Aplikacija za pomoć u praćenju rada privatne tvrtke

Rabi, Branko

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:053194

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-10**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

APLIKACIJA ZA POMOĆ U PRAĆENJU RADA PRIVATNE TVRTKE

Završni rad

Branko Rabi

Osijek, 2017. godina

Sadržaj

1. UVOD 3

1.1. Zadatak završnog rada 3

2. OPIS ZADAĆE APLIKACIJE 4

2.1. Podatci potrebni za izračune 4

2.2. Način izračuna izdataka (dohodak, doprinosi, porez, prirez, neto primitak) 6

3. IZVORNI KOD PROGRAMA I NAČIN RADA 10

3.1. Izgled baze podataka 10

3.2. Naredbe za izradu baze podataka i pripadajućih tablica 11

3.3. Opis izvornog koda programa 14

3.3.1. Klasa za izradu baze podataka i pripadajućih tablica 14

3.3.2. Klasa za unos novog zaposlenika 16

3.3.3. Klasa za brisanje zaposlenika 17

3.3.4. Klasa zaposlenik 18

3.3.5. Klasa komunikator 19

3.3.6. Klasa promjeniOIB 21

3.3.7. Klasa Evidencija 21

3.3.7. Izbor i obrada podataka zaposlenika 27

3.3.8. Funkcije obrasca *izbor i obrada podataka zaposlenika* 27

3.3.9. Tijelo obrasca *izbor i obrada podataka zaposlenika* 33

3.3.10. Tijelo obrasca podatci o radnom vremenu zaposlenika 41

3.3.11. Obrazac Izračun plaće 56

3.4. Način korištenja aplikacije 57

4. ZAKLJUČAK 62

LITERATURA 63

SAŽETAK 64

ABSTRACT 65

ŽIVOTOPIS 66

1. UVOD

Prema Pravilniku o sadržaju i načinu vođenja evidencije o radnicima (NN 73/2017) Članak 8, svaka tvrtka dužna je voditi evidenciju o radnom vremenu zaposlenika. Evidentira se vrsta radnih sati (redovni, prekovremeni, terenski…) i broj radnih sati. Tvrtka je dužna čuvati evidenciju najmanje 3 godine. Na osnovu broja radnih sati, vrste radnih sati i satnice izračunava se bruto primitak (plaća) zaposlenika.

Na osnovu visine bruto primitka izračunavaju se davanja za mirovinsko i zdravstveno osiguranje. Visina poreza ovisi o visini dohotka, broju uzdržavanih članova (djeca, roditelji, bračni drug ili osoba kojoj je porezni obveznik imenovani skrbnik) i o eventualnoj invalidnosti jednog ili više uzdržavanih članova. Visina prireza porezu ovisi o mjestu stanovanja poreznog obveznika.

U drugom poglavlju opisuju se podatci potrebni za uspješno obavljanje zadatka aplikacije i točna metoda izračuna dohotka, doprinosa, odbitaka, poreza, prireza i neto plaće zaposlenika.

U trećem poglavlju opisan je kod programa. Opisane su klase (korisnički tip podatka) i funkcije koje program poziva u izvođenje tokom svog rada. Opisan je glavni dio programa (detaljan opis), stvaranje objekata i pozivi funkcijama, kao i koja aktivnost pokreće koji dio koda izvršava.

1.1. Zadatak završnog rada

Zadatak Aplikacija za pomoć u praćenju rada privatne tvrtke je omogućiti unos i spremanje podataka o radnom vremenu zaposlenika u bazu podataka. Omogućiti dohvaćanje podataka od strane korisnika u cijelosti ili željenom dijelu te izmjenu, brisanje i dodavanje podataka.

Uz pomoć podataka o radnom vremenu zaposlenika (broj i vrsta radnih sati, satnica) aplikacija treba izraditi izračun bruto plaće. Nakon toga uz pomoć podataka o zaposleniku (broj uzdržavanih članova uže obitelji, visina prireza porezu) aplikacija treba izračunati doprinose, dohodak, porez, prirez i neto plaću.

2. OPIS ZADAĆE APLIKACIJE

U uvodu je dan kratki opis zadaće aplikacije. U ovom poglavlju slijedi detaljan opis zadaće aplikacije s primjerom. Slijedi popis podataka potrebnih za izračune izdataka (neto primitak, doprinosi, dohodak, porez i prirez), kao i način izračuna.

2.1. Podatci potrebni za izračune

U tablici koja slijedi prikazani su podatci na osnovu kojih se izračunava bruto plaća zaposlenika i davanja prema državi.

Tab. 2.1. – ***Podatci o zaposleniku.***

R. br.	Opis
1.	OIB (Osobni identifikacijski broj)
2.	Ime
3.	Prezime
4.	Datum rođenja
5.	Adresa
6.	Mjesto
7.	Satnica
8.	Mirovinski stup
9.	Prirez [%] – ovisi o mjestu stanovanja
10.	Broj uzdržavanih članova uže obitelji
11.	Broj uzdržavane djece
12.	Invalidnost poreznog obveznika ili uzdržavanih članova obitelji
13.	100%-tna invalidnost poreznog obveznika ili uzdržavanih članova obitelji

Na osnovu podataka (8. – 13.) iz Tablice 2.1. (Članak 14. Točka (4) Zakona o porezu na dohodak iz 2017. godine, u daljnjem tekstu zakon) izračunavaju se izdatci prema državi u skladu sa zakonom. Satnica (7.) iz Tablice 2.1. je potrebna radi izračuna mjesečnog bruto primitka zaposlenika. Podatci (1. – 6.) Tablice 2.1. su osnovni podatci o zaposleniku.

Tab. 2.2. – *Broj i vrsta radnih sati i pripadajući koeficijenti.*

R. br.	Opis
1.	Datum
2.	Početak rada
3.	Završetak rada
4.	Zastoj od
5.	Zastoj do
6.	Terenski rad
7.	Sati pripravnosti
8.	Dnevni odmor
9.	Tjedni odmor
10.	Godišnji odmor
11.	Plaćeni ne radni dani, blagdani
12.	Bolovanje
13.	Komplikacije u trudnoći
14.	Rodiljni dopust
15.	Roditeljski dopust
16.	Mirovanje radnog odnosa
17.	Druga prava prema propisu
18.	Plaćeni dopust
19.	Ne plaćeni dopust
20.	Dopušteni izostanak
21.	Ne dopušteni izostanak
22.	Štrajk
23.	Isključenje s rada
24.	Noćni redovni
25.	Noćni blagdanom
26.	Noćni prekovremeni
27.	Redovan rad
28.	Redovni blagdanom
29.	Redovni prekovremeni

Svaka vrsta radnih sati (6. -29.) Tablica 2.2. ima uz sebe podatak, koeficijent za koji se uvećava ili umanjuje iznos satnice. Primjer: cijena rada noću je 1,5 x satnica, bolovanje 0,6 x satnica. Satnica se odnosi na redovan rad. Vrijednosti spomenutih koeficijenata nisu zakonom propisane.

Tab. 2.3. – *Porezni podatci.*

R. br.	Opis
1.	Osnovica osobnog odbitka (2.500,00) [kn]
2.	Koeficijent osobnog odbitka (1,5) [kn]
3.	Uzdržavani članovi obitelji (0,7) [koeficijent]
4.	Prvo uzdržavano dijete (0,7) [koeficijent]
5.	Drugo uzdržavano dijete (1,0) [koeficijent]
6.	Treće uzdržavano dijete (1,4) [koeficijent]
7.	Četvrto uzdržavano dijete (1,9) [koeficijent]
8.	Peto uzdržavano dijete (2,5) [koeficijent]
9.	Šesto uzdržavano dijete (3,2) [koeficijent]
10.	Sedmo uzdržavano dijete (4,0) [koeficijent]
11.	Osmo uzdržavano dijete (4,9) [koeficijent]
12.	Deveto uzdržavano dijete (5,9) [koeficijent]
13.	Slijedeće uzdržavano dijete (1,1) [koeficijent]
14.	Invalidnost poreznog obveznika, svakog člana obitelji ili djeteta (0,4) [koeficijent]
15.	Invalidnost 100% (1,5) [koeficijent] – korištenje invalidnost 100% (14.) isključuje korištenje invalidnost (13)

Svi podatci u Tablici 2.3. služe za izračun osobnog odbitka prema.

