SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž 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, srpanj 2022.

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.

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

Sadržaj

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

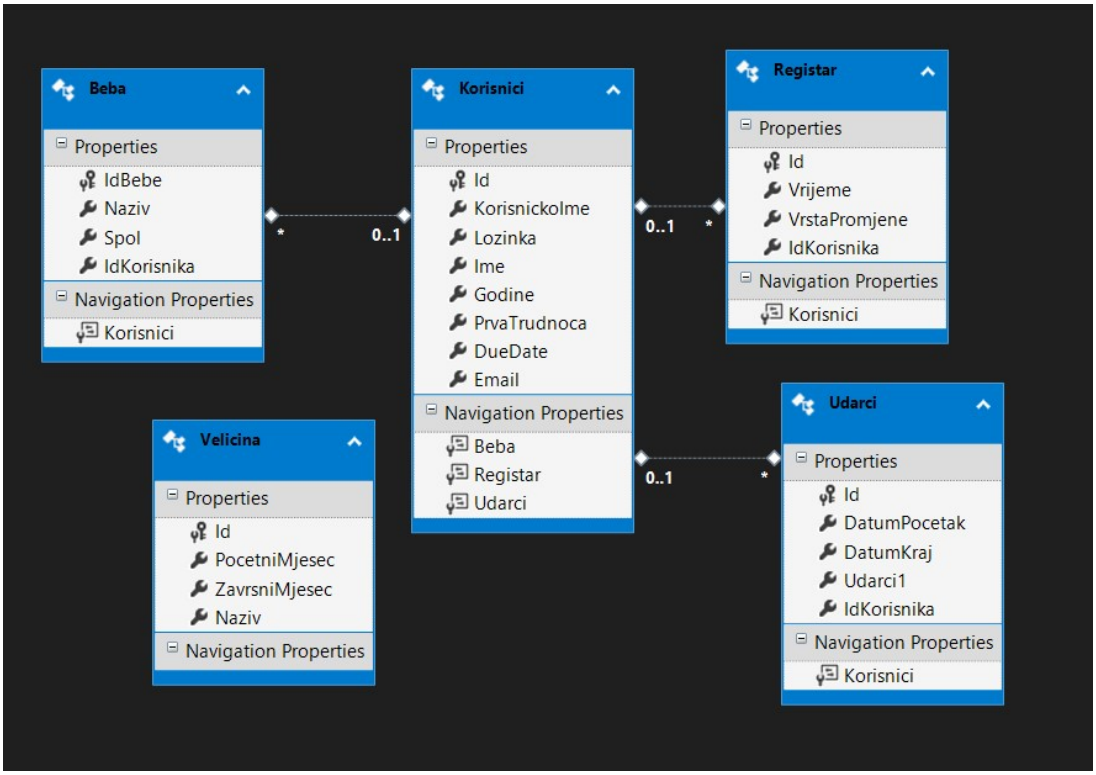
Temporalne baze podataka omogućuju uvođenje temporalne dimenzije, odnosno vremenske dimenzije, što znači da svaki objekt ima svoj životni ciklus. Životni ciklus objekta je moguće onda pratiti jer se može pohraniti prošlo, sadašnje i buduće vrijeme.[1]

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
```

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 prosljeđuje 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() == Ž"ensko")
                        {
                            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();
                }
            }
        }
```

