

**SVEUČILIŠTE U ZAGREBU**  
**FAKULTET ORGANIZACIJE I INFORMATIKE**  
**V A R A Ź D I N**

**Hrvoje Pavić**

**Matični broj: 45027**

**Studij: Organizacija poslovnih sustava**

**IZRADA APLIKACIJE ZA PRAĆENJE TRUDNOĆE - TRUDNOCAAPP**

**Mentor:**

Ph.D. Bogdan Okreša Đurić

**Varaždin, siječanj 2023.**

*Hrvoje Pavić*

### **Izjava o izvornosti**

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

*Autor/ica potvrdio/la prihvaćanjem odredbi u sustavu FOI-radovi*

---

## Sažetak

U ovom radu kreirana je aplikacija za praćenje trudnoće. Prvo je opisan teoretski dio, a zatim je prikazan model naše baze podataka. Nadalje objašnjeno je kako se baza podataka koristi u samoj aplikaciji, odnosno interakcija baze podataka i aplikacije. Na kraju su prikazane slike same aplikacije i opisano što se točno događa na svakoj formi. **Sama aplikacija** kreirana je za prethodni termin obrane projekta, kada je i obranjena. Međutim falilo je temporalnih komponenti, a kako se obranjena aplikacija nije mogle prenijeti u novu akademsku godinu, kostur same aplikacije je i dalje uzet, a temporalne komponente koje su falile su dodane u istu, te su određene funkcionalnosti poboljšane.

**Ključne riječi:** sql; sql server; linq; trudnoća; c#; win forms;

# Sadržaj

<b>1. Opis aplikacijske domene</b>	<b>1</b>
<b>2. Teorijski uvod</b>	<b>2</b>
2.1. Aktivne baze podataka	2
2.2. Temporalne baze podataka	2
2.3. LINQ	2
<b>3. Model baze podataka</b>	<b>3</b>
<b>4. Implementacija</b>	<b>4</b>
4.1. Baza podataka	4
4.2. Aplikacija i baza podataka	5
<b>5. Prikaz aplikacije - tekst</b>	<b>19</b>
<b>6. Prikaz aplikacije - slike</b>	<b>21</b>
<b>7. Zaključak</b>	<b>28</b>
<b>Popis literature</b>	<b>29</b>
<b>Popis slika</b>	<b>30</b>
<b>Popis tablica</b>	<b>31</b>

# 1. Opis aplikacijske domene

U ovom radu napravljena je windows form aplikacija za praćenje trudnoće. Korisnik aplikacije nakon što se uspješno prijavi može pratiti razvoj svoje bebe. Moguće je mjeriti broj udaraca koje beba napravi u određenom vremenskom intervalu, te dodijeliti razne informacije bebi.

Baza podataka koja se koristi u ovom radu je kreirana pomoću Microsoft SQL Management Studia, a sve SQL naredbe su pisane u standardnom DDL obliku za kreiranje tablica i kreiranje trigger-a. Sama aplikacija je razvijena u programskom jeziku C# preko Visual Studia 2022. Manipulacija podataka je napravljeno uz pomoć LINQ framework-a.

## **2. Teorijski uvod**

### **2.1. Aktivne baze podataka**

Kako bi omogućili napredne karakteristike baza podataka potrebno je uvesti aktivnu komponentu u relacijske baze podataka. Aktivna komponenta može biti okidač, upozorenje, napredno ograničenje, zaštita... U ovom radu koristili smo okidače prilikom dodavanja novih vrijednosti u tablicu i prilikom ažuriranja vrijednosti. Uz pomoć okidača uspjeli smo prilikom dodavanja ili ažuriranja podataka određene podatke upisati u neke druge tablice s koje nismo trenutno koristili.[1]

### **2.2. Temporalne baze podataka**

Temporalne baze podataka omogućuju uvođenje temporalne dimenzije, odnosno vremenske dimenzije, što znači da svaki objekt ima svoj životni ciklus. Životni ciklus objekta je moguće onda pratiti jer se može pohraniti prošlo, sadašnje i buduće vrijeme.[1]

U ovom radu, konkretno za temporalnu komponentu kreirane su dvije tablice Rekreacija i Medicina preko kojih je sama temporalna komponenta najbolje objašnjena. Navedene tablice su detaljnije objašnjene u nastavku rada.

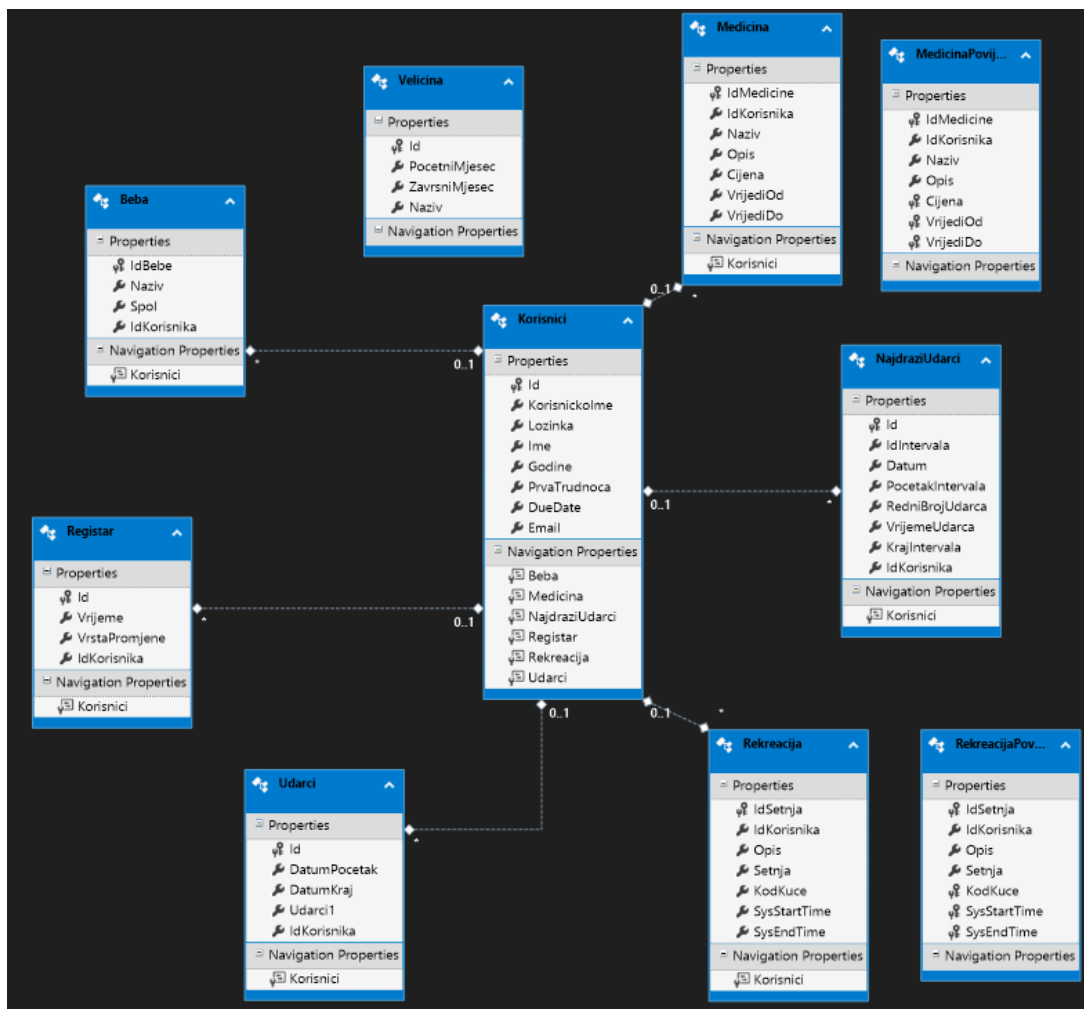
### **2.3. LINQ**

Jezično integrirani upit (LINQ) moćan je skup tehnologija koje se temelje na integraciji mogućnosti upita izravno u C# jezik.

LINQ (Language Integrated Query) je jednolična sintaksa upita u C# za dohvaćanje podataka iz različitih izvora i oblika. Integriran je u C#, čime se uklanja neusklađenost između programskih jezika i baza podataka, kao i pruža jedinstveno sučelje za upite za različite vrste izvora podataka.[2]

### 3. Model baze podataka

Prilikom izrade aplikacije koristili smo 5 kreiranih tablica(1). Glavna tablica je Korisnici koja je povezana na tablicu Beba, Registar i Udarce. Tablica Velicina je sporedna koja ima svoje unesene vrijednosti za veličinu određenog povrća/voća u odnosu na starost bebe.