2.2. Način izračuna izdataka (dohodak, doprinosi, porez, prirez, neto primitak)

Osnovica osobnog odbitka tablica 2.3. (1.) predstavlja osnovicu pomoću koje, množenjem sa koeficijentima tablica 2.3. (2. – 14.), dobijemo osobni odbitak poreznog obveznika. Prema Zakonu o porezu na dohodak iz 2017. godine članak 14. točka (1), osnovica osobnog odbitka iznosi 2.500,00 kn.

Koeficijent osobnog odbitka tablica 2.3. (2.**)** množenje ovog koeficijenta sa osnovicom osobnog odbitka daje osnovni osobni odbitak. Prema Zakonu o porezu na dohodak iz 2017. godine članak 14. točka (3) koeficijent osobnog odbitka iznosi 1,5.

Osnovni osobni odbitak poreznog obveznika predstavlja osnovni neoporezivi dio dohotka. Osnovni osobni odbitak dobije se množenjem osnovice osobnog odbitka sa koeficijentom osobnog odbitka. Formula:

$$P_{OD} = O_{OD} \text{ x } K_{OD} = 2500\text{x } 1{,}5 = 3750 \quad (2\text{–}1)$$

rezultat se zaokružuje na 100, što daje rezultat 3800.

P_{OD} – Osnovni osobni odbitak poreznog obveznika [kn].

O_{OD} – Osnovica osobnog odbitka [kn].

K_{OD} – Koeficijent osobnog odbitka.

Osobni odbitak poreznog obveznika predstavlja potpuni neoporezivi dio dohotka. Dobije se uvećanjem osnovnog osobnog odbitka s obzirom na broj uzdržavane djece i/ili članova obitelji, te eventualne invalidnosti istih. Ako porezni obveznik uzdržava djecu, koeficijent uz pripadajući broj djece množi se sa osnovicom osobnog odbitka i zbraja sa osnovnim osobnim odbitkom. Ako porezni obveznik uzdržava članove uže obitelji, broj uzdržavanih članova množi se sa koeficijentom iz tablice 2.3. (3.) pa množi sa osnovicom osobnog odbitka te zbraja osnovnom osobnom odbitku. Na isti način dodaje se i odbitak ostvaren eventualnom invalidnošću uzdržavanih članova. Formula:

$$O_D = P_{OD} + k_{DJECE} \text{ x } O_{OD} + b_{UZDR} \text{ x } k_{UZDR} \text{ x } O_{OD} + b_{INV} \text{ x } k_{INV} \text{ x } O_{OD} + b_{INV111} \text{ x } \text{ x } k_{INV111} \text{ x } O_{OD} \quad (2\text{-}2)$$

b_{DJECE}, b_{UZDR}, b_{INV}, b_{INV100} – broj djece, uzdržavanih članova, invalida, invalida 100%.

k_{DJECE}, k_{UZDR}, k_{INV}, b_{INV100} – koeficijent uz broj djece, koeficijent za uzdržavane članove obitelji, koeficijent za invalidnost poreznog obveznika i/ili djeteta i/ili uzdržavanog člana, koeficijent u slučaju uzdržavanja 100%-tnog invalida.

Primjer izračuna svih izdataka propisanih Zakonom o Porezu na Dohodak. Za primjer uzmimo zaposlenika koji uzdržava 2 djeteta i 2 roditelja, od kojih je jedan invalid i jedan invalid 100%. Korisnik je drugog stupa mirovinskog osiguranja, živi u području gdje je prirez 10%.

Tab. 2.4. – *Primjer izračuna doprinosa i poreza na dohodak.*

R. br.	Opis	Vrijednost [kn]
1.	Ukupni primitak (bruto plaća)	20.000,00
2.	Doprinos za mirovinsko osiguranje I. Stup (r.br. 1 x 0,15)	3.000,00

3.	Doprinos za mirovinsko osiguranje II. Stup (r.br. 1 x 0,05)	1.000,00
4.	Obvezni doprinosi – UKUPNO (r.br. 2 + r.br 3)	4.000,00
5.	Dohodak (r.br. 1 – r.br. 4)	16.000,00
6.	Osnovni osobni odbitak	3.800,00
7.	Osobni odbitak za uzdržavanu djecu (1 x 2.500,00)*	2.500,00
8.	Osobni odbitak za uzdržavane članove obitelji (2 x 0,7 x 2.500,00)	3.500,00
9.	Invalidnost poreznog obveznika i/ili uzdržavanih članova (1 x 0,4 x 2.500,00)	1.000,00
10.	Invalidnost 100% uzdržavanih članova (1 x 1,5 x 2500)	3.750,00
11.	Osobni odbitak (r.br. 6 + 7 + 8 + 9 + 11)	14.550,00
12.	Porezna osnovica (r.br. 5 – r.br. 11)	1.450,00
13.	Porezna osnovica do 17.500,00za primjenu poreza od 24%	1.450,00
14.	Porez po stopi 24% (r.br. 13 x 1,24)	348,00
15.	Porezna osnovica iznad 17.511,11 za primjenu poreza od 36%	0,00
16.	Porez po stopi 36% (r.br. 15 x 1,36)	0,00
17.	Ukupna obveza poreza (r.br. 14 + r.br. 15)	348,00
18.	Prirez porezu na dohodak (11%) (r.br. 17 x 1,1)	34,80
19.	Obveze prireza i poreza (r.br. 17 + r.br. 18)	382,80
20.	Neto primitak – neto plaća (r.br. 5 – r.br. 19)	15.617,20
	Obveze poslodavca:	
21.	Doprinos za zdravstveno osiguranje (r.br. 1 x 0,15)	3.000,00
22.	Doprinos za zaštitu zdravlja na radu (r.br. 1 x 0,005)	100,00
23.	Doprinos za zapošljavanje (r.br. 1 x 0,017)	340,00
24.	Ukupni izdatci poslodavca (r.br. 1 + 21 + 22 + 23)	23.440,00

* - Koeficijent za broj uzdržavane djece se ne izračunava već uzima iz tablice.

Prema Zakonu o porezu na dohodak, bruto primitak (plaća) je ukupna količina novca (ali i novčana protuvrijednost roba i/ili usluga) s kojom poslodavac plaća zaposlenika

Dohodak se dobije odbijanjem doprinosa za mirovinsko osiguranje od bruto primitka.

Neto primitak se dobije oporezivanjem dohotka umanjenog za osobni odbitak poreznog obveznika.

Osobni odbitak poreznog obveznika je neoporezivi dio dohotka, koji ovisi o broju uzdržavanih članova uže obitelji i eventualnoj invalidnosti poreznog obveznika i/ili članova obitelji.

3. IZVORNI KOD PROGRAMA I NAČIN RADA

Aplikacija za svoj rad koristi klase ugrađene u .NET Framework i bazu podataka. U .NET-u su definirane klase za komunikaciju sa bazom podataka. Aplikacija se sastoji od tri obrasca.

Prvi obrazac služi za unos podataka za spajanje na bazu podataka, kreiranje baze podataka, unošenje novog zaposlenika, brisanje postojećeg zaposlenika, promjenu OIB-a zaposlenika. Zatim za izbor zaposlenika i izbor intervala podatka koji se želi dohvatiti.

Drugi obrazac služi za prikaz dohvaćenih podatka, unos, promjenu i brisanje podataka i predračun.

U treći obrazac služi za prikaz izračuna bruto plaće, doprinosa, poreza, prireza, odbitaka i neto plaće.

Program funkcionira na način da se spoji na bazu podataka, napravi upit, prihvati podatke, zatvori vezu. Nakon obrade podataka na korisničkom računalu, na zahtjev korisnika, program se spaja na bazu podataka, sinkronizira bazu podataka sa promijenjenim podatcima na korisničkom računalu te zatvara vezu.

Za komunikaciju s bazom podataka, prihvaćanje podataka i vraćanje promijenjenih podataka bazi podataka, koriste se ugrađene klase programskog jezika C#.

