

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Ivan Pavlić

APLIKACIJA ZA PRAĆENJE RAZVOJA TRUDNOĆE

PROJEKT

Varaždin, 2020.

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

Ivan Pavlić

Matični broj: 44030/15–R

Studij: Organizacija poslovnih procesa

APLIKACIJA ZA PRAĆENJE RAZVOJA TRUDNOĆE

PROJEKT

Mentor/Mentorica:

Dr. sc. Bogdan Okreša Đurić

Varaždin, Ožujak 2020.

Izjava o izvornosti

Izjavljujem da je moj projekt 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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema mog projektnog zadatka za potrebe kolegija Teorija baza podataka je aplikacija za praćenje trudnoće. Napravljenja je aplikacija koja omogućuje doktoru odnosno korisniku pregled pacijenta odnosno trudnica te naručivanje trudnica na redoviti pregled. Pri izradi aplikacije osnovni fokus bio je na izradi baze podataka primjenom aktivnih i temporalnih komponenata. Za izradu ovog projekta koristio sam PostgreSQL, Navicat 12 for PostgreSQL. U Navicatu 12 for PostgreSQL kreirao sam sve potrebne tablice, veze između tablica te okidače. Grafičko sučelje aplikacija izradio sam u razvojnom okruženju Visual Studio u obliku Windows Forms aplikacije u .NET Frameworku C. U bazi se nalazi 7 tablica i 2 okidača. Aplikacija je namijenjena doktorima za pregleda trudnoća kod svojih pacijentica, naručivanje pacijentica, vođenje informacija o tijeku trudnoće kod pacijentica.

Ključne riječi: baze podataka, aktivne baze podataka, temporalne baze podataka, PostgreSQL, Navicat, okidači, .NET Framework

Sadržaj

1. Opis aplikacijske domene	1
2. Teorijski uvod	2
2.1. Navicat	2
2.2. PostgreSQL	2
2.3. Aktivne baze podataka	2
2.4. Temporalne baze podataka	2
3. Model baze podataka	3
3.1. ERA model	3
4. Okidači	5
4.1. Rezervacija termina pregleda	5
4.2. Radno vrijeme	6
5. Implementacija aplikacije	7
5.1. Povezivanje s bazom podataka	7
5.2. Klasa Doktor	8
6. Korištenje aplikacije	10
7. Zaključak	15
Popis literature	16
Popis slika	17

1. Opis aplikacijske domene

Tema ovog projektnog zadatka je izrada aplikacije za praćenje trudnoće. Aplikacija osmišljena na način da pristup ima doktor odnosno korisnik koji ima mogućnost dodavanja, brisanja i izmjena svojih pacijentica/trudnica, dodavanja podataka o trudnoći za svaku pacijenticu posebno. Isto tako pomoću okidača osmišljeno je naručivanje pacijentica na redovan pregled tokom njihove trudnoće. Fokus izrade ove aplikacije je na izradi baze podataka korištenjem kombinacije aktivnih i temporalnih baza podataka.

Baza podataka za ovu aplikaciju izrađena je u Navicat 12 for PostgreSQL. Ovaj alat sam izabrao iz razloga što smo se sa njim upoznao na ranijim kolegijima na preddiplomskom studiju. Grafičko sučelje aplikacije izrađeno je alatu Visual Studio u obliku Windows form aplikacije koja je pisana u .NET Frameworku u jeziku C#. Osnovni razlog što sam odabrao ove alate je već spomenuto predznanje sa prethodnih kolegija, ali isto tako jer imaju dobru podršku.

Rad je podijeljen u nekoliko cjelina kako bi se lakše shvatio cijeli princip izrade same aplikacije, ali isto tako kako bi se objasnilo korak po korak kako se aplikacija koristi te koje su njezine mogućnosti u smislu praćenja trudnoće od strane doktora za svoje pacijentice odnosno trudnice.

2. Teorijski uvod

2.1. Navicat

Za izradu same baze podataka, što je i osnova ovog projekta, koristio sam Navicat for PostgreSQL. Navicat je alat s grafičkim korisničkim sučeljem koji služi za kreiranje i administriranje baza podataka. Može se spajati na MySQL, Oracle, PostgreSQL, SQLite, SQL Server ili MariaDB baze podataka.[1] Navicat je dostupan za razne operacijske sustave pa je tako dostupan za Microsoft Windows, Mac OS X te Linux platformi.