Slika 1: Model

## 4. Implementacija

### 4.1. Baza podataka

Baza podataka je kreirana u MS SQL Studiu, a SQL naredbe su priložene uz ovaj dokument za kreiranje pojedinih tablica. Također su kreirani pojedini triggeri.

Triggere smo mogli napraviti i za obavijesti prilikom rada s bazom podataka, odnosno unosa podataka, međutim obavijesti i restrikcije su riješene programski, a triggeri su korišteni kako bi prilikom dodavanja ili ažuriranja, potrebne podatke dodali u nove tablice.

Prvi trigger koji ćemo pojasniti kreira red u tablici Beba za Korisnika nakon što se on registrira u sustav. Kako svaki korisnik mora imati podatke o bebi, ovo je izvršeno preko triggera.

```
CREATE TRIGGER [dbo].[Dodaj_bebu]
    ON [dbo].[Korisnici]
    AFTER INSERT
AS
BEGIN
    INSERT INTO [dbo].[Beba] (spol, IdKorisnika) VALUES ('đNeodreeno', (SELECT
        Id From INSERTED))
END
```

Nadalje prilikom ažuriranja bebe kreiran je trigger koji stavlja podatke u tablicu Registar koja onda služi Adminu za kontrolu.

```
CREATE TRIGGER [dbo].[Napuni_registarUpdate]
    ON [dbo].[Beba]
    AFTER UPDATE
AS
BEGIN
    INSERT INTO [dbo].[Registar] (Vrijeme, VrstaPromjene, IdKorisnika) VALUES (
        GETDATE(), 'Azuriranje bebe', (SELECT IdKorisnika From INSERTED) )
END
```

Svaki puta kada korisnik zabilježi udarac bebe, odnosno kada sesija udaraca završi i ti podaci se unesu u tablicu Udarci, ovaj trigger upisuje u tablicu Registri u koje vrijeme se dogodilo unošenje u tablicu od strane korisnika.

```
CREATE TRIGGER [dbo].[Upisi_u_registar]
    ON [dbo].[Udarci]
    AFTER INSERT
AS
BEGIN
    INSERT INTO [dbo].[Registar] (Vrijeme, VrstaPromjene, IdKorisnika) VALUES (
        GETDATE(), 'Dodavanje udarca', (SELECT IdKorisnika From INSERTED) )
END
```

Prilikom mjerenja udaraca bebe, u slučaju da se u određenom intervalu osjeti više od 20 udaraca ispisuje se poruka u aplikaciji koja kaže da je potrebno posjetiti doktora. Brojka 20



udaraca nije nužno bitna da je nakon nje potrebno posjetiti doktora, ali je ovdje stavljena (a ne neka veća vrijednost) radi jednostavnosti prilikom demonstracije. Veći broj udaraca bebe može značiti da beba ima nekih problema i da je stvarno potrebno odmah posjetiti dežurnog doktora.

```
CREATE TRIGGER [dbo].[Javi_poruku]
    ON [dbo].[NajdraziiUdarci]
    AFTER INSERT
AS
    IF ((SELECT RedniBrojUdarca From INSERTED) > 20)
BEGIN
    RAISERROR('Posjetite doktora',16,1);
END;
```

## 4.2. Aplikacija i baza podataka

Aplikacija i baza podataka u programu se prvi put susreću prilikom Registracije korisnika. Korisnik mora popuniti određene podatke te zatim ako su podaci ispravno popunjeni oni se spremaju u samu tablicu dbo.Korisnici.

```
private void RegistrirajMe()
{
    using (var context = new TrudnocaAppEntities())
    {
        string korisnickoIme = tbKorisnickoImeRegistracija.Text;
        string lozinka = tbLozinka.Text;
        string ime = tbIme.Text;
        int godine = int.Parse(cbGodine.SelectedItem.ToString());
        DateTime dueDate = dtpDueDate.Value.Date;
        string email = tbEmail.Text;

        if (rbPrvaTrudnocaDa.Checked == true)
        {
            Korisnici noviKorisnik = new Korisnici
            {
                KorisnickoIme = korisnickoIme,
                Lozinka = lozinka,
                Ime = ime,
                Godine = godine,
                PrvaTrudnoca = true,
                DueDate = dueDate,
                Email = email
            };

            context.Korisnici.Add(noviKorisnik);
            context.SaveChanges();
        }
        else if (rbPrvaTrudnocaNe.Checked == true)
        {
            Korisnici noviKorisnik = new Korisnici
            {
                KorisnickoIme = korisnickoIme,
```

```

        Lozinka = lozinka,
        Ime = ime,
        Godine = godine,
        PrvaTrudnoca = false,
        DueDate = dueDate,
        Email = email
    };

    context.Korisnici.Add(noviKorisnik);
    context.SaveChanges();
}
}

Close();
}

```

Skoro sva ograničenja su napravljena na razini aplikacije, što će biti prikazano u sljedećem poglavlju. Međutim ograničenje korisničkog imena je napravljeno preko upita s bazom, gdje se provjerava da korisničko ime koje korisnik prilikom registracije želi ne postoji u bazi podataka.

```

private bool ProvjeriDuploKorisnickoIme()
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from k in context.Korisnici
                     select k.KorisnickoIme;

        foreach (var item in query)
        {
            if (item == tbKorisnickoImeRegistracija.Text)
                return true;
        }

        return false;
    }
}

```

Nakon registracije slijedi prijava u aplikaciju. Unesemo nove podatke za korisničko ime i lozinku te ako su oni ispravni ulazimo u samu aplikaciju.

```

private bool ProvjeriKorisnickoIme()
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from k in context.Korisnici
                     select k.KorisnickoIme;

        foreach (var item in query)
        {
            if (item == tbKorisnickoIme.Text)
                return true;
        }
    }
}

```

```

        return false;
    }
}

private bool ProvjeriLozinku()
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from k in context.Korisnici
                     where k.KorisnickoIme == tbKorisnickoIme.Text
                     select k.Lozinka;

        foreach (var item in query)
        {
            if (item == tbLozinka.Text.ToString())
                return true;
        }

        return false;
    }
}

```

Također ako su podaci ispravno uneseni želimo dohvatiti Id korisnika kako bi ga mogli proslijediti aplikaciji.

```

private int DohvatiIdKorisnika()
{
    int idKorisnika = 0;

    using (var context = new TrudnocaAppEntities())
    {
        var query = from k in context.Korisnici
                     where k.KorisnickoIme == tbKorisnickoIme.Text
                     select k.Id;

        foreach (var item in query)
        {
            idKorisnika = int.Parse(item.ToString());
        }
    }

    if(idKorisnika > 0)
        return idKorisnika;
    return 0;
}

```

Nakon što smo se uspješno prijavili u aplikaciju, idućoj formi se proslijeđuje Id korisnika koji se prijavio. Kako bi prikazali korisničko ime korisnika koji je prijavljen, napravili smo upit na bazu podataka gdje želimo ispisati na zaslon korisničko ime na temelju proslijeđenog Id-a.

```

private void RefreshGUI()
{

```

```

        using (var context = new TrudnocaAppEntities())
        {
            var query = from k in context.Korisnici
                        where k.Id == idKorisnika
                        select k.KorisnickoIme;

            foreach (var item in query)
            {
                tbPrijavljeniKorisnik.Text = item;
            }
        }
    }
}

```

Trenutno u aplikaciji imamo više opcija, ako želimo odabrati opciju Moja Beba izvršit će se nekoliko upita da bi nam se svi potrebni podaci prikazali u novoj formi. Prije svega potrebno je popuniti same podatke o bebi. Oni se rade preko upita :

```

private void PopuniPodatkeBebe()
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from b in context.Beba
                    where b.IdKorisnika == korisnikovId
                    select b;

        foreach (var item in query)
        {
            if(item.Naziv != null)
            {
                tbImeBebe.Text = item.Naziv.ToString();
                labelMojaBebaImeBebeDrugi.Text = tbImeBebe.Text;
                labelMojaBebaImeBebeDrugi.Visible = true;
            }
            else
            {
                labelFaliMiIme.Visible = true;
            }

            cbSpol.Text = item.Spol.ToString();
        }
    }
}