3.1. Izgled baze podataka

Baza podataka sastoji se od tri tablice plus jedna za svakog zaposlenika. Broj tablica u bazi podataka biti će tri uvećano za broj zaposlenika. Ako je broj zaposlenika 10, broj tablica u bazi podataka bit će 3 + 10 = 13. Tablica u koju se unosi radno vrijeme zaposlenika izrađuje se u trenutku kreiranja novog zaposlenika u bazi podataka. Ostale se izrađuju sa bazom podataka.

Tablice prisutne u bazi podataka su:

- Tablica u koju se zapisuju podatci o zaposleniku iz tablice 2.1.
- Tablica u koju se zapisuju porezni podatci iz tablice 2.3.
- Tablica u koju se zapisuju koeficijenti pojedine vrste radnih sati iz tablice 2.2., od rednog broja 6 – 29.
- Tablica u koju se zapisuje broj i vrsta radnih sati koju je zaposlenik odradio u određenom danu, podatci iz tablice 2.2. Ova tablica izrađuje se za svakog zaposlenika.

3.2. Naredbe za izradu baze podataka i pripadajućih tablica

Ove naredbe bit će poslane od programa bazi podataka prilikom pritiska na dugme *stvori bazu podataka* unutar programa.

SQL naredba za izradu baze podataka:

```
CREATE DATABASE evidencija
```

SQL naredba za izradu tablice zaposlenika:

```
CREATE TABLE zaposlenici(
        oib VARCHAR(11),
        ime VARCHAR(31) NOT NULL,
        prezime VARCHAR(31) NOT NULL,
        rodjenje DATETIME NOT NULL,
        adresa VARCHAR(41) NOT NULL,
        mjesto VARCHAR(31) NOT NULL,
        satnica DECIMAL(5,2),
        mjesecna_placa DECIMAL(8,2),
        mirovinski_stup SMALLINT,
        prirez DECIMAL(2,1),
        uzdrzavani_clanovi SMALLINT,
        br_uzdrzavane_djece SMALLINT,
        invalid_por_obvez_ili_obit SMALLINT,
        invalid_111_posto SMALLINT,
        CONSTRAINT pk_oib PRIMARY KEY(oib),
        CONSTRAINT chk_mir_stup CHECK (mirovinski_stup IN ('1', '2')));
```

Naredba bazi podataka koja definira relaciju. Broj atributa n-torke, nazive atributa, vrstu podatka pojedinog atributa. Postavlja primarni ključ relacije *oib*. Postavlja domenu vrijednosti atributa mirovinski stup, 1 ili 2.

SQL naredba za izradu tablice koeficijenti:

```
CREATE TABLE koeficijenti (
        oib_k VARCHAR(11) NOT NULL DEFAULT 1,
        redovan_rad_k DECIMAL(2,1) NOT NULL DEFAULT 1,
```

```
    prekovremeni_rad_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    rad_blagdan_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    terenski_rad_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    nocni_redovni_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    nocni_prekovrem_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    nocni_blagdan_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    sati_pripravnosti_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    placeni_blagdan_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    bolovanje_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    komplikacije_trud_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    rodiljni_dopust_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    roditeljski_dopust_k DECIMAL(2,1) NOT NULL DEFAULT 1,
    mirovanje_rad_odnos_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    druga_prava_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    placeni_dopust_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    NE_placeni_dopust_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    izostanak_zahtjevani_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    izostanak_neopravdani_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    strajk_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    iskljucenje_srada_k DECIMAL (2,1) NOT NULL DEFAULT 1,
    CONSTRAINT koeff_pk PRIMARY KEY(oib_k));
```

SQL naredba za izradu tablice porez:

```
CREATE TABLE porez(
    oznaka INT IDENTITY(1,1),
    Osnovica INT NOT NULL,
    koef_odbitka DECIMAL(2,1),
    uzdrzavan_obitelj DECIMAL(2,1),
    prvo_dijete DECIMAL(2,1),
    drugo_dijete DECIMAL(2,1),
    trece_dijete DECIMAL(2,1),
    cetvrto_dijete DECIMAL(2,1),
    peto_dijete DECIMAL(2,1),
    sesto_dijete DECIMAL(2,1),
```

sedmo_dijete DECIMAL(2,1),

osmo_dijete DECIMAL(2,1),

deveto_dijete DECIMAL(2,1),

sljedece_dijete DECIMAL(2,1),

invalidnost DECIMAL(2,1),

invalidnost_100 DECIMAL(2,1),

CONSTRAINT pk_oznaka PRIMARY KEY(oznaka));

SQL naredba za izradu tablice evidencije radnih sati zaposlenika:

CREATE TABLE OIB[oib] (datum DATE,

Pocetak_rada SMALLINT DEFAULT 0,

Zavrsetak_rada SMALLINT DEFAULT 0,

Zastoj_od SMALLINT DEFAULT 0,

Zastoj_do SMALLINT DEFAULT 0,

Terenski_rad SMALLINT DEFAULT 0,

Sati_pripravnosti SMALLINT DEFAULT 0,

Dnevni_odmor SMALLINT DEFAULT 0,

Tjedni_odmor SMALLINT DEFAULT 0,

Godisnji_odmor SMALLINT DEFAULT 0,

Placeni_neradni SMALLINT DEFAULT 0,

Bolovanje SMALLINT DEFAULT 0,

Trudnoca_komplikacije SMALLINT DEFAULT 0,

Rodiljni_dopust SMALLINT DEFAULT 0,

Roditeljski_dopust SMALLINT DEFAULT 0,

Mirovanje_rad_odnosa SMALLINT DEFAULT 0,

Druga_prava SMALLINT DEFAULT 0,

Placeni_dopust SMALLINT DEFAULT 0,

Ne_placeni_dopust SMALLINT DEFAULT 0,

Dopusteni_izostanak SMALLINT DEFAULT 0,

Ne_dopusteni_izostanak SMALLINT DEFAULT 0,

strajk SMALLINT DEFAULT 0,

Iskljucenje_s_rada SMALLINT DEFAULT 0,

Nocni_redovni SMALLINT DEFAULT 0,

Nocni_blagdani SMALLINT DEFAULT 0,

Nocni_prekovremeni SMALLINT DEFAULT 0,
Redovan_rad SMALLINT DEFAULT 0,
Redovni_blagdanom SMALLINT DEFAULT 0,
Redovni_prekovremeni SMALLINT DEFAULT 0,
CONSTRAINT OIB[oib]_pk PRIMARY KEY(datum));

Prilikom slanja naredbe za izradu relacije bazi podataka [oib] će biti zamijenjen sa stvarnom vrijednošću OIB-a zaposlenika, unesenom od strane korisnika.

3.3. Opis izvornog koda programa

Slijedi opis klasa na čiju sliku će se stvarati objekti koje će glavni program stvarati da obavljaju dane zadatke. Iza slijedi izvorni kod glavnog programa.

3.3.1. Klasa za izradu baze podataka i pripadajućih tablica

Prilikom prvog pokretanja programa potrebno je na bazi podatka stvoriti relacije u koje će se upisivati podatci. Slijedi opis klase koja obavlja ovaj zadatak.

U zaglavlju klase nalaze se imenici koje će klasa koristiti. Ako nisu prisutni, treba dodati imenike *System.Data* i *System.Data.SqlClient*. U ovim imenicima su opisi klasa za spajanje na bazu podataka i izvršavanje naredbi i upita.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
```

Klasa koristi objekte klase *SqlConnection* i *SqlCommand* za komunikaciju sa bazom podataka. *SqlConnection* sadržava podatke za spajanje na bazu podataka, a *SqlCommand* naredbu ili upit bazi podataka.

```
class StvoriTablice
{
 SqlConnection stvori_spoj; // deklaracija objekta klase SqlConnection.
 SqlCommand stvori_zapovijed; // deklaracija objekta SqlCommand.

 public StvoriTablice() { } // Zadani konstruktor klase, prazan.
```

Klasa sadržava ugrađenu funkciju za spajanje na bazu podataka i izrada tablica. Funkcija prihvaća 2 argumenta tipa *string*, koji sadrže podatke za spajanje na bazu podataka, unesenih od

strane korisnika, preko grafičkog korisničkog sučelja. Podatci iz prvog argumenta koriste se za spajanje na bazu evidencija, a iz drugog za spajanje na bazu master.