2.2. PostgreSQL

PostgreSQL je sustav za upravljanje objektno relacijskim bazama podataka. Objektno relacijska baza podataka je baza podataka koja podatke sprema u odvojene tablice, što daje brzinu i fleksibilnost bazama podataka. Tablice su povezane definiranim relacijama što omogućuje kombiniranje podataka iz nekoliko tablica.[2]

2.3. Aktivne baze podataka

Aktivne baze podataka se temelje na bazi pravila koja se najčešće sastoje od događaja i akcija (engl. Event Condition Action – ECA). Pojavom događaja provjerava se niz uvjeta i ukoliko jedan zadovoljava, obavlja se popratna akcija. Ovo svojstvo aktivnih baza podataka omogućuje autonomnost prilikom rada i uklanjanja potrebu za npr. učestalim provjerama stanja nekih podataka u bazi podataka.[3] Aktivne baze podataka, tj. baze podataka sa podrškom definiranja ECA pravila predstavljaju prirodnu evoluciju relacijskog modela na što ukazuju činjenice da SQL:1999 standard definira okidače (engl. triggers).

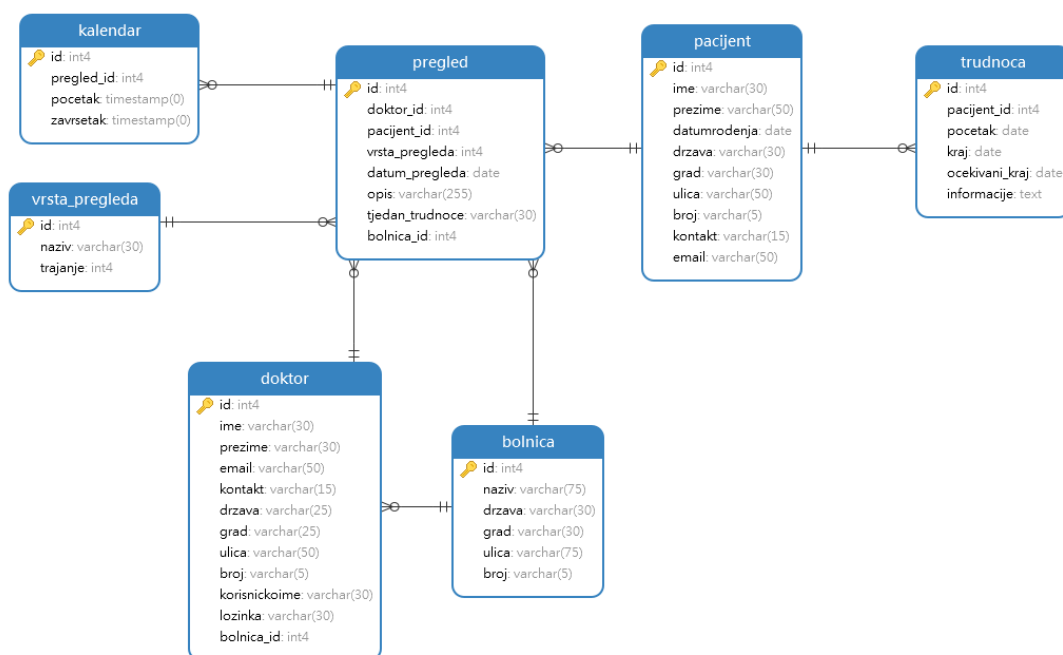
2.4. Temporalne baze podataka

Temporalne baze podataka se od ostalih baza podataka razlikuju po dimenziji vremena kroz koju su svi podaci u bazi podataka zapisani. Isto tako, temporalne baze podataka nisu vezane samo za relacijski tip baza podataka, nego se mogu koristiti i nerelacijski tipovi.[4] Dimenzije vremena koje se koriste kod ovog koncepta su date, time, timestamp te interval. U projektu se koristi PostgreSQL pa treba napomenuti da su u njemu podržani aspekti temporalnih baza podataka kao što su vremenski tipovi i operacije za rad nad njima.[1]

3. Model baze podataka

U ovom poglavlju prikazan je i opisan ERA model na kojem je temeljen rad cijele aplikacije. Sam model izrađen je u alatu Navicat kojeg smo upoznali na ranijim kolegijima. Alat omogućuje definiranje primarnih i vanjskih ključeva te samim time definiranje veza između tablica, također mogu se kreirati okidači te još mnoge mogućnosti koje nam pruža taj alat.