```

Također se popunjavaju podaci za datum termina, koji se dohvaćaju iz tablice Korisnici te se onda ti podaci uređuju kako bi ljepše izgledali na prikazu. Također pozivamo pomoćnu tablicu Velicina iz koje uzimamo podatke o veličini bebe na temelju starosti bebe.

```

private void PopuniVelicinu()
{
    int starost = int.Parse(labelStarostPopunjeno.Text.Substring(0, 1));

    using (var context = new TrudnocaAppEntities())
    {

```

```

        var query = from v in context.Velicina
                     where v.PocetniMjesec <= starost && v.ZavrsniMjesec >=
                        starost
                     select v;

        foreach (var item in query)
        {
            labelMojaBebaTekstSesti.Text = item.Naziv.ToString();
        }
    }
}

```

Također ako korisnik želi ažurirati podatke o bebi potrebno je napraviti UPDATE nad bazom podataka ako su svi uneseni podaci ispravni.

```

private void AzurirajPromjene()
{
    int brojacPromjena = 0;

    using (var context = new TrudnocaAppEntities())
    {
        var query = from b in context.Beba
                     where b.IdKorisnika == korisnikovId
                     select b;

        foreach (Beba item in query)
        {
            if (tbImeBebe.Text != item.Naziv && tbImeBebe.Text != string.
                Empty)
            {
                item.Naziv = tbImeBebe.Text;
                brojacPromjena++;
                labelFaliMiIme.Visible = false;
            }
            if (cbSpol.SelectedItem.ToString() != item.Spol)
            {
                item.Spol = cbSpol.SelectedItem.ToString();
                brojacPromjena++;
                if (cbSpol.SelectedItem.ToString() == "šMuko")
                {
                    labelMojaBebaTekstPrvi.BackColor = Color.LightBlue;
                    labelMojaBebaTekstTreci.BackColor = Color.LightBlue;
                    labelMojaBebaTekstPeti.BackColor = Color.LightBlue;
                }
                else if (cbSpol.SelectedItem.ToString() == "Žensko")
                {
                    labelMojaBebaTekstPrvi.BackColor = Color.LightPink;
                    labelMojaBebaTekstTreci.BackColor = Color.LightPink;
                    labelMojaBebaTekstPeti.BackColor = Color.LightPink;
                }
                else
                {
                    labelMojaBebaTekstPrvi.BackColor = Color.White;

```

```

        labelMojaBebaTekstTreci.BackColor = Color.White;
        labelMojaBebaTekstPeti.BackColor = Color.White;
    }
}

if (brojacPromjena == 0)
    labelNemaPromjena.Visible = true;
else
    context.SaveChanges();
}
}

```

Nadalje korisnik može ažurirati svoje podatke. Podaci se dohvaćaju i prikazuju na ekranu kao na navedenim primjerima do sada, te se onda uz provjeru da su svi podaci ispravni i ažuriraju.

```

private void IzvrsiAzuriranje()
{
    int brojacPromjena = 0;

    using (var context = new TrudnocaAppEntities())
    {
        var query = from k in context.Korisnici
                     where k.KorisnickoIme == korisnikovaBeba
                     select k;

        foreach (Korisnici item in query)
        {
            if (tbImeAzuriraj.Text != item.Ime)
            {
                item.Ime = tbImeAzuriraj.Text;
                brojacPromjena++;
            }
            if (cbGodineAzuriraj.SelectedItem != null)
            {
                item.Godine = int.Parse(cbGodineAzuriraj.SelectedItem.
                    ToString());
                brojacPromjena++;
            }
            if (dtpDueDateAzuriraj.Value != item.DueDate)
            {
                item.DueDate = dtpDueDateAzuriraj.Value;
                brojacPromjena++;
            }
            if (tbEmailAzuriraj.Text != item.Email)
            {
                item.Email = tbEmailAzuriraj.Text;
                brojacPromjena++;
            }
            if (rbPrvaTrudnocaDaAzuriraj.Checked == true)
            {
                if (item.PrvaTrudnoca == false)

```

```

        {
            item.PrvaTrudnoca = true;
            brojacPromjena++;
        }
    }
    if (rbPrvaTrudnocaDaAzuriraj.Checked == false)
    {
        if (item.PrvaTrudnoca == true)
        {
            item.PrvaTrudnoca = false;
            brojacPromjena++;
        }
    }
}

if (brojacPromjena == 0)
    labelNemaPromjena.Visible = true;
else
{
    context.SaveChanges();
}
}
}

```

Sljedeća opcija koju korisnik može je mjeriti udarce bebe. Prilikom ulaska na opciju Udarci korisniku se prikazuje tablica preko data grid view-a u kojoj se prikazuju podaci o udarcima koje korisnik ima do sada. Podatke za data grid view dohvaćamo :

```

private object NapuniDGV()
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from u in context.Udarci
                     where u.IdKorisnika == korisnikovId
                     select u;

        return query.ToList();
    }
}

```

Korisnik mora prvo kliknuti na početak perioda kako bi krenulo mjerenje udaraca. Sa tim klikom izvršava se sljedeći dio koda :

```

public void btnUdarci_Click(object sender, EventArgs e)
{
    btnStop.Enabled = true;
    roundButtonUdarac.Enabled = true;

    using (var context = new TrudnocaAppEntities())
    {
        var query = from u in context.NajdrziUdarci
                     where u.IdKorisnika == korisnikovId
                     select u;
    }
}

```

```

        if (query.Count() > 0)
        {
            idIntervala = query.ToList()[query.Count()-1].IdIntervala + 1;
        }
        else
        {
            idIntervala = 1;
        }
    }

    datum = DateTime.Now.Date;
    pocetakIntervala = DateTime.Now.TimeOfDay;
    vrijemeUdarca = DateTime.Now.TimeOfDay;
    krajIntervala = null;

    labelPocVrijZadInt.Visible = true;
    labelPocVrijZadInt.Text = pocetakIntervala.ToString();
    labelZavVrijZadInt.Visible = false;

    NajdraziUdarci udarci = new NajdraziUdarci()
    {
        IdIntervala = idIntervala,
        Datum = datum,
        PocetakIntervala = pocetakIntervala,
        RedniBrojUdarca = 1,
        VrijemeUdarca = vrijemeUdarca,
        KrajIntervala = krajIntervala,
        IdKorisnika = korisnikovId
    };

    using (var context = new TrudnocaAppEntities())
    {
        context.NajdraziUdarci.Add(udarci);
        context.SaveChanges();
    }

    UcitajDGV();

    btnUdarci.Enabled = false;
    labelBrojUdaraca.Text = redniBrojUdarca.ToString();
    labelRazlikaVremena.Visible = false;
    labelTrajanjeSesije.Visible = false;
}

```

Interval je pokrenut te korisnik svaki puta kada osjeti udarac klikom na gumb udarac zabilježava pojedini udarac u tablicu, a kod koji se izvršava prilikom klika na udarac je :

```

private void roundButtonUdarac_Click(object sender, EventArgs e)
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from u in context.NajdraziUdarci

```



```

        where u.IdKorisnika == korisnikovId
        select u;

        idIntervala = query.ToList()[query.Count() - 1].IdIntervala;
    }

    datum = DateTime.Now.Date;
    pocetakIntervala = null;
    redniBrojUdarca++;
    vrijemeUdarca = DateTime.Now.TimeOfDay;
    krajIntervala = null;
    labelBrojUdaraca.Text = redniBrojUdarca.ToString();

    try
    {
        NajdraziUdarci udarci = new NajdraziUdarci()
        {
            IdIntervala = idIntervala,
            Datum = datum,
            PocetakIntervala = pocetakIntervala,
            RedniBrojUdarca = redniBrojUdarca,
            VrijemeUdarca = vrijemeUdarca,
            KrajIntervala = krajIntervala,
            IdKorisnika = korisnikovId
        };
        using (var context = new TrudnocaAppEntities())
        {
            context.NajdraziUdarci.Add(udarci);
            context.SaveChanges();
        }

        UcitajDGV();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.InnerException.GetBaseException().ToString());
    }
}

```

Također prilikom mjerenja udaraca kada korisnik klikne na gumb Stop, interval završava. Detalji o ovoj funkcionalnosti nalaze se u sljedećem poglavlju prilikom tekstualnog opisa aplikacije. Kod koji se izvršava :

```

private void btnStop_Click(object sender, EventArgs e)
{
    using (var context = new TrudnocaAppEntities())
    {
        var query = from u in context.NajdraziUdarci
                     where u.IdKorisnika == korisnikovId
                     select u;

        idIntervala = query.ToList()[query.Count() - 1].IdIntervala;
        vrijemeUdarca = query.ToList()[query.Count() - 1].VrijemeUdarca;
    }
}