Zadnja 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();
            }
        }
```

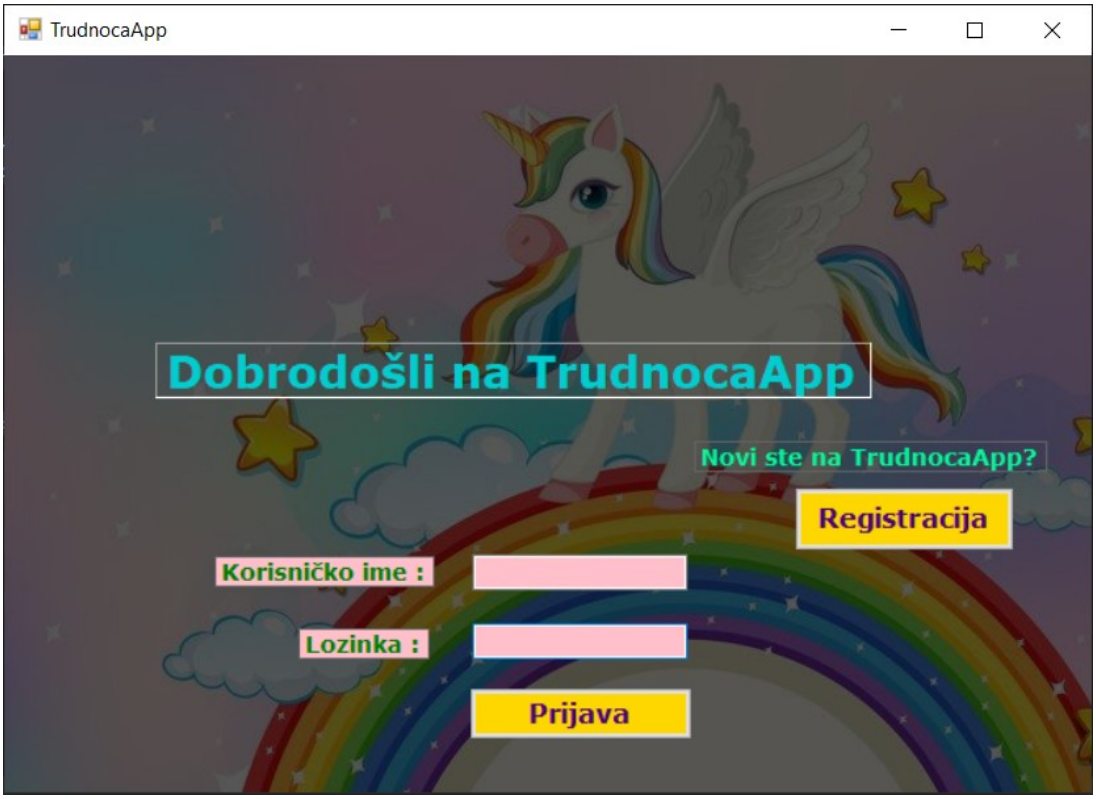
Također prilikom mjerenja udaraca kada korisnik klikne na gumb Stop, podaci se dodaju u tablicu.

```
private void btnStop_Click(object sender, EventArgs e)
        {
            vrijemeKraj = DateTime.Now;

            var razlikaVremena = vrijemeKraj.Subtract(vrijemePocetak);
            labelRazlikaVremena.Text = razlikaVremena.Seconds.ToString();
            labelRazlikaVremena.Visible = true;
            labelTrajanjeSesije.Visible = true;

            Udarci udarci = new Udarci()
            {
                DatumPocetak = vrijemePocetak,
                DatumKraj = vrijemeKraj,
                Udarci1 = brojUdaraca,
                IdKorisnika = korisnikovId
            };
            using(var context = new TrudnocaAppEntities())
            {
                context.Udarci.Add(udarci);
                context.SaveChanges();
            }