3.1. ERA model



Slika 1: ERA model (Izvor: Vlastita izrada)

U **ERA modelu** prikazanu se tablice koje se koriste u radu aplikacije, a to su:

- Doktor
- Pregled
- Pacijent
- Trudnoca
- Bolnica
- Kalendar
- Vrsta-pregleda

Tablica **Doktor** sastoji se od dvanaest atributa. Od tih dvanaest atributa jedan je vanjski ključ na tablicu **Bolnica**. Važno je napomenuti da se u toj tablici nalaze atributi koji su potrebni za

prijavu u aplikaciju pošto je aplikacija zamišljena kao praćenje trudnoće pacijenata od strane doktora.

Sljedeća tablica u modelu je **Pacijent** koja se sastoji od deset atributa. Zatim imamo tablicu **Pregled** koja ima značajnu ulogu u radu aplikacije. Tablica Pregled ima vanjske ključeve na tablice Doktor, Pacijent i Vrsta-pregleda. Sve tri veze su definirane kao 1:N, odnosno jedan na-prema više. Tablica **Kalendar** sastoji se od četiri atributa te je vanjskim ključem povezana na tablicu Pregled. Temeljem datuma pregleda koji se unosi u tablicu Pregled, automatski se popuni timestamp pocetak i zavrsetak termina u tablici Kalendar. Tablica **Vrsta-pregleda** sastoji se od tri atributa, a njezina funkcionalnost da definira sam opseg odnosno trajanje pregleda.

Zadnja tablica koja se koristi u radu aplikacije je tablica **Trudnoca**. Tablica se sastoji od šest atributa od čega je jedan vanjski ključ na tablicu Pacijenta. Svaki pacijent može imati više trudnoća te se u njima nalaze osnovne informacije o svakoj pojedinoj trudnoći za svakog pacijenta posebno.

4. Okidači

U ranijim poglavljima spomenuto je kako je jedna od prednosti korištenja aktivnih baza podataka mogućnost kreiranja okidača (triggera). Kada se izvrši neka operacija unutar baze nad tablicom ili pregledom, okidač omogućuje da se određena funkcija u tom trenutku automatski izvrši. U nastavku su prikazana dva okidača koja su korišteni u ovoj aplikaciji.

4.1. Rezervacija termina pregleda

```
CREATE OR REPLACE FUNCTION "public"."pregled"()
RETURNS "pg_catalog"."trigger" AS $BODY$

DECLARE datum DATE;
DECLARE termin TIMESTAMP;
DECLARE trajanjePregleda INT;
DECLARE dan INTEGER;

BEGIN
SELECT DATE(NEW."datum_pregleda") INTO datum;
SELECT "trajanje" into trajanjePregleda FROM "vrsta_pregleda" WHERE NEW."vrsta_pregleda" = "vrsta_pregleda"."id";
SELECT "zavrsetak" INTO termin FROM "kalendar" WHERE "zavrsetak" BETWEEN datum AND CAST(datum || ' ' || '21:00:00' AS TIMESTAMP) ORDER BY
"zavrsetak" DESC LIMIT 1;
SELECT EXTRACT(DAY FROM NEW."datum_pregleda") INTO dan;

IF (dan % 2 = 0) THEN
IF (termin IS NULL) THEN
INSERT INTO "kalendar" ("pregled_id", "pocetak", "zavrsetak") VALUES (new."id", CAST(datum || ' ' || '07:00:00' AS TIMESTAMP), CAST(datum || '
' || '07:00:00' AS TIMESTAMP) + trajanjePregleda * '1 minute'::interval);
ELSE
INSERT INTO "kalendar" ("pregled_id", "pocetak", "zavrsetak") VALUES (new."id", termin, termin + trajanjePregleda * '1 minute'::interval);
END IF;
ELSE
IF (termin IS NULL) THEN
INSERT INTO "kalendar" ("pregled_id", "pocetak", "zavrsetak") VALUES (new."id", CAST(datum || ' ' || '14:00:00' AS TIMESTAMP), CAST(datum || '
' || '14:00:00' AS TIMESTAMP) + trajanjePregleda * '1 minute'::interval);
ELSE
INSERT INTO "kalendar" ("pregled_id", "pocetak", "zavrsetak") VALUES (new."id", termin, termin + trajanjePregleda * '1 minute'::interval);
END IF;
END IF;
-- Routine body goes here...

RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
```