```

```

    }

    datum = DateTime.Now.Date;
    krajIntervala = DateTime.Now.TimeOfDay;

    NajdraziiUdarci udarci = new NajdraziiUdarci()
    {
        IdIntervala = idIntervala,
        Datum = datum,
        PocetakIntervala = null,
        RedniBrojUdarca = null,
        VrijemeUdarca = vrijemeUdarca,
        KrajIntervala = krajIntervala,
        IdKorisnika = korisnikovId
    };

    using (var context = new TrudnocaAppEntities())
    {
        context.NajdraziiUdarci.Add(udarci);
        context.SaveChanges();
    }

    UcitajDGV();

    using (var context = new TrudnocaAppEntities())
    {
        var query2 = from u in context.NajdraziiUdarci
                      where u.IdKorisnika == korisnikovId
                      select u;

        TimeSpan zadnjeVrijeme = (TimeSpan)query2.ToList()[query2.Count() - 1].KrajIntervala;
        labelZavVrijZadInt.Text = zadnjeVrijeme.ToString();
    }

    var razlikaVremena = DateTime.Parse(labelZavVrijZadInt.Text.ToString()).
        Subtract(DateTime.Parse(labelPocVrijZadInt.Text.ToString()));
    labelRazlikaVremena.Text = razlikaVremena.ToString();
    labelRazlikaVremena.Visible = true;
    labelTrajanjeSesije.Visible = true;

    labelPocVrijZadInt.Visible = true;
    labelZavVrijZadInt.Visible = true;
    labelRazlikaVremena.Visible = true;
    labelTrajanjeSesije.Visible = true;

    redniBrojUdarca = 1;
    btnUdarci.Enabled = true;
    roundButtonUdarac.Enabled = false;
    btnStop.Enabled = false;
    labelBrojUdaraca.Text = "0";
}

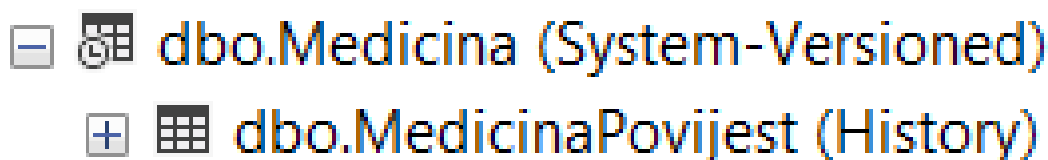
```

Prije samog objašnjenja koda vezanih za **temporalnu komponentu** potrebno je prikazati na koji način smo kreirali temporalne tablice :

```
CREATE TABLE Rekreacija
(
    IdSetnja INT IDENTITY PRIMARY KEY,
    [IdKorisnika] [int],
    Opis VARCHAR(50),
    Setnja INT,
    KodKuce BIT NOT NULL DEFAULT 1,
    SysStartTime datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
    SysEndTime datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
    FOREIGN KEY (IdKorisnika) REFERENCES Korisnici(Id)
)
WITH
(
    SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.RekreacijaPovijest)
)
```

```
CREATE TABLE Medicina
(
    IdMedicine INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
    IdKorisnika int,
    Naziv varchar(50),
    Opis varchar(50),
    Cijena decimal (10,2) NOT NULL,
    VrijediOd datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
    VrijediDo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
    PERIOD FOR SYSTEM_TIME (VrijediOd, VrijediDo),
    FOREIGN KEY (IdKorisnika) REFERENCES Korisnici(Id)
)
WITH
(
    SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.MedicinaPovijest)
)
```

Izgled samih temporalnih tablica je i u bazi drugačiji, imaju vremensku oznaku :



Slika 2: Medicina SQL server

Za formu Medicina na sljedeći način se dodaje novi lijek :

```
private void buttonDodajLijek_Click(object sender, EventArgs e)
{
    if (textBoxNazivLijeka.Text != string.Empty && textBoxOpisLijeka.Text !=
        string.Empty && textBoxCijena.Text != string.Empty)
```

```

{
    using (var context = new TrudnocaAppEntities())
    {
        string naziv = textBoxNazivLijeka.Text;
        string opis = textBoxOpisLijeka.Text;
        decimal cijena = decimal.Parse(textBoxCijena.Text);

        context.Database.ExecuteSqlCommand(String.Format("INSERT INTO
            Medicina (IdKorisnika, Naziv, Opis, Cijena) VALUES ({0},
                '{1}', '{2}', {3})", korisnikovId, naziv, opis, cijena));
    }

    UcitajDGVMedicinaTemporal();
    UcitajDGVMedicinaPovijest();
}
else
{
    MessageBox.Show("Popunite naziv, opis i cijenu kako bi mogli dodati
        novi lijek!");
}
}

```

**Također ako se mijenja cijena izvršava se sljedeći dio koda :**

```

private void buttonNovaCijena_Click(object sender, EventArgs e)
{
    if (textBoxTrenutniNaziv.Text != string.Empty)
    {
        using (var context = new TrudnocaAppEntities())
        {
            var query = (from m in context.Medicina
                where m.Naziv == textBoxTrenutniNaziv.Text
                select m).SingleOrDefault();

            if (textBoxNovaCijena.Text != string.Empty)
                query.Cijena = decimal.Parse(textBoxNovaCijena.Text);
            else
                MessageBox.Show("ŠUpiite novu cijenu kako bi se promjene
                    mogle šizvriti!");

            context.SaveChanges();
        }

        UcitajDGVMedicinaTemporal();
        UcitajDGVMedicinaPovijest();
    }
    else
    {
        MessageBox.Show("Odaberite red u tablici trenutnih lijekova kako bi
            mogli žaurirati novu cijenu!");
    }
}

```

Te na kraju moguće je obrisati lijek sa popisa trenutnih lijekova :

```
private void buttonObrisi_Click(object sender, EventArgs e)
{
    Medicina chosenMedicina = GetChosenMedicina();

    if(chosenMedicina != null)
    {
        using (var context = new TrudnocaAppEntities())
        {
            context.Medicina.Attach(chosenMedicina);
            context.Medicina.Remove(chosenMedicina);
            context.SaveChanges();
        }
    }

    UcitajDGVMedicinaTemporal();
    UcitajDGVMedicinaPovijest();
}
```

Dodavanje nove rekreacije :

```
using (var context = new TrudnocaAppEntities())
{
    string opis = textBoxNovaRekreacija.Text;
    int setnja = 0;

    context.Database.ExecuteSqlCommand(String.Format("INSERT INTO
        Rekreacija (IdKorisnika, Opis, Setnja) VALUES ({0}, '{1}',
        {2})", korisnikovId, opis, setnja));
}
```

Početak rekreacije :

```
using (var context = new TrudnocaAppEntities())
{
    var query = (from r in context.Rekreacija
        where r.Opis == textBoxTrenutniNazivRekreacije.Text
        select r).SingleOrDefault();

    query.KodKuce = false;

    context.SaveChanges();
}
```

Završetak rekreacije :

```
using (var context = new TrudnocaAppEntities())
{
    var query = (from r in context.Rekreacija
        where r.Opis == textBoxTrenutniNazivRekreacije.Text
        select r).SingleOrDefault();
}
```

```
        query.KodKuce = true;  
        query.Setnja = int.Parse(tbTrajanjeSetnje.Text.ToString());  
  
        context.SaveChanges();  
    }
```

## 5. Prikaz aplikacije - tekst

U ovom poglavlju prikaza ćemo kako aplikacija izgleda(3) te objasniti svaki zaslon posebno. Prilikom otvaranja aplikacije prikazuje se početna forma.[3]

Korisnik se prvo mora registrirati(4), tako da klikom na gumb registracija se otvara forma za registraciju.

Postoje razni uvjeti da se korisnik može registrirati, npr. da dužina korisničkog imena ne može biti prazan string ili preko 50 znakova... Ako uvjeti nisu zadovoljeni aplikacija javlja koje stvari se moraju popuniti/poboljšati(5).

Ako je registracija uspješna korisnik je preusmjeren na početnu stranicu gdje ako unese točne podatke za prijavu je preusmjeren u glavnu formu aplikacije(6) gdje postoje opcije da korisnik vidi podatke o bebi (Moja Beba), ažurira svoje korisničke podatke, broji udarce bebe ili se odjavi.