            UcitajDGV();
```

```
            btnStop.Enabled = false;
            labelBrojUdaraca.Text = "0";
            brojUdaraca = 0;
        }
```

5. Primjeri korištenja

U ovom poglavlju prikaza ćemo kako aplikacija izgleda(2) te objasniti svaki zaslon posebno. Prilikom otvaranja aplikacije prikazuje se početna forma.[3]



Slika 2: Pocetna forma

Korisnik se prvo mora registrirati(3), 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(4).

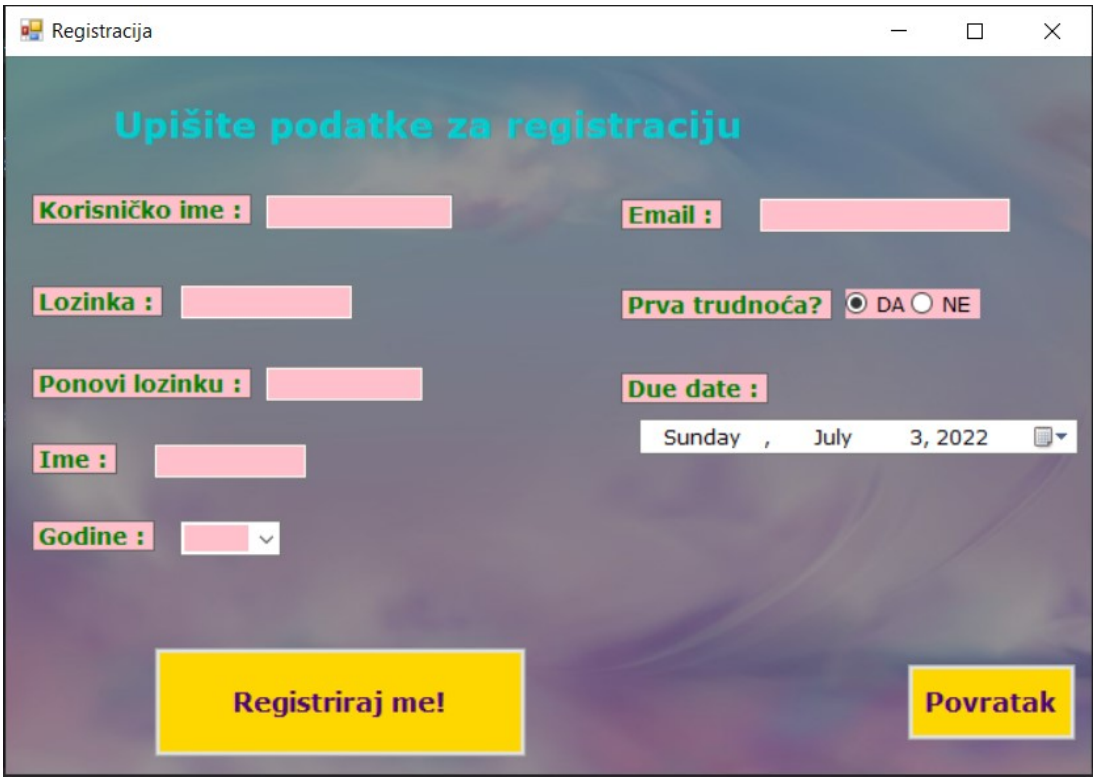
