Milenović, IN30/2017

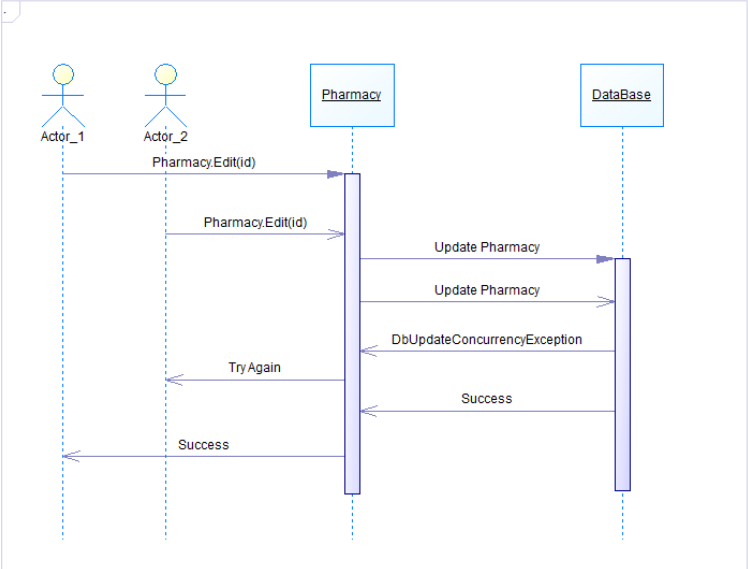
1.Izmena podataka apoteke

Opis funkcionalnosti:

Svaki administrator apoteke može da uređuje profil apoteke, pod uslovom da je provereno da on radi bas u njoj. On otvara stranicu podataka o apoteci, menja podatke koje zeli i pritiskom na dugme potvrdjuje izmene.

Opis problema:

Prilikom izmene podataka moze se desiti da su dva ili vise administratora apoteka, koji rade u istoj ovoj apoteci, takodje poslali zahtev za izmenu podataka apoteke. Neophodno je obezbediti da ne dolazi do preplitanja i da se samo prvi zahtev upiše u bazu.



Rešenje problema:

Za rešavanje ovog problema korišćen je optimistički metod, zato što je veoma mala verovatnoca dogadjanja ove vrste preplitanja. Prilikom pokušaja upisivanja promena može desi da se neka promena već izvršila u bazi dok je administrator kucao vrednosti koje je hteo da izmeni. Kako bi se ovakve situacije mogle detektovani uvedeno je polje „RowVersion“. Prilikom upisa vrši se provera da li se razlikuju vrednosti „RowVersion“-a i ako se razlikuju dolazi do izuzetka „DbUpdateConcurrencyException“ prilikom kog se ovaj zahtev nece dovrsiti i samim tim nece se modifikovati vrednosti podataka apoteke u bazi. Administrator apoteke se obaveštava o grešci.

```
// POST: Pharmacies/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
0 references
public async Task<IActionResult> Edit(Guid id, [Bind("Id,Name,Location,RowVersion")] Pharmacy pharmacy)
{
    if (id != pharmacy.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(pharmacy);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!PharmacyExists(pharmacy.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(pharmacy);
}
```

2.Rezervacija termina kod dermatologa

Opis funkcionalnosti:

Pacijent ima mogucnost rezervacije termina koje je administrator apoteke vec unapred kreirao.

Opis problema:

Prilikom rezervacije termina korisnik salje zahtev za njegovo zauzimanje, medjutim moze doci da dva pacijenta u isto vreme posalju zahtev gde se treba obezbediti rezervacija prvog poslatog.

Rešenje problema:

Kao i u prethodnom primeru ovaj problem je resen koriscenjem optimisticke metode. Prilikom pokušaja rezervacije moze se desiti da se termin vec zauzeo u bazi. Radi detekcije ovih situacija ponovo je uvedeno polje „RowVersion“. Prilikom upisa vrši se provera da li se razlikuju vrednosti „RowVersion“-a i ako se razlikuju dolazi do izuzetka „DbUpdateConcurrencyException“. Pacijent je obavesten ukoliko nije u mogucnosti da rezervise termin.

```
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Patient")]
0 references
public async Task<IActionResult> CreateAppointmentRequest([Bind("PharmacyId,PatientId,MedicalWorkerId,StartTime,EndTime,Rating,Price,Comment,Id,RowVersion")] Appointment appointment)
{
    //patients dont get to choose the length of the appointment
    appointment.EndTime = appointment.StartTime.AddMinutes(30);

    var medExpertsIds = _context.WorkingContract.
        Where(x => x.PharmacyId == appointment.PharmacyId)
        .Select(x => x.WorkerId)
        .ToList();

    var freeMedExperts = _context.Users
                        .Where(x => medExpertsIds.Contains(x.Id)).ToList();

    foreach(var medExpert in freeMedExperts.ToList())
    {
        var appointments = _context.Appointment
                        .Where(x => x.MedicalWorkerId == medExpert.Id)
                        .ToList();

        bool isFree = true;

        foreach(var app in appointments)
        {
            if(IsOverlapping(app, appointment))
            {
                isFree = false;
                break;
            }
        }

        if (!isFree)
            freeMedExperts.Remove(medExpert);

        if ((await _userManager.GetRolesAsync(medExpert)).FirstOrDefault() == "Admin")
            freeMedExperts.Remove(medExpert);
    }

    if(freeMedExperts.Count == 0)
    {
        return View("ErrorMessage", "No free med experts for this appointment");
    }

    ViewData["freeMedExperts"] = freeMedExperts;
    appointment.PatientId = (await _userManager.GetUserAsync(User)).Id;

    return View("Create", appointment);
}

1 reference
public bool IsOverlapping(Appointment a1, Appointment a2)
{
    if (a1.StartTime <= a2.StartTime && a1.EndTime >= a2.EndTime)
        return true;
    if (a1.StartTime >= a2.StartTime && a1.EndTime <= a2.EndTime)
        return true;

    return false;
}
```

```
// GET: Appointments/TakeAppointment/5
[Authorize(Roles = "Patient")]
0 references
public async Task<IActionResult> TakeAppointment(Guid? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var appointment = await _context.Appointment.FindAsync(id);

    appointment.PatientId = (await _userManager.GetUserAsync(User)).Id;

    try
    {
        _context.Update(appointment);
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!AppointmentExists(appointment.Id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return View("Successful");
}
```

3.Izmena leka

Opis funkcionalnosti:

Administrator ima mogucnost izmene podataka jednog leka.

Opis problema:

Prilikom menjanja leka salje se zahtev za njegovu izmenu, ali opet moze doci do preplitanja i situacije gde dva administratora salju isti zahtev, pored ovog moze se desiti i da u tabu pored bude izvrsena neka izmena.

Rešenje problema:

Kao i u prethodnom primeru ovaj problem je resen koriscenjem optimisticke metode. Pri izmeni moze se desiti da se vrednost „RowVersion“-a u bazi promenilo i onda ce se sadasnja vrednost razlikovati od te. Ovim dolazi do izuzetka „DbUpdateConcurrencyException“. Admininstrator dobija obavestenje da je lek vec izmenjen.

```
// POST: Drugs/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin,Pharmacist")]
0 references
public async Task<IActionResult> Edit(Guid id, [Bind("Name,Description,Ingredients,Notes,Id,RowVersion")] Drug drug)
{
    if (id != drug.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(drug);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!DrugExists(drug.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(drug);
}
```