Klikom na Moja beba otvara se forma(7).

Gdje korisnik može postaviti ime bebe ako se prvi put prijavio, ili izmijeniti(8) ime i spol ako je već prethodno odabrao.

Korisnik može promijeniti(9) svoje podatke ako to želi, uz uvjet da će opet doći do provjera valjanosti unosa.

Na kraju korisnik može i mjeriti udarce(10) koje beba napravi u određenom vremenskom intervalu, a oni se prikazuju tablično. Ovo je primjer forme koja se nalazi u staroj verziji aplikacije. U novoj verziji aplikacije ova forma je poboljšana te je opisana pri dnu rada.

Također ako se admin(11) prijavi u stavu njemu se učitava tablica Registar, gdje su zabilježene sve promjene koje su se dogodile u sustavu, a samo upisivanje u registar izvršavaju triggeri.

**Dodane su nove funkcionalnosti u aplikaciju :** Prije svega Udarci sada izgledaju drugačije, poboljšano. Na ovoj slici (12) se sada može vidjeti kako izgleda forma Udarci. Korisnica kada osjeti da se beba miče u trbuhu može pokrenuti interval unutar kojeg se mjere udarci te se zapisuje vrijeme kada se koji udarac dogodio. Prvo je potrebno kliknuti na tamnozeleni gumb početak intervala. U tom trenu se u tablici Udarci dodaje novi red s vrijednostima Id intervala, Datum, Početak intervala, Redni broj udarca i Vrijeme udarca. Treba primijetiti da je vrijeme za Kraj intervala prazno jer interval još nije završio. Također pritiskom na gumb početak intervala sada je moguće kliknuti na gumb UDARAC ili na gumb kraj intervala. Ako se dogodi novi udarac korisnica klikne na gumb UDARAC i taj novi udarac se doda u tablicu. Id intervala je i dalje isti, Datum je isti, početnog vremena nema jer interval već traje, Redni broj udarca je uvećan i upisuje se novo vrijeme kada se dogodi Udarac. Vrijeme za kraj intervala je i dalje prazno sve dok korisnica ne klikne na crveni gumb kraj intervala. Također valja primijetiti da sa svakim udarcem (koji se mogu pratiti po rednom broju u tablici) se također povećava broj ukupno udaraca u intervalu, za lakše praćenje. Kada korisnica osjeti da nema više udaraca klikne na crveni gumb kraj intervala te u tom trenu interval završava, a u tablici se dodaje zadnji red za taj interval gdje

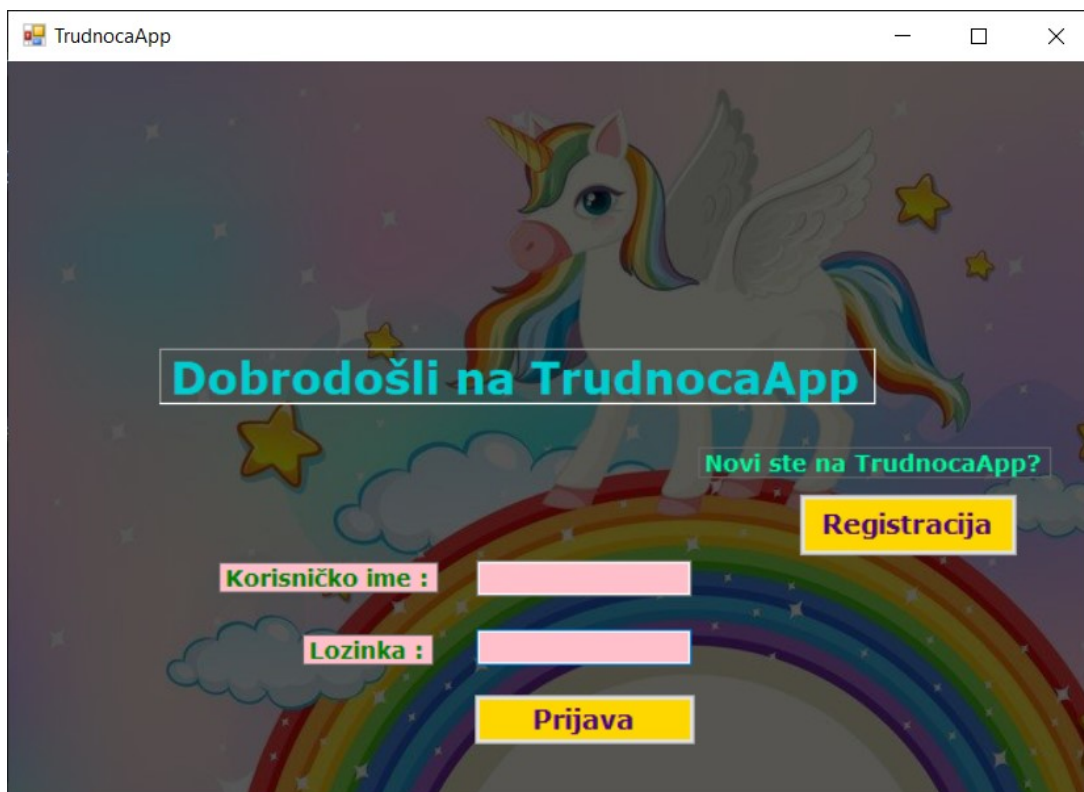
se zapisuje Vrijeme zadnjeg udarca koje je bilo (udarac koji se dogodi u predzadnjem redu), Id intervala, Datum i vrijeme za Kraj intervala. Također kada se pritisne crveni gumb za kraj intervala pojave se vrijednosti u sredini forme koje prikazuju vrijeme početka intervala, vrijeme kraja intervala i ukupno trajanje intervala (razlika vremena kraja intervala i početka intervala). Postupak se može ponoviti. Također u slučaju da se radi o velikom broju udaraca (u ovom slučaju 20) unutar intervala pojavi se poruka da korisnica treba posjetiti doktora.

**TEMPORALNA KOMPONENTA!** Dodana je nova funkcionalnost Medicina. Medicina se nalazi na glavnoj formi, a klikom na nju otvara se nova forma koja izgleda kao na slici (13). Kao što je prikazano na slici glavna poanta ove forme su dvije tablice: trenutni podaci o lijeku i povijesni podaci o lijeku. Trenutni podaci o lijeku su temporalna komponenta ove aplikacije, te kreiranjem ove tablice u bazi kao što je prikazano na početku rada automatski se kreira tablica o povijesti lijeka. Korisnica mora prvo unijeti podatke o lijeku. Ako su uneseni naziv, opis i cijena klikom na gumb Dodaj lijek oni se dodaju u tablicu trenutni podaci o lijeku. Podaci o vremenu vrijedi od i vrijedi do se automatski kreiraju u bazi jer su to kolone GENERATED ALWAYS. Vrijeme vrijedi od se odnosi na vrijeme kada je podatak unesen u bazu, odnosno kada je dodan lijek, a vrijeme vrijedi do je maksimalna vrijednost jer u trenutku dodavanja ovog podatka u tablicu taj lijek ima tu cijenu sve dok se ona ne promijeni. Također ako nisu unesena sva 3 podatka prilikom dodavanja lijeka javlja se poruka da svi podaci moraju biti uneseni. Nadalje, nakon što imamo određeni lijek u tablici mi možemo pratiti kretanje cijene tog lijeka kroz vrijeme. To se radi tako da u slučaju da lijek ima novu cijenu u polje nova cijena možemo upisati tu cijenu i kliknuti na gumb dodaj novu cijenu. Ako kliknemo na gumb bez da smo unijeli novu cijenu javlja se poruka da nova cijena mora biti unesena. Nakon što su uneseni podaci o novoj cijeni tu novu cijenu za odabrani lijek sada vidimo u trenutnim podacima o lijeku (gornja tablica), a podaci koji su bili vezani za taj lijek od trenutka kada smo ga mi unijeli u sustav do trenutka promjene cijene su upisani u tablicu povijesni podaci o lijeku. U tablici povijesni podaci o lijeku vrijeme vrijedi od se odnosi na vrijeme kada je lijek unesen u sustav, a vrijeme vrijedi do nam govori do kojeg trenutka je taj lijek imao prethodnu cijenu (prije nego li smo je mi promijenili). Tako možemo pratiti cijenu lijeka od prošlosti do sadašnjosti. Također ako nas neki lijek više ne interesira (ne planiramo ga koristiti) isti možemo obrisati te se onda također u tablici povijesni podaci o lijeku upisuje red koji označava zadnju aktivnu cijenu lijeka od kad do kad.