```
public void napraviTablice(string evidencija_string, string master_string)
{
 string zapovijed;
 zapovijed = "CREATE DATABASE evidencija"; // Zapovijed koju treba izvršiti na BP
 stvori_spoj = new SqlConnection(); // stvaranje objekta klase SqlConnection.
 stvori_spoj.ConnectionString = master_string; // pridruživanje podataka objektu.
 stvori_zapovijed = new SqlCommand(); // stvaranje objekta klase SqlCommand.
 stvori_zapovijed.Connection = stvori_spoj; // Pridruživanje podataka preko objektu.
 stvori_zapovijed.CommandText = zapovijed; // Pridruživanje podataka objektu.
 stvori_spoj.Open(); // Otvaranje veze sa bazom podataka.
 stvori_zapovijed.ExecuteNonQuery(); // Pridruživanje zapovijedi (ne upit).
```

Gornji kod prikazuje stvaranje objekata klase *SqlConnection* i *SqlCommand*. Pridruživanje niza znakova sa podatcima za spoj sa bazom podataka, koji trebaju imati oba objekta. Pridruživanje niza znakova sa naredbom koja treba biti izvršena na bazi podataka.

Nakon toga se otvara veza sa bazom podataka. Na kraju izvršava se naredba koja nije upit (*ExecuteNonQuery();*) na bazi podataka. Ovime je stvorena baza podataka evidencija.

Slijedeći kod provjerava je li veza sa bazom podataka otvorena, ako je zatvara ju.

```
if (stvori_spoj != null)
{
 stvori_spoj.Close();
}
```

Varijabli zapovijed pridružuje se slijedeća naredba koja će biti izvršena na bazi podataka.

```
zapovijed = "CREATE TABLE zaposlenici(oib VARCHAR(11), ime VARCHAR(30) NOT NULL, prezime
VARCHAR(30) NOT NULL, rodjenje DATETIME NOT NULL, adresa VARCHAR(40) NOT NULL, mjesto
VARCHAR(30) NOT NULL, satnica DECIMAL(5,2), mjesecna_placa DECIMAL(8,2), mirovinski_stup
SMALLINT, prirez DECIMAL(2,0), uzdrzavani_clanovi SMALLINT, br_uzdrzavane_djece SMALLINT,
invalid_por_obvez_ili_obit SMALLINT, invalid_100_posto SMALLINT, CONSTRAINT pk_oib
PRIMARY KEY(oib),
CONSTRAINT chk_mir_stup CHECK (mirovinski_stup IN ('1', '2')));";
```

Varijabla zapovijed se izjednačava sa sljedećom naredbom, koja služi izradu tablice zaposlenika u bazi podataka. Ovaj puta program se spaja sa novo napravljenom bazom evidencija te izrađuje tablicu u njoj.

```
stvori_spoj.ConnectionString = evidencija_string;
stvori_zapovijed.Connection = stvori_spoj;
stvori_zapovijed.CommandText = zapovijed;
stvori_spoj.Open();
stvori_zapovijed.ExecuteNonQuery();
```

Sada se objektima za spajanje na bazu podataka pridružio niz znakova iz argumenta za spajanje na bazu evidencija. Objektu za izvršavanje naredbi pridružila se nova naredba, otvara se veza sa

bazom podataka i na njoj se izvršava naredba. Na isti način izrađuju se ostale tablice i početni unosi u njih.

Prikaz naredbe za ubacivanje podataka u tablicu koeficijenti.

```
zapovijed = "INSERT INTO koeficijenti VALUES('osnova', '1.0', '1.0', '1.0', '1.0', '1.0',
'1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', '1.0', 1.0',
'1.0', '1.0', '1.0')";

stvori_zapovijed.CommandText = zapovijed;
stvori_zapovijed.ExecuteNonQuery();
```

S obzirom da je veza sa bazom podataka otvorena, nije potrebno pridruživanje znakovnih nizova objektima za spajanje i izvršavanje naredbi. Niti otvaranje veze. Na isti način rade se i ostali potrebni unosi. Kada su sve naredbe odrađene zatvara se veza sa bazom podataka.

3.3.2. Klasa za unos novog zaposlenika

U zaglavlje klase, kao i kod prethodne, potrebno je upisati imenike u kojima su opisi klasa za komunikaciju sa bazom podataka. Spajanje sa bazom podataka i izvršavanje naredbi na istoj izvode se na isti način kao i kod prethodno opisane klase. Na početku klase su zadani konstruktor i objekti za komunikaciju s bazom podataka.

```
class UnosZaposlenika
{
 public UnosZaposlenika() { }

 SqlConnection unos_spajanje;
 SqlCommand unos_zapovijed;
```

Slijedi metoda za spajanje na bazu podataka i izvršavanje naredbi na istoj. Metoda kao argumente prihvaća podatke o zaposleniku unesene od strane korisnika preko korisničkog sučelja. Podatke koristi za izradu tablice radnog vremena zaposlenika i unose podataka o zaposleniku u ostale tablice.

Prilikom izrade tablice u ime tablice umeće se OIB unesen u korisničko sučelje i predan metodi prilikom poziva. Isti OIB koristi se i za nazive *constraint-a* primarnog ključa prilikom izrade tablice.

Ostali podatci iz liste argumenata koriste se za unose u tablice zaposlenici i tablicu koeficijenti.

Princip otvaranja veze i izvršavanja naredbi isti je kao u prethodnom podpoglavlju.

```
public void spojiSe(string spoj_string, string oib, string ime, string prezime, string
rodjenje, string adresa, string mjesto)
{
```

```
string zapovijed;

zapovijed = "CREATE TABLE OIB" + oib + "(datum DATE, Pocetak_rada SMALLINT DEFAULT 0,
Zavrsetak_rada SMALLINT DEFAULT 0, Zastoj_od SMALLINT DEFAULT 0, Zastoj_do SMALLINT
DEFAULT 0, Terenski_rad SMALLINT DEFAULT 0, Sati_pripravnosti SMALLINT DEFAULT 0,
Dnevni_odmor SMALLINT DEFAULT 0, Tjedni_odmor SMALLINT DEFAULT 0, Godisnji_odmor
SMALLINT DEFAULT 0, Placeni_neradni SMALLINT DEFAULT 0, Bolovanje SMALLINT DEFAULT 0,
Trudnoca_komplikacije SMALLINT DEFAULT 0, Rodiljni_dopust SMALLINT DEFAULT 0,
Roditeljski_dopust SMALLINT DEFAULT 0, Mirovanje_rad_odnosa SMALLINT DEFAULT 0,
Druga_prava SMALLINT DEFAULT 0, Placeni_dopust SMALLINT DEFAULT 0, Ne_placeni_dopust
SMALLINT DEFAULT 0, Dopusteni_izostanak SMALLINT DEFAULT 0, Ne_dopusteni_izostanak
SMALLINT DEFAULT 0, strajk SMALLINT DEFAULT 0, Iskljucenje_s_rada SMALLINT DEFAULT 0,
Nocni_redovni SMALLINT DEFAULT 0, Nocni_blagdani SMALLINT DEFAULT 0, Nocni_prekovremeni
SMALLINT DEFAULT 0, Redovan_rad SMALLINT DEFAULT 0, Redovni_blagdanom SMALLINT DEFAULT
0, Redovni_prekovremeni SMALLINT DEFAULT 0, CONSTRAINT OIB" + oib + "_pk PRIMARY
KEY(datum));";

unos_spajanje = new SqlConnection();
unos_spajanje.ConnectionString = spoj_string;
unos_zapovijed = new SqlCommand();
unos_zapovijed.Connection = unos_spajanje;
unos_zapovijed.CommandText = zapovijed;
unos_spajanje.Open();
unos_zapovijed.ExecuteNonQuery();

zapovijed = "INSERT INTO zaposlenici (OIB, ime, prezime, rodjenje, adresa, mjesto)
VALUES'" + oib + "', '" + ime + "', '" + prezime + "', '" + rodjenje + "', '" + adresa
', '" + mjesto + "');";

unos_zapovijed.CommandText = zapovijed;
unos_zapovijed.ExecuteNonQuery();

zapovijed = "INSERT INTO koeficijenti VALUES('" + oib + "', '1', '1', '1', '1', '1', "
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1');";

unos_zapovijed.CommandText = zapovijed;
unos_zapovijed.ExecuteNonQuery();
}
```

3.3.3. Klasa za brisanje zaposlenika

Klasa za brisanje zaposlenika sadržava metodu nazvanu Obrisi, koja prihvaća 2 argumenta. Podatke za spajanje na bazu podataka sa korisničkog sučelja i oib zaposlenika sa korisničkog sučelja. Oib je ključ za brisanje pripadajuće tablice i unosa u tablicama koeficijenti i zaposlenici. Spajanje na bazu podataka i izvršavanje tablica izvodi se na prije opisani način.