Slika 2: Okidač za rezervaciju termina pregleda (Izvor: Vlastita izrada)

Ovaj okidač omogućuje rezerviranje prvog slobodnog termina pregleda kada korisnik (doktor) unese željeni datum pregleda za određenog pacijenta. Važno je napomenuti da se rezervira prvi slobodni termin za doktora koji vrši rezervaciju. Okidač na samom početku provjerava je li varijabla dan parna ili neparna. Parni dani predstavljaju radno vrijeme prijepodne (07:00 – 14:00), a neparni dan poslijepodne (14:00 – 20:00). Nakon toga okidač provjerava zauzetost termina pregleda. Ako je termin null, rezervirat će se pregled od početka radnog vremena određenog doktora uvećano za trajanje samog pregleda. U slučaju kada termin nije null, tada se rezervira termin od završetka zadnjeg termina pregleda tok dana pa do tog vremena također uvećano za trajanje pregleda.

4.2. Radno vrijeme

```
CREATE OR REPLACE FUNCTION "public"."radno_vrijeme"()
  RETURNS "pg_catalog"."trigger" AS $BODY$

  DECLARE dan INTEGER;
  DECLARE datum DATE;

  BEGIN
    SELECT EXTRACT(DAY FROM NEW."pocetak") INTO dan;
    SELECT DATE(NEW."zavrsetak") INTO datum;
    IF (dan % 2=0) THEN
      IF (NEW."zavrsetak" >= CAST(datum || ' ' || '14:00:00' AS TIMESTAMP)) THEN
        RAISE EXCEPTION 'Termin se ne može rezervirati jer je radno vrijeme do 14:00!';
        RETURN NULL;
      ELSE
        RETURN NEW;
      END IF;
    ELSE
      IF (NEW."zavrsetak" >= CAST(datum || ' ' || '20:00:00' AS TIMESTAMP)) THEN
        RAISE EXCEPTION 'Termin se ne može rezervirati jer je radno vrijeme do 20:00!';
        RETURN NULL;
      ELSE
        RETURN NEW;
      END IF;
    END IF;
    -- Routine body goes here...
  END
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
```

Slika 3: Okidač za provjeru radnog vremena (Izvor: Vlastita izrada)

Ovaj okidač se nadovezuje na prethodno opisani okidač za rezervaciju termina pregleda. U tablicu Kalendar upisuje se početak i završetak termina nakon što se rezervira pregled. Varijabla datum predstavlja datum rezerviranog pregleda dok varijabla dan predstavlja dan iz tog datuma. Ukoliko je dan paran, doktor radi do 14:00, a ako je dan neparan, tada doktor radi do 20:00. U slučaju kada je dan paran, ako je završetak termina koji se unese u tablicu Kalendar veći od 14:00 ispisat će se greška. Ako je rezervirani termin manji od 14:00 tada će se pregled rezervirati u ispravnom terminu. Isti situacija je kada je dan neparan. Ukoliko je završetak termina koji se unosi u tablicu Kalendar veći od 20:00 javit će se greška, no ako se rezervira termin koji je manji od 20:00, pregled će se ispravno rezervirati.

5. Implementacija aplikacije

Kao što je već ranije spomenuto, sama aplikacija je implementirana u Visual Studiu upotrebom jezika C. Projekt je podijeljen u nekoliko klasa od kojih je su najvažnije klasa Baza koja služi za povezivanje s bazom te Doktor u kojoj su implementirane metode za provjeru prilikom prijave u aplikaciju, dohvaćanje i brisanje zaposlenika. Ostale klase su napravljene po istom principu kao klasa Doktor.