Ako je registracija uspješna korisnik je preusmjeren na početnu stranicu gdje ako unese točne podatke za prijavu je preusmjeren u glavnu formu aplikacije(5) gdje postoje opcije da korisnik vidi podatke o bebi (Moja Beba), ažurira svoje korisničke podatke, broji udarce bebe ili se odjavi.

Klikom na Moja beba otvara se forma(6).

Gdje korisnik može postaviti ime bebe ako se prvi put prijavio, ili izmjeniti(7) ime i spol ako je već prethodno odabrao.

Korisnik može promijeniti(8) svoje podatke ako to želi, uz uvjet da će opet doći do provjera valjanosti unosa.

Na kraju korisnik može i mjeriti udarce(9) koje beba napravi u određenom vremenskom

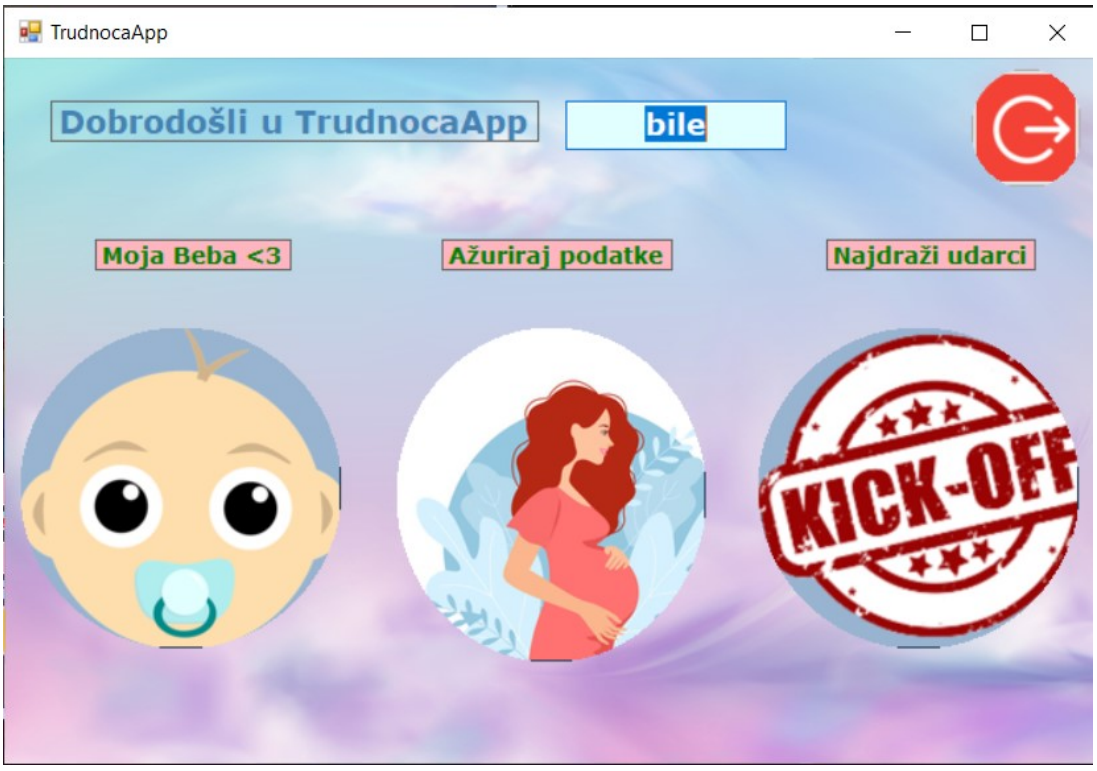


Slika 3: Registracija forma

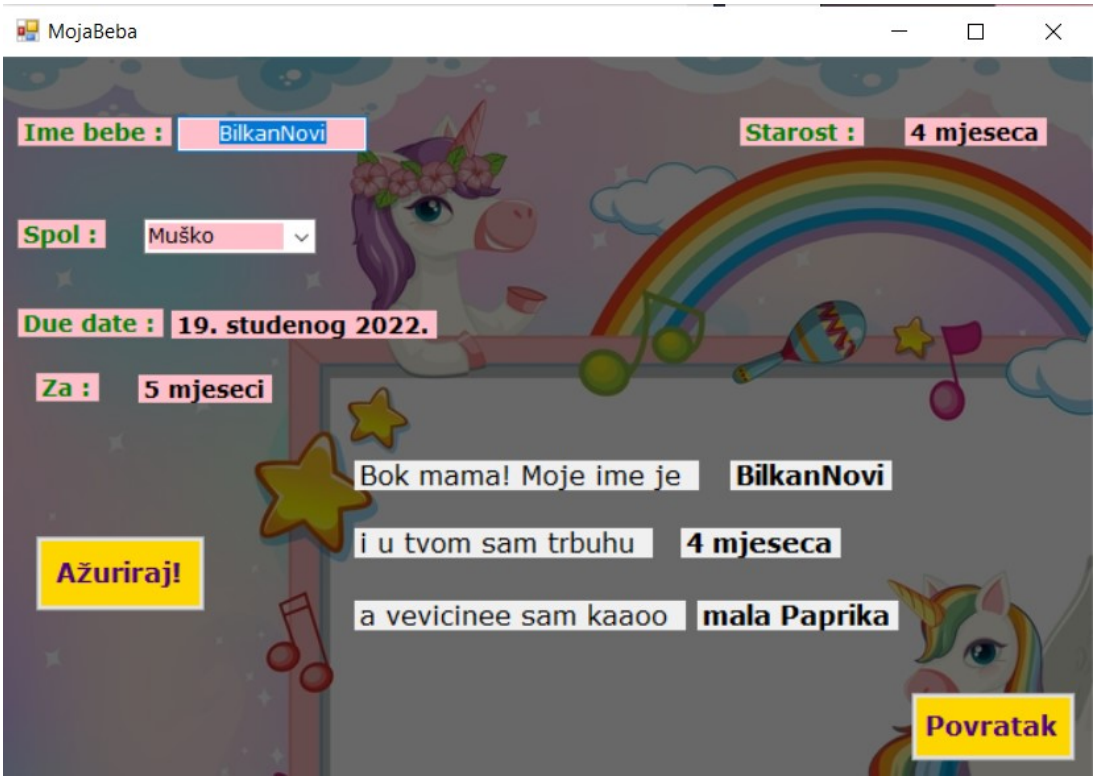
Slika 4: Registracija forma uvjeti

intervalu, a oni se prikazuju tablično.

Također ako se admin(10) prijavi u stavu njemu se učitava tablica Registar, gdje su



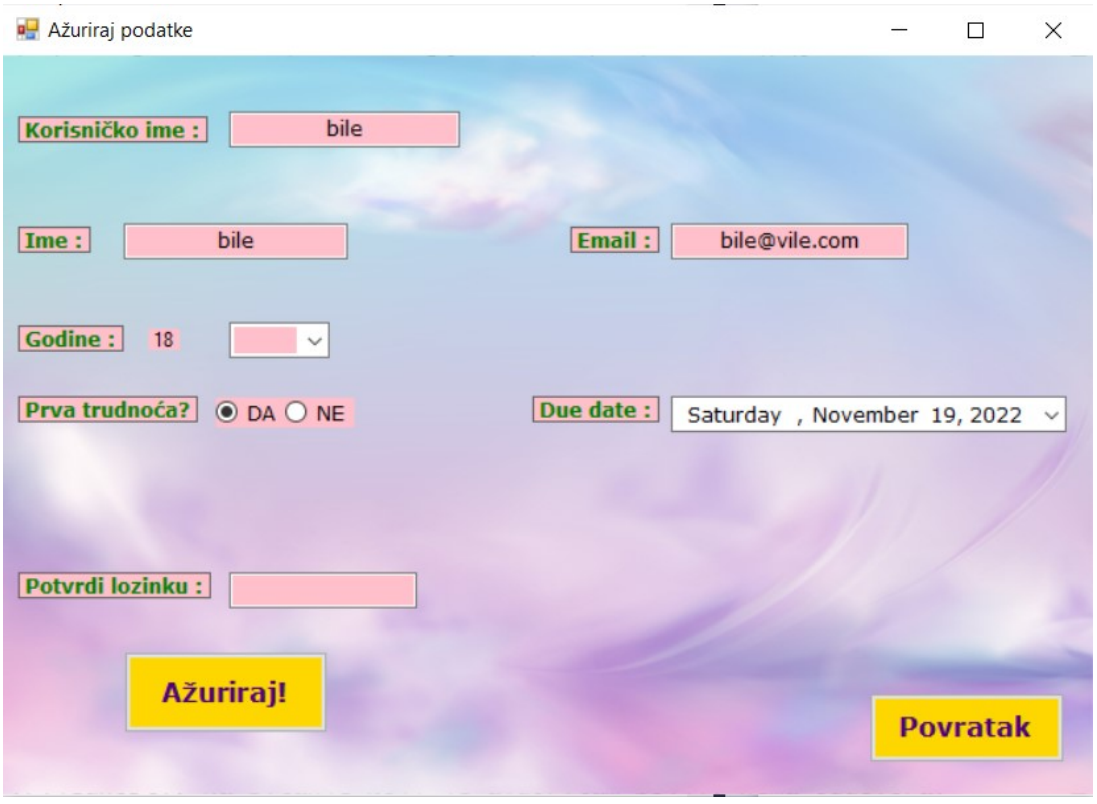
Slika 5: Glavna forma



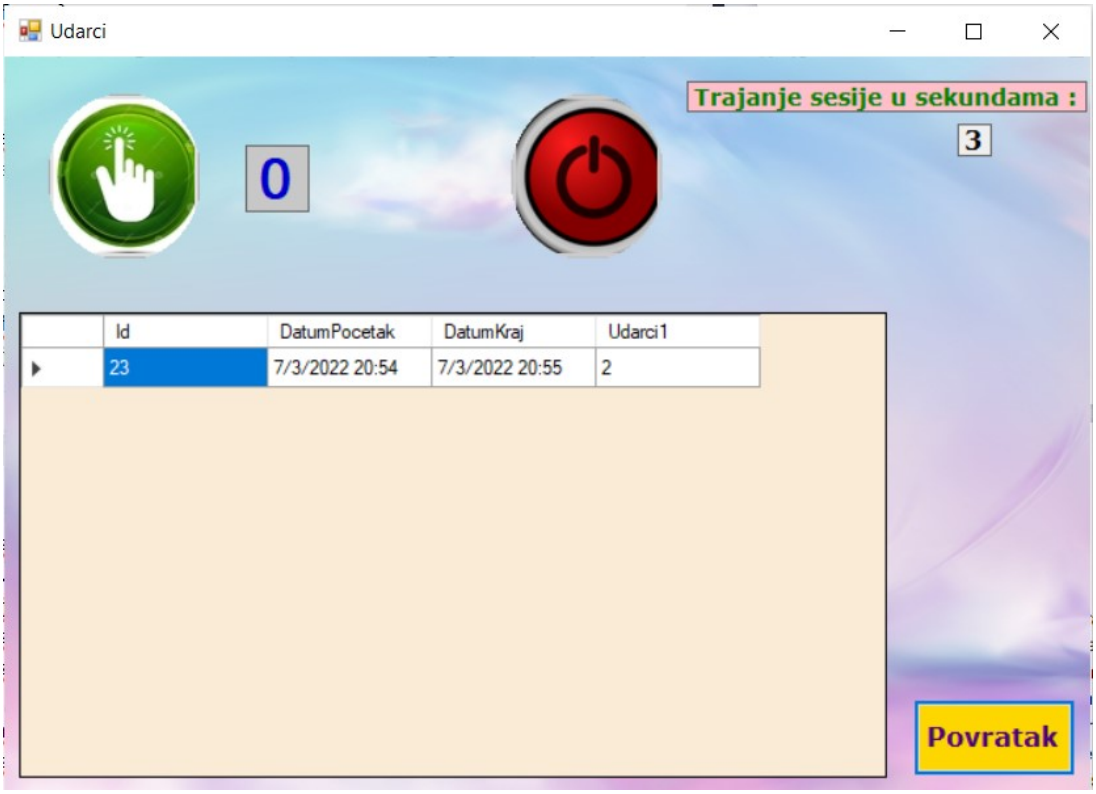