Zadnja komponenta su Rekreatije (14). Korisnica unosi tip rekreacije s kojim se želi baviti (kratka šetnja, trčanje...). Zatim odabire tu rekreaciju i klikom na gumb kreni u šetnju rekreacija kreće. Kada rekreacija završi korisnica unosi podatke o trajanju rekreacije i klikom na gumb završava rekreaciju.



## 6. Prikaz aplikacije - slike



TrudnocaApp

Dobrodošli na TrudnocaApp

Novi ste na TrudnocaApp?

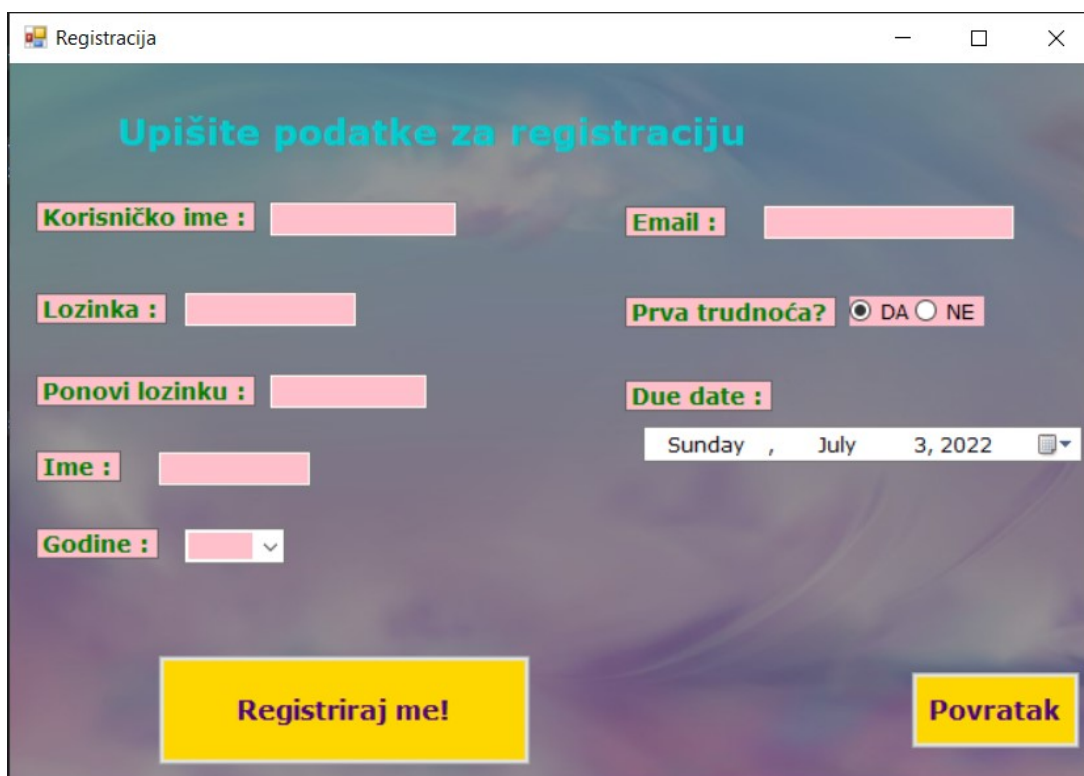
Registracija

Korisničko ime :

Lozinka :

Prijava

Slika 3: Pocetna forma



Registracija

Upišite podatke za registraciju

Korisničko ime :

Email :

Lozinka :

Ponovi lozinku :

Ime :

Godine :

Prva trudnoća? ☒ DA ☐ NE

Due date : Sunday , July 3, 2022

Registriraj me!

Povratak

Slika 4: Registracija forma

Registracija

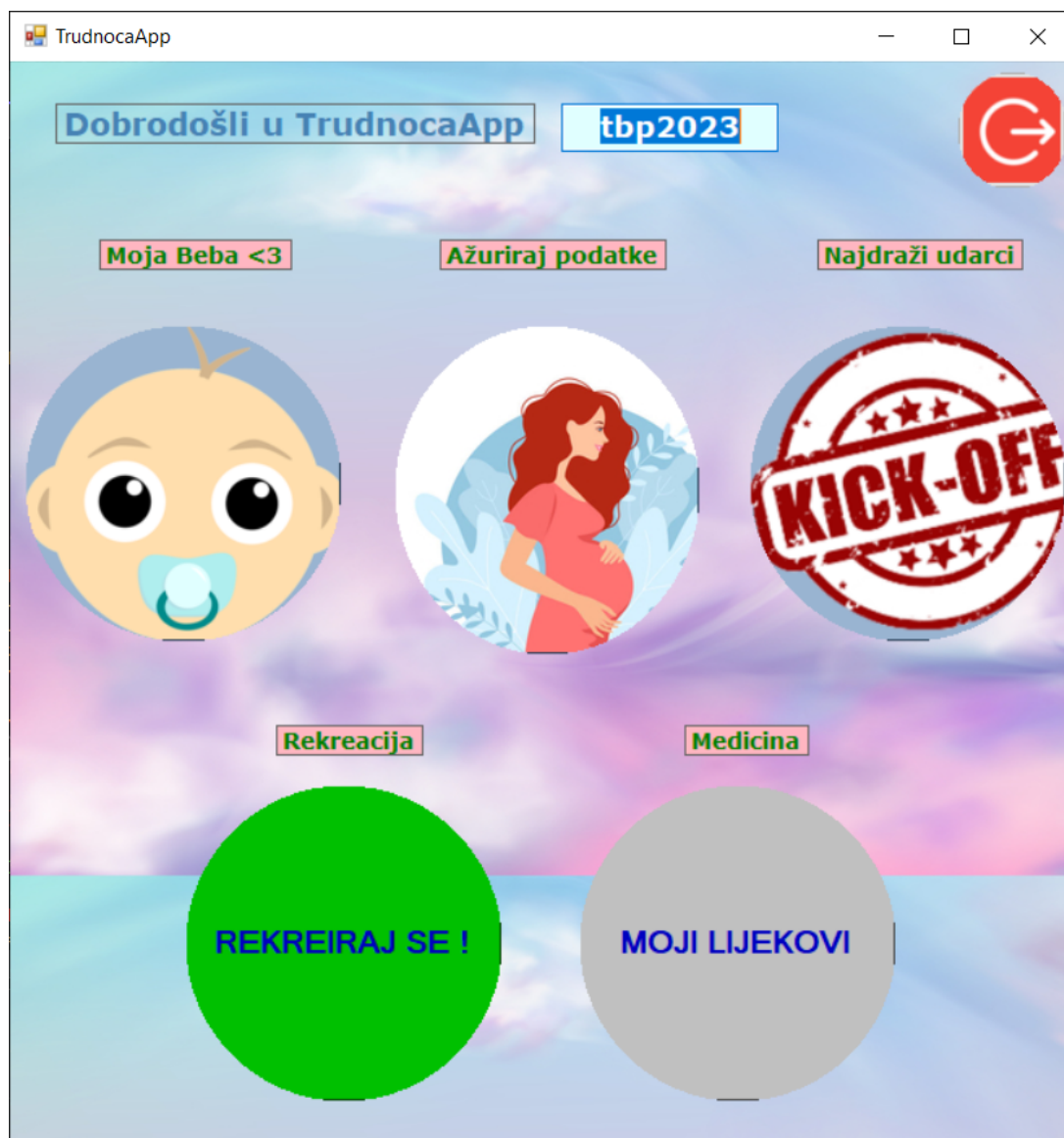
## Upišite podatke za registraciju

<b>Korisničko ime :</b> <input type="text"/>	<b>Email :</b> <input type="text"/>
<i>Korisničko ime ne smije biti prazno!</i>	<i>Email nije u ispravnom formatu!</i>
<b>Lozinka :</b> <input type="password"/>	<b>Prva trudnoća?</b> <input checked="" type="radio"/> DA <input type="radio"/> NE
<i>Lozinka ne smije biti prazna!</i>	
<b>Ponovi lozinku :</b> <input type="password"/>	<b>Due date :</b>
	Sunday , July 3, 2022
<b>Ime :</b> <input type="text"/>	
<i>Ime ne smije biti prazno!</i>	
<b>Godine :</b> <input type="text"/>	
<i>Odaberite godine!</i>	