5.1. Povezivanje s bazom podataka

```
6 using Npgsql;
7
8 namespace TrudnocaApp.Klase
9 {
10     18 references
11     class Baza
12     {
13         private static Baza instance; //Singleton objekt
14         private string connectionString; //Putanja i ostali podaci za spajanje na bazu
15         private NpgsqlConnection connection; //Konekcija prema bazi
16         14 references
17         public static Baza Instance
18         {
19             get
20             {
21                 if (instance == null)
22                 {
23                     instance = new Baza();
24                 }
25                 return instance;
26             }
27         }
28         2 references
29         public string ConnectionString
30         {
31             get { return connectionString; }
32             private set { connectionString = value; }
33         }
34         4 references
35         public NpgsqlConnection Connection
36         {
37             get { return connection; }
38             private set { connection = value; }
39         }
40         1 reference
41         private Baza() //Konstruktor klase
42         {
43             ConnectionString = "Server=localhost;Port=5432;User Id=postgres;Password=5432;Database=postgres";
44             Connection = new NpgsqlConnection(ConnectionString);
45             Connection.Open();
46         }
47         8 references
48         public NpgsqlDataReader DohvatiDataReader(string sqlUpit)
49         {
50             NpgsqlCommand command = new NpgsqlCommand(sqlUpit, Connection);
51             return command.ExecuteReader();
52         }
53         6 references
54         public int IzvrsiUpit(string sqlUpit)
55         {
56             NpgsqlCommand command = new NpgsqlCommand(sqlUpit, Connection);
57             return command.ExecuteNonQuery();
58         }
59     }
60 }
```

Slika 4: Klasa za povezivanje s bazom (Izvor:Vlastita izrada)

Na slici možemo vidjeti da se za spajanje na bazu koristio NpgSql pružatelja podataka za .NET. Za spajanje se koristi objekt Connection te se poziva NpgSqlConnection kojeg sadrži

ranije naveden pružatelj podataka Npgsql. Također možemo vidjeti da je u connectionString koji nam je potreban za spajanje na bazu, potrebno navesti IP ili ime servera, port na kojeg se želimo spojiti, imena korisnika, pripadne lozinke baze podataka kojom želimo raditi. Pomoću naredbe conn.Open() otvaramo konekciju prema bazi, a pomoću naredbe conn.Close() zatvaramo odnosno prekidamo vezu sa bazom.

5.2. Klasa Doktor

Na slici je prikazana klasa Doktor te su na taj način napravljene i ostale klase koje se koriste u radu aplikacije.

```
8 namespace TrudnocaApp.Klase
9 {
10     21 references
11     public class Doktor
12     {
13         3 references
14         public int Id { get; set; }
15         4 references
16         public string Ime { get; set; }
17         4 references
18         public string Prezime { get; set; }
19         2 references
20         public string Email { get; set; }
21         2 references
22         public string Kontakt { get; set; }
23         1 reference
24         public string Drzava { get; set; }
25         1 reference
26         public string Grad { get; set; }
27         1 reference
28         public string Ulica { get; set; }
29         1 reference
30         public string Broj { get; set; }
31         2 references
32         public string KorisnickoIme { get; set; }
33         1 reference
34         public string Lozinka { get; set; }
35         1 reference
36         public int Bolnica_Id { get; set; }
37         9 references
38         public static Doktor PrijavljeniDoktor { get; set; }
39         0 references
40         public string PunoIme
41         {
42             get
43             {
44                 return Ime + " " + Prezime;
45             }
46         }
47     }
48 }
```

Slika 5: Klasa doktor 1. dio (Izvor: Vlastita izrada)

U klasi su navedeni atributi koji se isto tako nalaze u tablici Doktor u ERA modelu. Osim atributa iz tablice Doktor, dodani su još i PrijavljeniDoktor i PunoIme. PrijavljeniDoktor nam služi za praćenje prijavljenog zaposlenika/doktora, a PunoIme vraća spojene attribute Ime i Prezime.

Na slici 6. su prikazane metode koje se koriste za rad s tablicom Doktor te konstruktor.

```
33 public Doktor()
34 {
35 }
36
37
38 2 references
39 public Doktor(NpgsqlDataReader dr)
40 {
41     if (dr != null)
42     {
43         Id = int.Parse(dr["id"].ToString());
44         Ime = dr["ime"].ToString();
45         Prezime = dr["prezime"].ToString();
46         Email = dr["email"].ToString();
47         Kontakt = dr["kontakt"].ToString();
48         Drzava = dr["drzava"].ToString();
49         Grad = dr["grad"].ToString();
50         Ulica = dr["ulica"].ToString();
51         Broj = dr["broj"].ToString();
52         KorisnickoIme = dr["korisnickoime"].ToString();
53         Lozinka = dr["lozinka"].ToString();
54         Bolnica_Id = int.Parse(dr["bolnica_id"].ToString());
55     }
56 }
57
58 1 reference
59 public static Doktor DohvatiDoktora(string korisnickoIme, string lozinka)
60 {
61     Doktor doktor = null;
62     string sqlUpit = "SELECT * FROM Doktor WHERE korisnickoime = '" + korisnickoIme + "' AND lozinka = '" + lozinka + "' LIMIT 1;";
63     NpgsqlDataReader dr = Klase.Baza.Instance.DohvatiDataReader(sqlUpit);
64     while (dr.Read())
65     {
66         doktor = new Doktor(dr);
67     }
68     dr.Close();
69     return doktor;
70 }
```