```
class BrisiZaposlenog
{
 SqlConnection brisi_konekcija;
 SqlCommand brisi_zapovijed;

 public BrisiZaposlenog() {  }

 public void Obrisi(string oib, string spojni_string)
 {
 try
 {
  brisi_konekcija = new SqlConnection();
```

```
 brisi_konekcija.ConnectionString = spojni_string;
 brisi_konekcija.Open();
 brisi_zapovijed = new SqlCommand();
 brisi_zapovijed.Connection = brisi_konekcija;
 brisi_zapovijed.CommandText = "DROP TABLE OIB" + oib;
 brisi_zapovijed.ExecuteNonQuery();

 brisi_zapovijed.CommandText = "DELETE FROM zaposlenici WHERE oib = '" + oib + "'";
 brisi_zapovijed.ExecuteNonQuery();

 brisi_zapovijed.CommandText = "DELETE FROM koeficijenti WHERE oib_k = '" + oib + "'";
 brisi_zapovijed.ExecuteNonQuery();
}
catch (Exception ex)
{
 throw ex;
}
finally
{
 if (brisi_konekcija != null)
 {
  brisi_konekcija.Close();
 }
}}}
```

Kod *finally* izvršava se bez obzira dali se dogodila iznimka ili ne te zatvara vezu sa bazom podataka ako je otvorena.

3.3.4. Klasa zaposlenik

Klasa zaposlenik čuva osnovne podatke o zaposleniku. Služi za izbor zaposlenika na čijim podatcima će se raditi, kao i za prikaz osnovnih podataka o zaposleniku na grafičkom sučelju. Sve varijable sa podatcima o zaposleniku su privatne. Postoje metode za dohvaćanje vrijednosti, ali ne i za postavljanje. Postoji javni parametarski konstruktor koji postavlja vrijednosti varijabli. Podatci o zaposleniku se dohvaćaju iz baze podataka.

```
class Zaposlenik
{
 private string ime;
 private string prezime;
 private DateTime rodjenje;
 private string adresa;
 private string mjesto;
 private string oib;

 public Zaposlenik() { }

// Metode za dohvaćanje vrijednosti varijabli.
 public string Ime
 {
  get { return ime; }
 }

 public string Prezime
 {
  get { return prezime; }
```

```
}

public DateTime Rodjenje
{
 get { return rodjenje; }
}

public string Adresa
{
 get { return adresa; }
}

public string Mjesto
{
 get { return mjesto; }
}

public string Oib
{
 get { return oib; }
}

// Parametarski konstruktor.
public Zaposlenik(string o, string i, string p, DateTime r, string a, string m)
{
 ime = i;
 prezime = p;
 oib = o;
 adresa = a;
 mjesto = m;
 rodjenje = r;
}

// Prepisana metoda ToString() sa zadanim formatom ispisa.
public override string ToString()
{
 return string.Format("{0}, {1}, {2}, {3}, OIB: {4}", ime, prezime, adresa, mjesto,
 oib);
}
}
```

Klasa sadrži prepisanu metodu *ToString()* koja kada pozvana ispisuje podatke na od programera zadani način.

3.3.5. Klasa komunikator

Klasa komunikator služi za spajanje na bazu podataka i izrade liste objekata klase *Zaposlenik* i vraćanje liste pozivajućem kodu. U ovoj klasi stvara se objekt klase *SqlDataReader* koji prihvaća podatke iz baze podataka. I stvara se generička lista objekata klase zaposlenik. Lista se popunjava objektima prema podatcima povučenim iz baze podataka.

```
class Komunikator
{
 SqlConnection komunikator_conn;
 SqlCommand zaposlenik_zapovijed;
 SqlDataReader ucitavac_podataka;
 List<Zaposlenik> lista_zaposlenika;
```

```
public Komunikator() { }
```

U klasi se nalazi javna metoda koja vraća listu objekata klase *Zaposlenik*. Kao argumente prihvaća niz znakova za spajanje na bazu podataka i naredbu koja će biti poslana bazi podataka. Izgled naredbe ovisi o unosu podataka na grafičko korisničko sučelje od strane korisnika.

```
public List<Zaposlenik> VratiListu(string spoj, string zapovijed)
{
try
{
 komunikator_conn = new SqlConnection();
 komunikator_conn.ConnectionString = spoj;
 zaposlenik_zapovijed = new SqlCommand();
 zaposlenik_zapovijed.Connection = komunikator_conn;
 zaposlenik_zapovijed.CommandText = zapovijed;
 komunikator_conn.Open();
```

Stvaranje generičke liste objekata klase zaposlenik.

```
 lista_zaposlenika = new List<Zaposlenik>();
```

Izvršava se upit na bazi podataka. Rezultati upita predaju se objektu tipa *SqlDataReader*. Dok god objekt *SqlDataReader* čita podatke dodaje novi objekt u generičku listu objekata. Dodavanje se vrši pomoću metode add definirane za generičku listu objekata. Sami objekti u listi se stvaraju pomoću parametarskog konstruktora klase *Zaposlenik*, kojem se kao parametri daju podatci iz objekta tipa *SqlDataReader*. Za svaki parametar koji se predaje kao argument konstruktoru definiran je tip podatka i broj stupca tablice u bazi podataka. Na kraju se lista objekata vraća pozivajućem kodu.

```
 ucitavac_podataka = zaposlenik_zapovijed.ExecuteReader();

 while (ucitavac_podataka.Read() == true)
 {
  lista_zaposlenika.Add(new Zaposlenik(ucitavac_podataka.GetString(0),
  ucitavac_podataka.GetString(1), ucitavac_podataka.GetString(2),
  ucitavac_podataka.GetDateTime(3), ucitavac_podataka.GetString(4),
  ucitavac_podataka.GetString(5)));
 }
 return lista_zaposlenika;
}

catch (Exception ex)
{
 throw ex;
}
finally
{
 if (komunikator_conn != null)
{
komunikator_conn.Close();
}
```

```
}}}
```

3.3.6. Klasa promjeniOIB

Klasa služi za promjenu OIB-a zaposlenika. Promjena OIB-a zahtjeva promjenu imena tablice u kojoj se zapisuju podatci o radnom vremenu zaposlenika uz promjenu unosa u ostalim tablicama. Za promjenu naziva tablice u bazi podataka, koristi se ugrađena funkcija u samoj bazi.

```
class PromjeniOIB
{
SqlConnection sql_spoj;
SqlCommand sql_zapovijed;

public PromjeniOIB() { }

public void promjenaOiba(string OIBstari, string OIBnovi, string spojni_string)
{
try
{
sql_spoj = new SqlConnection();
sql_spoj.ConnectionString = spojni_string;
sql_zapovijed = new SqlCommand();
sql_zapovijed.Connection = sql_spoj;
string zapovijed;
zapovijed = "EXEC sp_rename 'OIB" + OIBstari + "', 'OIB" + OIBnovi + "'";
sql_zapovijed.CommandText = zapovijed;
sql_spoj.Open();
sql_zapovijed.ExecuteNonQuery();

zapovijed = "UPDATE zaposlenici SET oib = '" + OIBnovi + "' WHERE oib = '" + OIBstari +
"'";
sql_zapovijed.CommandText = zapovijed;
sql_zapovijed.ExecuteNonQuery();

zapovijed = "UPDATE koeficijenti SET oib_k = '" + OIBnovi + "' WHERE oib_k = '" +
OIBstari + "'";
sql_zapovijed.CommandText = zapovijed;
sql_zapovijed.ExecuteNonQuery();
}
catch (Exception ex)
{
throw ex;
}
finally
{
if (sql_spoj != null)
{
sql_spoj.Close();
}}}}
```

3.3.7. Klasa Evidencija

Klasa evidencija služi za izradu naredbe za spajanje na bazu podataka, slanje upita bazi podataka, prihvaćanje podataka u ugrađeni objekt C#-a klase *SqlDataAdapter* te prosljeđivanje podataka ugrađenom objektu klase *DataSet*. *DataSet* je u biti skup tablica u jednom objektu.

Klasa *Evidencija* se sastoji od parametarskog konstruktora koji prihvaća argumente za spajanje na bazu podatak, naredbu bazi podataka, OIB zaposlenika u pitanju na osnovu kojega će raditi upite bazi podataka i argumenta za upit u tablicu koeficijenti.

U klasi su definirani parametri svih podataka kako bi mogle biti izvedene naredbe *INSERT*, *UPDATE* i *DELETE* na bazi podataka.

Izgled klase *Evidencija:*

Deklariranje objekata koji će biti korišteni u klasi i zadanog konstrukora.