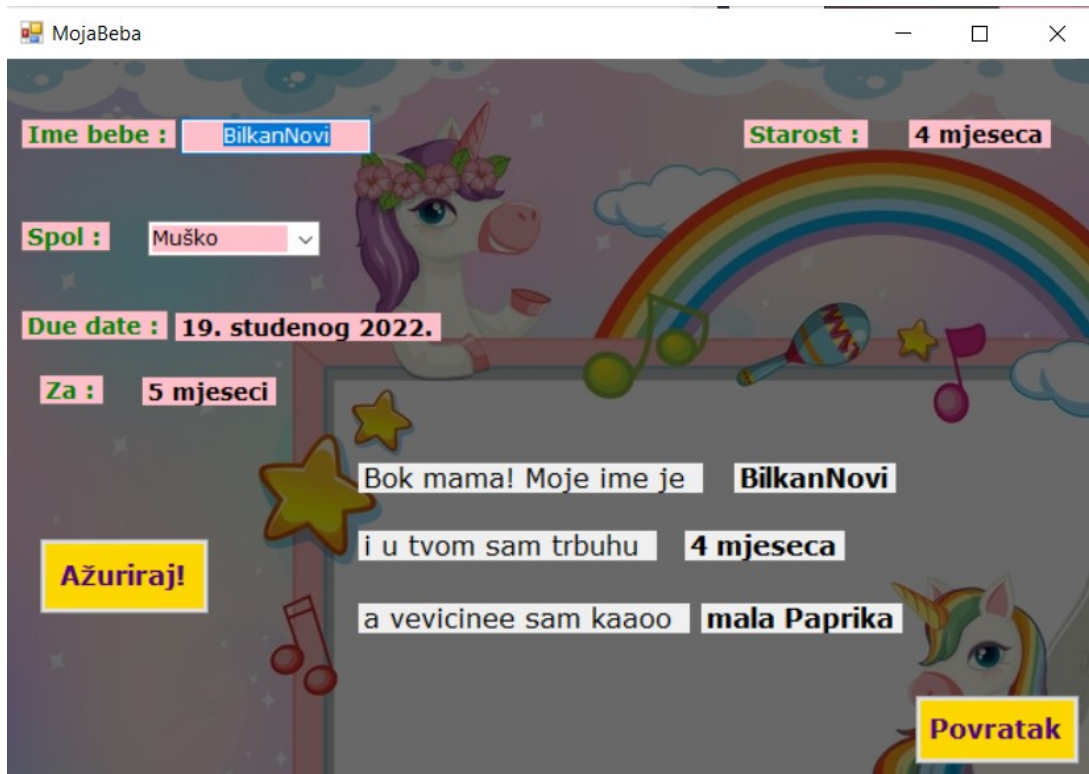
**Registriraj me!**

**Povratak**

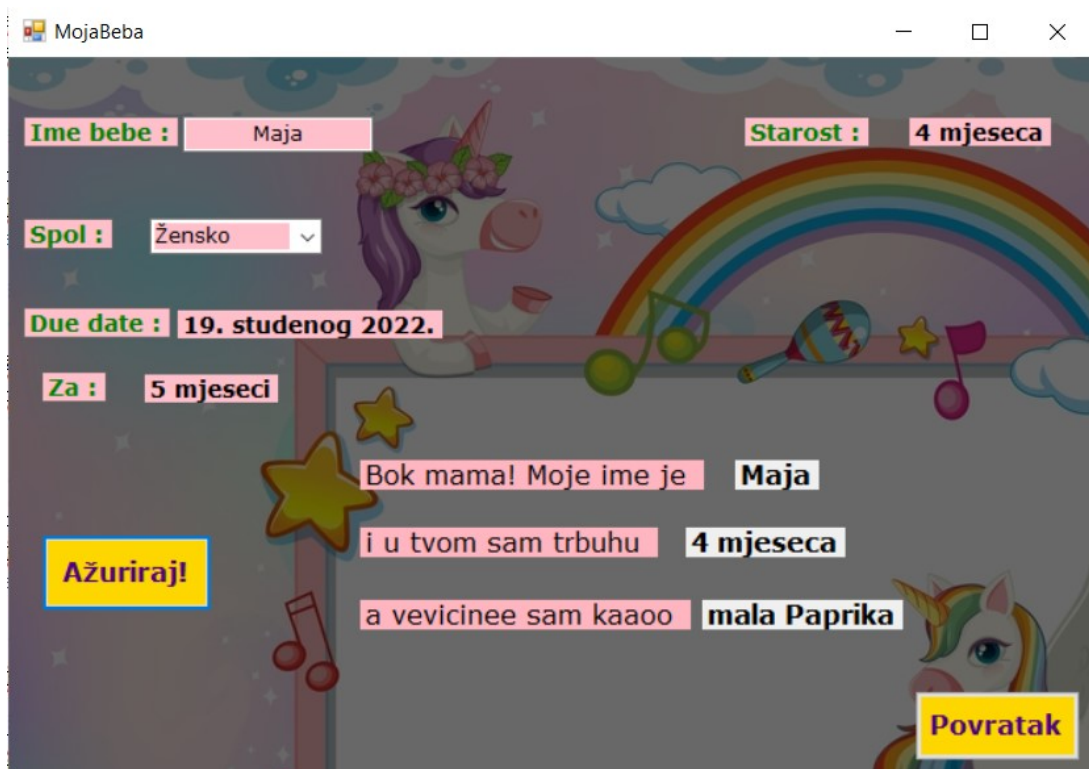
Slika 5: Registracija forma uvjeti



Slika 6: Glavna forma POBOLJŠANA verzija



Slika 7: Moja Beba forma



Slika 8: Moja Beba forma izmjene

Ažuriraj podatke

Korisničko ime :

Ime :  Email :

Godine :

Prva trudnoća? ☒ DA ☐ NE Due date :

Potvrdi lozinku :

**Ažuriraj!** **Povratak**

Slika 9: Azuriraj forma

Udarci

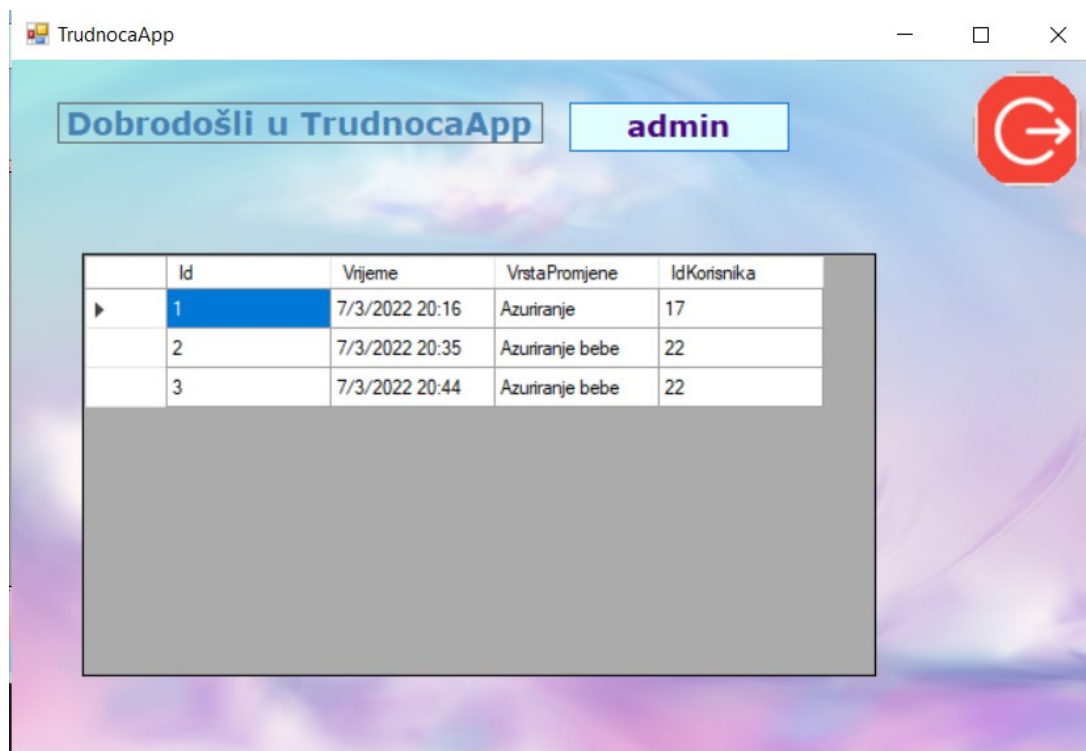
Trajanje sesije u sekundama :