Slika 6: Klasa doktor 2. dio (Izvor: Vlastita izrada)

Metoda DohvatiDoktora koristi se za prijavu u aplikaciju. Prima parametre tipa string: korisnickoime i lozinka. Te parametre uspoređuje sa podacima u bazi te na taj način provjerava postoji li doktor sa tim korisničkim imenom i lozinkom u tablici Doktor.

```
0 references
public static List<Doktor> DohvatiSveDoktore()
{
    List<Doktor> doktori = new List<Doktor>();
    string upit = "SELECT * FROM Doktor ORDER BY id;";

    NpgsqlDataReader dr = Klase.Baza.Instance.DohvatiDataReader(upit);

    while (dr.Read())
    {
        Doktor z = new Doktor(dr);
        doktori.Add(z);
    }

    dr.Close();
    return doktori;
}

0 references
public int ObrisiDoktora(int id)
{
    string upit = "DELETE FROM Doktor WHERE id = " + id;
    return Klase.Baza.Instance.IzvršiUpit(upit);
}

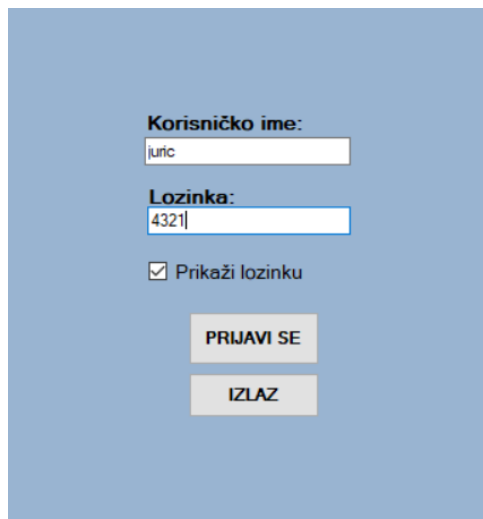
2 references
public override string ToString()
{
    return Ime + " " + Prezime;
}
}
```

Slika 7: Klasa doktor 3. dio (Izvor: Vlastita izrada)

Na slici 7. prikazane su metode koje služe za dohvaćanje svih doktora, brisanje doktora te metoda za vraćanje spojenih atributa ime i prezime.

6. Korištenje aplikacije

U ovom ćemo poglavlju opisati kako funkcionira aplikacija. Prilikom pokretanja aplikacija otvara se porozor za prijavu. Potrebno je unijeti korisničko ime i lozinku.



Slika 8: Prijava u aplikaciju (Izvor: Vlastita izrada)

Ukoliko je korisnik/doktor unio ispravne podatke prilikom prijave, otvara se glavni izbornik. Sa lijeve strane prikazani su podaci o prijavljenom korisniku, a sa desne strane su prikazane funkcionalnosti aplikacije.



Slika 9: Glavni izbornik (Izvor: Vlastita izrada)

Klikom miša na gumb Pregledi, otvara se prozor u kojem se nalazi lista svih rezerviranih pregleda za taj dan. Korisnik ima mogućnost odabrati datum te se za odabrani datum prikazuju svi rezervirani pregledi.

Datum pregleda: 28. travnja 2020.

	Ime	Prezime	Naziv pregleda	Opis	Početak	Završetak
▶	Petra	Petric	Veliki pregled	Sve u redu	28.4.2020. 7:00	28.4.2020. 7:30
	Tara	Teric	Opširan pregled	Problemi na zadnj...	28.4.2020. 7:30	28.4.2020. 8:15
	Petra	Petric	Opširan pregled	cfs	28.4.2020. 8:15	28.4.2020. 9:00

Rezerviraj pregled
Natrag

Slika 10: Prikaz rezerviranih pregleda (Izvor: Vlastita izrada)

Ako korisnik želi rezervirati novi pregled, potrebno je kliknuti na gumb rezerviraj pregled. Nakon što se klikne na taj gumb, otvara se forma za rezervaciju pregleda kao što je prikazani na slici 11.