```
class Evidencija
{
   SqlConnection evidencija_spoj;

   SqlCommand evidencija_zapovijed;
   SqlCommand koeficijenti_zapovijed;
   SqlCommand zaposlenik_zapovijed;
   SqlCommand porez_zapovijed;

   SqlDataAdapter evidencija_adapter;
   SqlDataAdapter koeficijenti_adapter;
   SqlDataAdapter zaposlenik_adapter;
   SqlDataAdapter porez_adapter;

   DataSet evidencija_tabla;

   public Evidencija() { }
```

Izgled parametarskog konstruktora:

U parametarskom konstruktoru stvaraju se objekti za spajanje na bazu podataka, za prosljeđivanje naredbi bazi podataka kojima se prosljeđuju vrijednosti od pozivajućeg koda. Uz to stvara se objekt klase *SqlDataAdapter* koji će prihvaćati podatke iz baze podataka, prosljeđivati ih objektu klase *DataSet*, ali i vraćati promijenjene podatke bazi podataka.

Svaki objekt ima zaseban *DataAdapter* koji posreduje između baze podataka i *DataSeta-a*. Sve tablice iz baze podataka se upisuju u isti *DataSet*. *DataAdapteru* se pridružuje *SELECT* naredba.

```
public Evidencija(string spoj, string oib, string zapovijed, bool koef)
{
   evidencija_spoj = new SqlConnection(spoj);
   evidencija_zapovijed = new SqlCommand();
   evidencija_zapovijed.CommandText = zapovijed;
   evidencija_zapovijed.Connection = evidencija_spoj;
   evidencija_adapter = new SqlDataAdapter();
   evidencija_adapter.SelectCommand = evidencija_zapovijed;
```

Slijedi definiranje *UPDATE* naredbe pomoću koje će promjene u podatcima na korisničkom računalu biti sinkronizirane sa bazom podataka. U naredbu je ubačen OIB zaposlenika koji se

koristi za adresiranje pripadajuće tablice, proslijeđen kao argument konstruktoru. Na kraju naredbe se nalazi objekt u kojem je podatak za spajanje na bazu podataka.

```
SqlCommand updatezapovijed = new SqlCommand("UPDATE OIB" + oib + " SET datum = @datum,
Pocetak_rada = @Pocetak_rada, Zavrsetak_rada = @Zavrsetak_rada, Zastoj_od = @Zastoj_od,
Zastoj_do = @Zastoj_do, Terenski_rad = @Terenski_rad, Sati_pripravnosti =
@Sati_pripravnosti, Dnevni_odmor = @Dnevni_odmor, Tjedni_odmor = @Tjedni_odmor,
Godisnji_odmor = @Godisnji_odmor, Placeni_neradni = @Placeni_neradni, Bolovanje =
@Bolovanje, Trudnoca_komplikacije = @Trudnoca_komplikacije, Rodiljni_dopust =
@Rodiljni_dopust, Roditeljski_dopust = @Roditeljski_dopust, Mirovanje_rad_odnosa =
@Mirovanje_rad_odnosa, Druga_prava = @Druga_prava, Placeni_dopust = @Placeni_dopust,
Ne_placeni_dopust = @Ne_placeni_dopust, Dopusteni_izostanak = @Dopusteni_izostanak,
Ne_dopusteni_izostanak = @Ne_dopusteni_izostanak, strajk = @strajk, Iskljucenje_s_rada =
@Iskljucenje_s_rada, Nocni_redovni = @Nocni_redovni, Nocni_blagdani = @Nocni_blagdani,
Nocni_prekovremeni = @Nocni_prekovremeni, Redovan_rad = @Redovan_rad, Redovni_blagdanom =
@Redovni_blagdanom, Redovni_prekovremeni = @Redovni_prekovremeni WHERE datum = @datum",
evidencija_spoj);
```

Slijedi definiranje i dodavanje parametara *UPDATE* naredbe. Ovi parametri opisuju podatke baze podataka. Ovo je potrebno kako bi se *UPDATE* naredba ispravno izvela. Kao argumenti daju se naziv zaglavlja tablice, tip podatka u bazi podataka, dužina podatka u bazi, te naziv zaglavlja u *DataAdapteru.* Za tip podatka *int* koji nema dužinu, upisana je nula, jer naredba zahtijeva točan broj argumenata. Na kraju je objektu *DataAdapter* pridružena gornja naredba pomoću metode *UpdateCommand.*

```
updatezapovijed.Parameters.Add("@datum", SqlDbType.Date, 8, "datum");
updatezapovijed.Parameters.Add("@Pocetak_rada", SqlDbType.SmallInt, 0, "Pocetak_rada");
updatezapovijed.Parameters.Add("@Zavrsetak_rada", SqlDbType.SmallInt, 0,
"Zavrsetak_rada");
updatezapovijed.Parameters.Add("@Zastoj_od", SqlDbType.SmallInt, 0, "Zastoj_od");
updatezapovijed.Parameters.Add("@Zastoj_do", SqlDbType.SmallInt, 0, "Zastoj_do");
updatezapovijed.Parameters.Add("@Terenski_rad", SqlDbType.SmallInt, 0, "Terenski_rad");
updatezapovijed.Parameters.Add("@Sati_pripravnosti", SqlDbType.SmallInt, 0,
"Sati_pripravnosti");
updatezapovijed.Parameters.Add("@Dnevni_odmor", SqlDbType.SmallInt, 0, "Dnevni_odmor");
updatezapovijed.Parameters.Add("@Tjedni_odmor", SqlDbType.SmallInt, 0, "Tjedni_odmor");
updatezapovijed.Parameters.Add("@Godisnji_odmor", SqlDbType.SmallInt, 0,
"Godisnji_odmor");
updatezapovijed.Parameters.Add("@Placeni_neradni", SqlDbType.SmallInt, 0,
"Placeni_neradni");
updatezapovijed.Parameters.Add("@Bolovanje", SqlDbType.SmallInt, 0, "Bolovanje");
updatezapovijed.Parameters.Add("@Trudnoca_komplikacije", SqlDbType.SmallInt, 0,
"Trudnoca_komplikacije");
updatezapovijed.Parameters.Add("@Rodiljni_dopust", SqlDbType.SmallInt, 0,
"Rodiljni_dopust");
updatezapovijed.Parameters.Add("@Roditeljski_dopust", SqlDbType.SmallInt, 0,
"Roditeljski_dopust");
updatezapovijed.Parameters.Add("@Mirovanje_rad_odnosa", SqlDbType.SmallInt, 0,
"Mirovanje_rad_odnosa");
updatezapovijed.Parameters.Add("@Druga_prava", SqlDbType.SmallInt, 0, "Druga_prava");
updatezapovijed.Parameters.Add("@Placeni_dopust", SqlDbType.SmallInt, 0,
"Placeni_dopust");
updatezapovijed.Parameters.Add("@Ne_placeni_dopust", SqlDbType.SmallInt, 0,
"Ne_placeni_dopust");
```