Slika 6: Moja Beba forma

zabilježene sve promjene koje su se dogodile u sustavu, a samo upisivanje u registar izvršavaju triggeri.

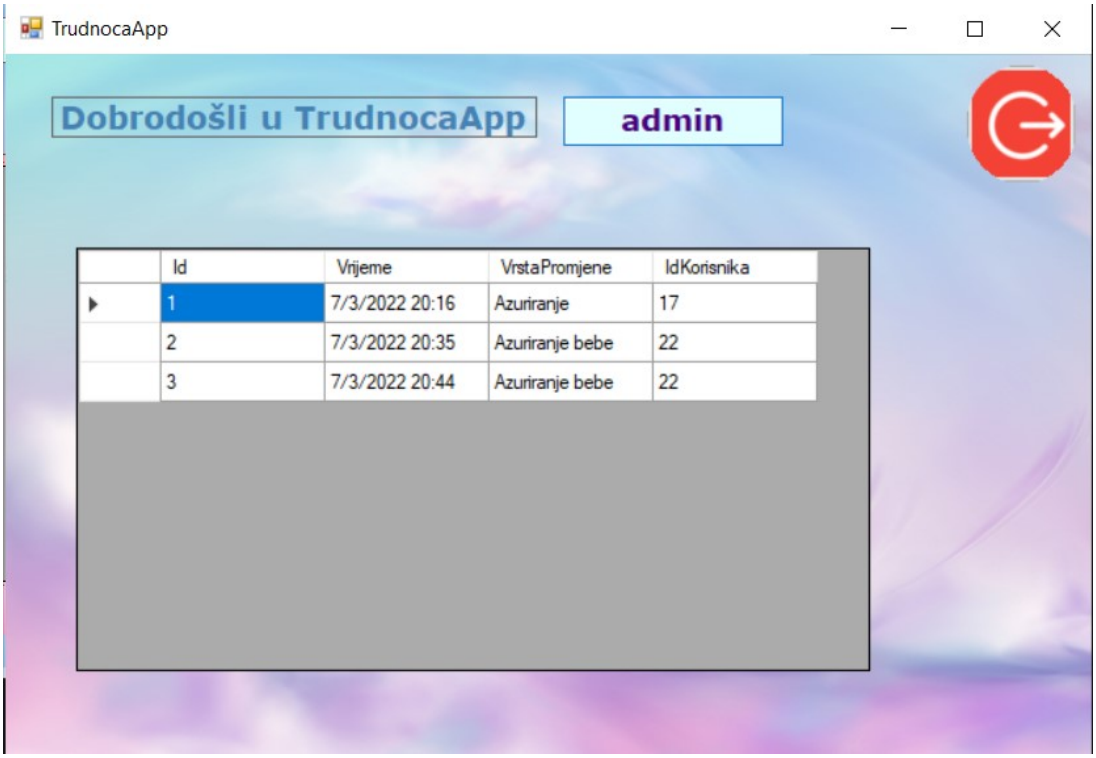
Slika 7: Moja Beba forma izmjene



Slika 8: Azuriraj forma



Slika 9: Udarci forma



Slika 10: Admin forma

6. 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 triggere 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] TutorialsTeacher, *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

Popis tablica