	Id	DatumPocetak	DatumKraj	Udarci1
▶	23	7/3/2022 20:54	7/3/2022 20:55	2

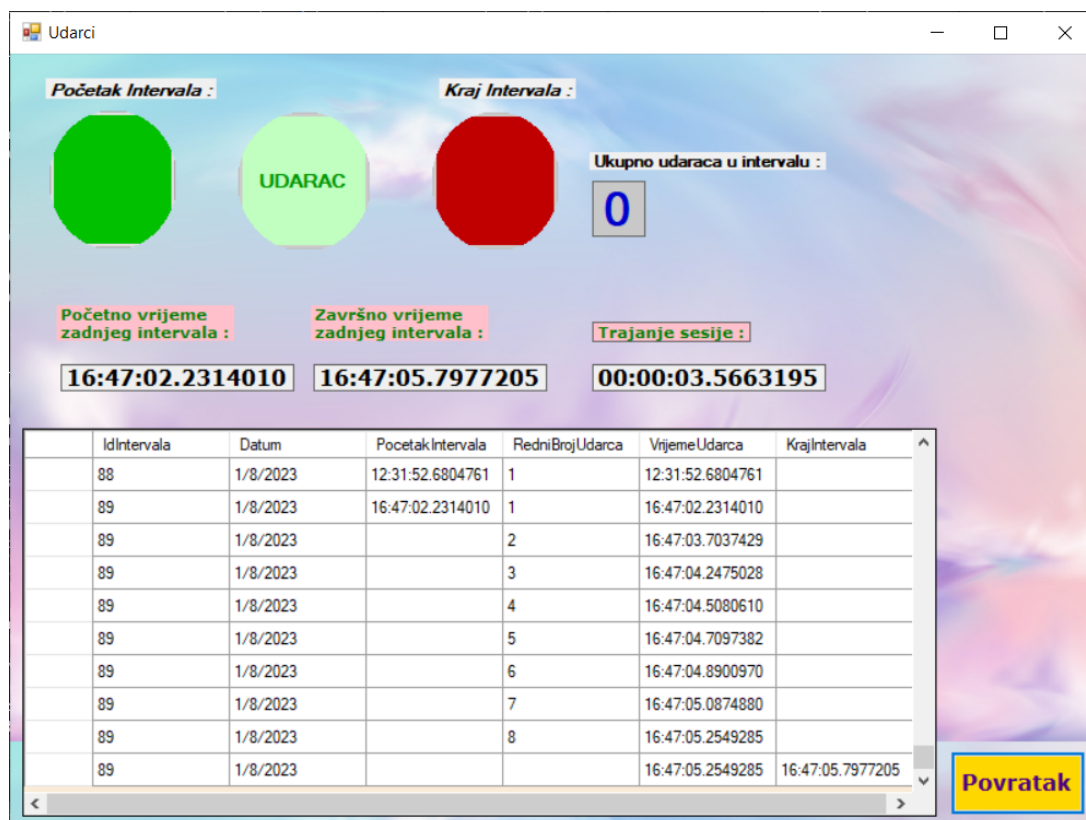
**Povratak**

Slika 10: Udarci forma STARA verzija





Slika 11: Admin forma



Slika 12: Udarci forma POBOLJŠANA verzija

Moji lijekovi

Dodajte podatke o lijeku : Naziv

Opis

Cijena

**DODAJ LIJEK**

**Trenutni podaci o lijeku**

	IdMedicine	Naziv	Opis	Cijena	VrijediOd	VrijediDo
▶	7	Ventolin v3	Dobar je jako	66.00	1/8/2023 15:03	12/31/9999 23:59
	8	Flixotide v2	ide protiv kaslja	77.00	1/8/2023 15:05	12/31/9999 23:59
	9	Flixotide v54	astma	66.00	1/8/2023 15:07	12/31/9999 23:59

Trenutna Cijena

**Obrisi**

**Povijesni podaci o lijeku** Nova Cijena

**DODAJ NOVU CIJENU**

	IdMedicine	Naziv	Opis	Cijena	VrijediOd	VrijediDo
▶	3	Vitamin C	Dobar je za sve	150.00	1/8/2023 14:12	1/8/2023 14:27
	3	Vitamin C	Dobar je za sve	175.00	1/8/2023 14:27	1/8/2023 14:29
	3	Vitamin C	Dobar je za sve	200.00	1/8/2023 14:29	1/8/2023 14:44
	5	Ventolin	astma	200.00	1/8/2023 14:56	1/8/2023 14:56
	4	Flixotide	Dobar za astmu	150.00	1/8/2023 14:30	1/8/2023 14:57
	5	Ventolin	astma	202.00	1/8/2023 14:56	1/8/2023 14:57
	9	Flixotide v54	astma	55.00	1/8/2023 15:06	1/8/2023 15:07

**Povratak**

Slika 13: Medicina forma NOVO

Rekreacija

Dodaj novu rekreaciju :

**DODAJ**

**Povratak**

	IdSetnja	Opis	Setnja	KodKuce	SysStartTime	SysEndTime
▶	1	duga setnja	25	<input checked="" type="checkbox"/>	1/8/2023 18:26	12/31/9999 23:59

Trenutni naziv rekreacije :

**KRENI U ŠETNJU**

	IdSetnja	Opis	Setnja	KodKuce	SysStartTime	SysEndTime
▶	1	duga setnja	0	<input checked="" type="checkbox"/>	1/8/2023 18:26	1/8/2023 18:26
	1	duga setnja	0	<input type="checkbox"/>	1/8/2023 18:26	1/8/2023 18:26

Trajanje rekreacije (min) :

**ZAVRŠI ŠETNJU**

Slika 14: Rekreacija forma NOVO

## 7. Zaključak

U ovom radu kreirana je aplikacija za praćenje trudnoće. U aplikaciji se nalazi nekoliko zanimljivih funkcionalnosti, koje se naravno mogu još detaljnije izvesti, ali za potrebe ovog rada smatram da je napravljeno dovoljno posla. Mnogo je načina na koje trudnice mogu pratiti svoju trudnoću, a jedan od popularnijih u današnje vrijeme je preko samih aplikacija koje imaju ugrađene razne alate za pomoć trudnicama. Mogu zaključiti da je praktičnije raditi mobilnu aplikaciju za praćenje trudnoće, dok je ova napravljena u windows form okruženju. Kod aplikacije gdje se stalno šalju upiti na bazu podataka treba voditi brigu o mogućim ograničenjima koja se javljaju. Primjenom triggera mnogo je lakše ta ograničenja savladati da aplikacija radi bez poteškoća. U ovom radu koristili smo trigger prilikom dodavanja ili ažuriranja podataka u tablicu, kako bi određene podatke onda zapisali u neke nove tablice. Postoji mnogo elegantnijih načina za izvedbu ovakve aplikacije, ja sam pokazao jedan primjer i smatram da je aplikacija dobra za generalni prikaz potrebnih informacija trudnicama u 21. stoljeću.



# Popis literature

- [1] M. S. Mirko Maleković, *Teorija i primjena baza podataka*, Na dan : 02.07.2022.
- [2] TutorialTeacher, *What is LINQ?* <https://www.tutorialsteacher.com/linq/what-is-linq>, Na dan : 02.07.2022.
- [3] Freepik, *Image*, <https://www.freepik.com/vectors/baby-elephant>, Na dan : 02.07.2022.

# Popis slika

1.	Model . . . . .	3
2.	Medicina SQL server . . . . .	15
3.	Pocetna forma . . . . .	21
4.	Registracija forma . . . . .	21
5.	Registracija forma uvjeti . . . . .	22
6.	Glavna forma POBOLJŠANA verzija . . . . .	23
7.	Moja Beba forma . . . . .	24
8.	Moja Beba forma izmjene . . . . .	24
9.	Azuriraj forma . . . . .	25
10.	Udarci forma STARA verzija . . . . .	25
11.	Admin forma . . . . .	26
12.	Udarci forma POBOLJŠANA verzija . . . . .	26
13.	Medicina forma NOVO . . . . .	27
14.	Rekreacija forma NOVO . . . . .	27

## **Popis tablica**