```
updatezapovijed.Parameters.Add("@Dopusteni_izostanak", SqlDbType.SmallInt, 0,
"Dopusteni_izostanak");
updatezapovijed.Parameters.Add("@Ne_dopusteni_izostanak", SqlDbType.SmallInt, 0,
"Ne_dopusteni_izostanak");
updatezapovijed.Parameters.Add("@strajk", SqlDbType.SmallInt, 0, "strajk");
updatezapovijed.Parameters.Add("@Iskljucenje_s_rada", SqlDbType.SmallInt, 0,
"Iskljucenje_s_rada");
updatezapovijed.Parameters.Add("@Nocni_redovni", SqlDbType.SmallInt, 0, "Nocni_redovni");
updatezapovijed.Parameters.Add("@Nocni_blagdani", SqlDbType.SmallInt, 0,
"Nocni_blagdani");
updatezapovijed.Parameters.Add("@Nocni_prekovremeni", SqlDbType.SmallInt, 0,
"Nocni_prekovremeni");
updatezapovijed.Parameters.Add("@Redovan_rad", SqlDbType.SmallInt, 0, "Redovan_rad");
updatezapovijed.Parameters.Add("@Redovni_blagdanom", SqlDbType.SmallInt, 0,
"Redovni_blagdanom");
updatezapovijed.Parameters.Add("@Redovni_prekovremeni", SqlDbType.SmallInt, 0,
"Redovni_prekovremeni");
evidencija_adapter.UpdateCommand = updatezapovijed;
```

Sve isto definirano je i za *INSERT* naredbu. Na kraju je iskorištena metoda *InsertCommand.*

```
SqlCommand insertzapovijed = new SqlCommand("INSERT INTO OIB" + oib + "(datum,
Pocetak_rada, Zavrsetak_rada, Zastoj_od, Zastoj_do, Terenski_rad, Sati_pripravnosti,
Dnevni_odmor, Tjedni_odmor, Godisnji_odmor, Placeni_neradni, Bolovanje,
Trudnoca_komplikacije, Rodiljni_dopust, Roditeljski_dopust, Mirovanje_rad_odnosa,
Druga_prava, Placeni_dopust, Ne_placeni_dopust, Dopusteni_izostanak,
Ne_dopusteni_izostanak, strajk, Iskljucenje_s_rada, Nocni_redovni, Nocni_blagdani,
Nocni_prekovremeni, Redovan_rad, Redovni_blagdanom, Redovni_prekovremeni) VALUES (@datum,
@Pocetak_rada, @Zavrsetak_rada, @Zastoj_od, @Zastoj_do, @Terenski_rad,
@Sati_pripravnosti, @Dnevni_odmor, Tjedni_odmor, @Godisnji_odmor, @Placeni_neradni,
@Bolovanje, @Trudnoca_komplikacije, @Rodiljni_dopust, @Roditeljski_dopust,
@Mirovanje_rad_odnosa, @Druga_prava, @Placeni_dopust, @Ne_placeni_dopust,
@Dopusteni_izostanak, Ne_dopusteni_izostanak, @strajk, @Iskljucenje_s_rada,
@Nocni_redovni, @Nocni_blagdani, @Nocni_prekovremeni, @Redovan_rad, @Redovni_blagdanom,
@Redovni_prekovremeni)", evidencija_spoj);

insertzapovijed.Parameters.Add("@datum", SqlDbType.Date, 8, "datum");
insertzapovijed.Parameters.Add("@Pocetak_rada", SqlDbType.SmallInt, 0, "Pocetak_rada");
insertzapovijed.Parameters.Add("@Zavrsetak_rada", SqlDbType.SmallInt, 0,
"Zavrsetak_rada");
insertzapovijed.Parameters.Add("@Zastoj_od", SqlDbType.SmallInt, 0, "Zastoj_od");
insertzapovijed.Parameters.Add("@Zastoj_do", SqlDbType.SmallInt, 0, "Zastoj_do");
insertzapovijed.Parameters.Add("@Terenski_rad", SqlDbType.SmallInt, 0, "Terenski_rad");
insertzapovijed.Parameters.Add("@Sati_pripravnosti", SqlDbType.SmallInt, 0,
"Sati_pripravnosti");
insertzapovijed.Parameters.Add("@Dnevni_odmor", SqlDbType.SmallInt, 0, "Dnevni_odmor");
insertzapovijed.Parameters.Add("@Tjedni_odmor", SqlDbType.SmallInt, 0, "Tjedni_odmor");
insertzapovijed.Parameters.Add("@Godisnji_odmor", SqlDbType.SmallInt, 0,
"Godisnji_odmor");
insertzapovijed.Parameters.Add("@Placeni_neradni", SqlDbType.SmallInt, 0,
"Placeni_neradni");
insertzapovijed.Parameters.Add("@Bolovanje", SqlDbType.SmallInt, 0, "Bolovanje");
insertzapovijed.Parameters.Add("@Trudnoca_komplikacije", SqlDbType.SmallInt, 0,
"Trudnoca_komplikacije");
insertzapovijed.Parameters.Add("@Rodiljni_dopust", SqlDbType.SmallInt, 0,
"Rodiljni_dopust");
insertzapovijed.Parameters.Add("@Roditeljski_dopust", SqlDbType.SmallInt, 0,
"Roditeljski_dopust");
insertzapovijed.Parameters.Add("@Mirovanje_rad_odnosa", SqlDbType.SmallInt, 0,
"Mirovanje_rad_odnosa");
insertzapovijed.Parameters.Add("@Druga_prava", SqlDbType.SmallInt, 0, "Druga_prava");
```

```
insertzapovijed.Parameters.Add("@Placeni_dopust", SqlDbType.SmallInt, 0,
"Placeni_dopust");
insertzapovijed.Parameters.Add("@Ne_placeni_dopust", SqlDbType.SmallInt, 0,
"Ne_placeni_dopust");
insertzapovijed.Parameters.Add("@Dopusteni_izostanak", SqlDbType.SmallInt, 0,
"Dopusteni_izostanak");
insertzapovijed.Parameters.Add("@Ne_dopusteni_izostanak", SqlDbType.SmallInt, 0,
"Ne_dopusteni_izostanak");
insertzapovijed.Parameters.Add("@strajk", SqlDbType.SmallInt, 0, "strajk");
insertzapovijed.Parameters.Add("@Iskljucenje_s_rada", SqlDbType.SmallInt, 0,
"Iskljucenje_s_rada");
insertzapovijed.Parameters.Add("@Nocni_redovni", SqlDbType.SmallInt, 0, "Nocni_redovni");
insertzapovijed.Parameters.Add("@Nocni_blagdani", SqlDbType.SmallInt, 0,
"Nocni_blagdani");
insertzapovijed.Parameters.Add("@Nocni_prekovremeni", SqlDbType.SmallInt, 0,
"Nocni_prekovremeni");
insertzapovijed.Parameters.Add("@Redovan_rad", SqlDbType.SmallInt, 0, "Redovan_rad");
insertzapovijed.Parameters.Add("@Redovni_blagdanom", SqlDbType.SmallInt, 0,
"Redovni_blagdanom");
insertzapovijed.Parameters.Add("@Redovni_prekovremeni", SqlDbType.SmallInt, 0,
"Redovni_prekovremeni");
evidencija_adapter.InsertCommand = insertzapovijed;
```

Naredba za brisanje napravljena je na isti način. Kao parametar definiran je samo primarni ključ po kojem se izvodi brisanje.

```
SqlCommand brisizapovijed = new SqlCommand("DELETE FROM OIB" + oib + " WHERE datum =
@datum", evidencija_spoj);

brisizapovijed.Parameters.Add("@datum", SqlDbType.Date, 8, "datum");
evidencija_adapter.DeleteCommand = brisizapovijed;
```

Za tablice porez, koeficijenti i zaposlenici napravljena je samo *UPDATE* naredba po istom principu, jer je u njima dozvoljeno samo mijenjanje podataka, ali ne brisanje i ubacivanje. Brisanje ili ubacivanje se odvija istodobno sa unošenjem ili brisanjem zaposlenika u bazi podataka.

Prilikom izrade upita bazi podataka o određenom zaposleniku, provjerava se koristi li se osnovni unos u tablici koeficijenti ili zaposlenikov posebni. Ovo je izvedeno uz pomoć argumenta *bool* koji se predaje konstruktoru. Na osnovu vrijednosti *true* ili *false* donosi se odluka koja naredba će biti predana bazi podataka.