Datum pregleda: 28. travnja 2020.

	Ime	Prezime	Pacijent:	Vrsta pregleda:	Bolnica:	Tjedan trudnoće:	Datum pregleda:	Opis pregleda:
▶	Petra	Petric	Petra Petric	Mali pregled	Opća bolnica Zabok	1	28. travnja 2020.	
	Tara	Teric						
	Petra	Petric						

Natrag
Rezervira
Rezerviraj pregled
Natrag

Slika 11: Rezerviranje pregleda (Izvor: Vlastita izrada)


U trenutku kada korisnik želi rezervirati termin pregleda koji je izvan radnog vremena, aktivira se okidač koji javlja grešku (slika 12.).

Datum pregleda: 28. travnja 2020.

	Ime	Prezime	Pacijent:		Završet
▶	Petra	Petric	Paula Tenšić	20. 7:00	28.4.202
	Tara	Teric	Opširan pregled	20. 7:30	28.4.202
	Petra	Petric		20. 8:15	28.4.202

Bolnica:
Opća bolnica Zabok

Upozorenje

 P0001: Termin se ne može rezervirati jer je radno vrijeme do 14:00!

[U redu](#)

[Natrag](#)
[Rezervira](#)
[Rezerviraj pregled](#)

Slika 12: Rezerviranje pregleda izvan radnog vremena (Izvor: Vlastita izrada)

Ukoliko na glavnom izborniku odaberemo kategoriju Pacijenti otvara se lista svih pacijenta. Na dnu prozora ponuđene su tri opcije: dodavanje novog pacijenta, uređivanje odabranog iz tablice te brisanje odabranog iz tablice.

	ID	Ime	Prezime	Datum rođenja	Država	Grad	Ulica
▶	1	Petra	Petric	10.10.1991.	Hrvatska	Zabok	Cesarica
	2	Agata	Petrović	1.3.1991.	Hrvatska	Zagreb	Lj.Gaja
	3	Paula	Tenšić	1.10.1899.	Hrvatska	Zagreb	Merlica
	4	Tara	Teric	28.4.1994.	Hrvatska	Zadar	Janka Lesk

[Dodaj](#)
[Uredi](#)
[Obriši](#)
[Natrag](#)

Slika 13: Pregled pacijenta (Izvor: Vlastita izrada)

Kada korisnik odabere dodavanje novog pacijenta, otvara se prozor gdje je potrebno upisati osnovne informacije o novom pacijentu koji se želi dodati.

	ID	Ime	Prezime	Datum rođenja	Država	Grad	Ulica
▶	1	Petra	Petric	10.10.1991.	Hrvatska	Zabok	Cesarica
	2	Agata				Zagreb	Lj.Gaja
	3	Paula				Zagreb	Merica
	4	Tara				Zadar	Janka Lesk

Ime

Prezime

Datum rođenja 28. travnja 2020.

Država

Grad

Ulica

Broj

Email

Kontakt

Slika 14: Dodavanje pacijenta (Izvor: Vlastita izrada)

Korisnik na glavnom izborniku ima mogućnost odabrati Evidencija podataka o trudnoći. Kada se klikne na taj gumb otvara se prozor na kojem su dvije tablice. Prva tablica prikazuje pacijente te odabirom pacijenta iz tablice u drugoj se prikazuje trudnoća ukoliko postoji.

Popis pacijentica

	ID	Ime	Prezime	Datum rođenja	Email	Kontakt
▶	1	Petra	Petric	10.10.1991.	petric@gmail.com	0924948329
	2	Agata	Petrović	1.3.1991.	agata@gmail.com	09163910432
	3	Paula	Tenšić	1.10.1899.	paula@gmail.com	01938753
	4	Tara	Teric	28.4.1994.	tara@gmail.com	09148503355

Pregled trudnoća

	Id	Pocetak	Ocekivani kraj	Informacije
▶	1	21.1.2020.	25.9.2020.	Uređan tijek

Slika 15: Evidencija trudnoća (Izvor: Vlastita izrada)

Kada korisnik želi dodati trudnoću za određenog pacijenta, odabire iz prve tablice pacijenta te se klikom na gumb Dodaj trudnoću otvara novi prozor. U tom prozoru unose se potrebni podaci za trudnoću poput datuma početka trudnoće te planiranog datuma poroda. Isto tako, unose se informacije o trudnoći te o samom pacijentu kako bi se kasnije pregledi lakše izvršavali te ažurirali.

The screenshot shows a web form with a blue header and a light blue background. At the top, there is a dropdown menu labeled 'Pacijent' with 'Petra Petric' selected. Below this, there are two date input fields: 'Početak trudnoće' (Start of pregnancy) and 'Očekivani kraj trudnoće' (Expected end of pregnancy), both set to '28. travnja 2020.'. Below the date fields is a large text area labeled 'Informacije o trudnoći' (Pregnancy information). At the bottom right, there are two buttons: 'Dodaj trudnoću' (Add pregnancy) and 'Natrag' (Back).

Slika 16: Dodavanje trudnoće za odabranog pacijenta (Izvor: Vlastita izrada)

7. Zaključak

U ovom projektu prikazana je i opisana izrada aplikacije za praćenje trudnoće. Naglasak prilikom izrade aplikacije je na aktivnim i temporalnim bazama podataka. Uz opis same izrade u alatima, prikazano je kako okidači mogu riješiti mnoge stvari umjesto programskog koda i to na puno bolji i efikasniji način. Isto tako, ovakav tip aplikacije mogao bi biti vrlo koristan na tržištu kako bi smanjio liste čekanja za preglede trudnoće, a samim time uštedio vrijeme i olakšao posao doktorima.

Za izradu aplikacije koristio sam znanja koja sam stekao na preddiplomskom studiju. Za kreiranje tablica i spajanje istih koristio sam Navicat 15 for PostgreSQL. Funkcionalnosti i izgled aplikacije izrađeni su u Visual Studiu 2019 u jeziku C koristeći .NET Framework. Sva dokumentacija napisana je u LaTeX-u.

Popis literature

- [1] M. Schatten, „Web-materijali s laboratorijskih vježbi na kolegiju Teorija baza podataka”, 2017., Dostupno na: <http://autopoiesis.foi.hr/>.
- [2] „PostgreSQL”, 2009., Dostupno na: <http://sigurnost.zemris.fer.hr/ostalo/baze/2007-galinovic/Modelikontrolapristupa-u-aktivnim-i-objektno-orijentiranim-bazama-podatka.pdf>.
- [3] A. Galinović, „Modeli kontrole pristupa u aktivnim i objektno-orijentiranim bazama podataka”, 2008., Dostupno na: [http://media.lukaperkov.net/lukaperkov.net/files/papers/Seminar\[2009\]Perkov](http://media.lukaperkov.net/lukaperkov.net/files/papers/Seminar[2009]Perkov.pdf).
- [4] S. Kajin, „Temporalne baze podataka [Završni rad]. Sveučilište u Zagrebu, Fakultet organizacije i informatike, Varaždin”, 2017., Dostupno na: <https://www.bib.irb.hr/895669>.

Popis slika

1.	ERA model (Izvor: Vlastita izrada)	3
2.	Okidač za rezervaciju termina pregleda (Izvor: Vlastita izrada)	5
3.	Okidač za provjeru radnog vremena (Izvor: Vlastita izrada)	6
4.	Klasa za povezivanje s bazom (Izvor: Vlastita izrada)	7
5.	Klasa doktor 1. dio (Izvor: Vlastita izrada)	8
6.	Klasa doktor 2. dio (Izvor: Vlastita izrada)	9
7.	Klasa doktor 3. dio (Izvor: Vlastita izrada)	9
8.	Prijava u aplikaciju (Izvor: Vlastita izrada)	10
9.	Glavni izbornik (Izvor: Vlastita izrada)	10
10.	Prikaz rezerviranih pregleda (Izvor: Vlastita izrada)	11
11.	Rezerviranje pregleda (Izvor: Vlastita izrada)	11
12.	Rezerviranje pregleda izvan radnog vremena (Izvor: Vlastita izrada)	12
13.	Pregled pacijenta (Izvor: Vlastita izrada)	12
14.	Dodavanje pacijenta (Izvor: Vlastita izrada)	13
15.	Evidencija trudnoća (Izvor: Vlastita izrada)	13
16.	Dodavanje trudnoće za odabranog pacijenta (Izvor: Vlastita izrada)	14