```
string koef_zapovijed;
if (koef == true)
{
   koef_zapovijed = "SELECT * FROM koeficijenti WHERE oib_k = 'osnova'";
}
else
{
   koef_zapovijed = "SELECT * FROM koeficijenti WHERE oib_k = '" + oib + "'";

}
```

Izvan konstruktora u klasi *Evidencija* nalaze se ugrađene funkcije. Funkcija koja puni *DataSet* podatcima iz *DataAdaptera* u koji su povučeni podatci iz baze podatka nazvana je *VratiEvidenciju.* Prihvaća dva argumenta, prvi niz znakova za spajanje na bazu podataka, drugi OIB zaposlenika za koji se vrši upit u bazi podataka. Funkcija vraća pozivajućem kodu objekt klase *DataSet.* U funkciji se stvara objekt klase *DataSet*. Pozivom ugrađene funkcije klase *DataAdapter* i predajom objekta klase *DataSet* kao argumenta funkciji i naziva tablice u bazi podataka tj. u *DataAdapteru* kao drugog argumenta izvršava se punjenje *DataSeta* podatcima. Izgled funkcije:

```
public DataSet VratiEvidenciju(string spoj, string oib)
{
   evidencija_tabla = new DataSet();
   evidencija_adapter.Fill(evidencija_tabla, "OIB" + oib);
   koeficijenti_adapter.Fill(evidencija_tabla, "koeficijenti");
   zaposlenik_adapter.Fill(evidencija_tabla, "zaposlenici");
   porez_adapter.Fill(evidencija_tabla, "porez");

   return evidencija_tabla;
}
```

Slijede četiri funkcije koje obavljaju zadatak vraćanja i sinkroniziranja podataka koji su izmijenjeni na korisničkom računalu. Funkcije kao argument prihvaćaju *DataSet* u kojem su promijenjeni podatci. Pomoću metode *update* na objektu *DataAdapter* kojoj kao prvi argument prosljeđuju objekt *DataSet* a kao drugi naziv tablice u *DataSetu*, promjene se prosljeđuju *DataAdapteru* te bazi podataka. Za svaku tablicu je napravljena ova funkcija. Izgled funkcija:

```
public void ObnoviZaposlenika(DataSet PromjenjenZapos)
{
   zaposlenik_adapter.Update(PromjenjenZapos, "zaposlenici");
}

public void obnoviPorez(DataSet PromjenaPoreza)
{
   porez_adapter.Update(PromjenaPoreza, "porez");
}

public void ObnoviPodatke(DataSet promjenjeniPodatci, string oib)
{
   evidencija_adapter.Update(promjenjeniPodatci, "OIB" + oib);
}

public void ObnoviPodatkeKoef(DataSet promjenjeniKoef)
{
   koeficijenti_adapter.Update(promjenjeniKoef, "koeficijenti");
}
```

S ovime su opisane sve klase koje program koristi u radu.

3.3.7. Izbor i obrada podataka zaposlenika

Prilikom pokretanja aplikacije prikazuje se obrazac naziva *Izbor i obrada podataka zaposlenika*. Ovaj obrazac podijeljen je na dva glavna dijela. Gornji dio za unos podataka za spajanje na bazu podataka (adresa baze podataka, korisničko ime i zaporka), spuštanje popisa zaposlenika iz baze podataka i izbor intervala za koji se žele spustiti podatci o radnom vremenu zaposlenika.

Donji dio služi za izradu baze podataka, dodavanje zaposlenika u bazu podataka, brisanje zaposlenika iz baze podataka i promjenu OIB-a zaposlenika u bazi podataka.

Kod iza obrasca za izbor i obradu podataka zaposlenika sastoji se od zaglavlja, tijela i popisa funkcija iza tijela. U zaglavlju je popis korištenih imenika, ništa nije dodavano niti brisano.

Prikaz zaglavlja:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

Ispod tijela programa nalaze se funkcije koje se pozivaju u samom programu.

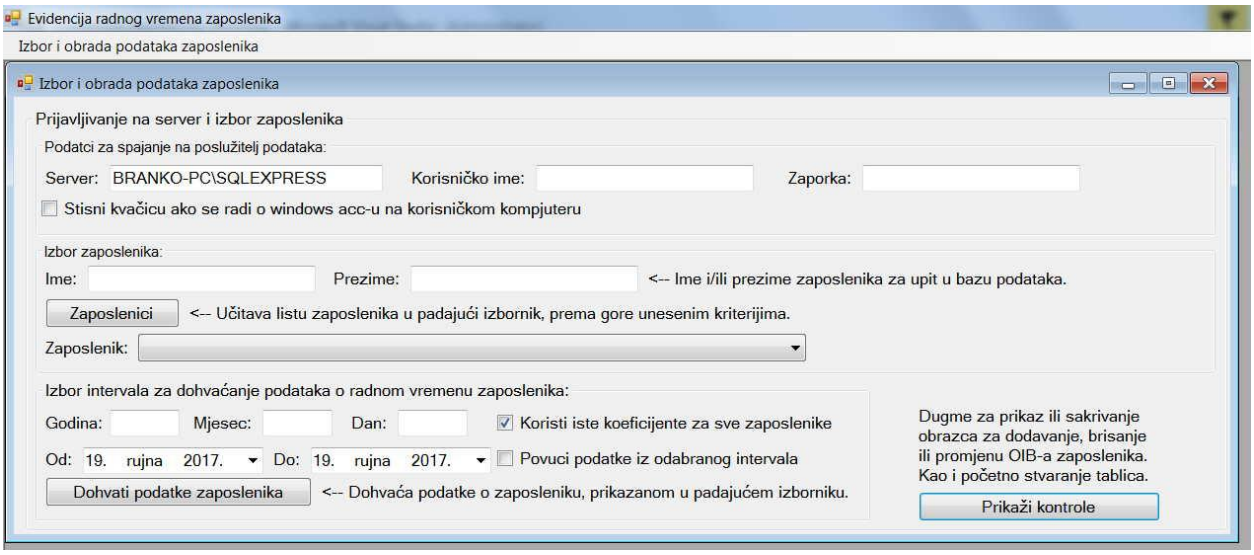
3.3.8. Funkcije obrasca *izbor i obrada podataka zaposlenika*

Funkcija za sastavljanje naredbi spajanja na bazu podataka

Funkcija naziva *spojUserPass* služi za sastavljanje naredbe za spajanje na bazu podataka uz pomoć korisničkog imena i zaporke. Funkcija ne prihvaća argumente, koristi kontrole za unos teksta sa korisničkog sučelja za sastavljanje naredbe, koje su joj dostupne jer su dio istog objekta (obrasca). Prikaz funkcije:

```
private string spojUserPass()
{
   return "Data Source=" + textBox1.Text + ";Initial Catalog=evidencija;User ID=" +
   textBox2.Text + ";Password=" + textBox3.Text;
}
```

TextBox1.Text je sadržaj iz kontrole za unos teksta *Server* sa slike 3.1. *TextBox2.Text* je sadržaj kontrole *Korisničko ime* sa slike 3.1. *TextBox3.Text* je sadržaj kontrole *Zaporka* sa slike 3.1.



Sl. 3.1. *Gornji dio obrasca izbor i obrada podataka zaposlenika.*

Funkcija naziva *spojWinAcc* služi za sastavljanje naredbe za spoj na bazu podataka, kada je baza podataka na korisničkom računalu, na istom operacijskom sustavu, te se spaja preko Windows *account-a*. Ne prihvaća argumente, koristi samo adresu baze podataka iz *TextBox1.Text* polja sa slike 3.1. naziva server. Izgled funkcije.

```
private string spojWinAcc()
{
   return "Data Source=" + textBox1.Text + ";Initial Catalog=evidencija;Integrated
   Security=True";
}
```

Funkcija za dohvaćanje liste zaposlenika

Funkcija naziva *dohvatiZaposlene* služi sastavljanju naredbe za upit bazi podataka koji će vratiti popis zaposlenika iz baze podataka i napuniti *comboBox1* naziva *Zaposlenik* sa slike 3.1. unutar *groupBox-a* sa slike 3.1. naziva *Izbor zaposlenika.* Funkcija vraća podatak tipa *string*, koji je upit bazi podataka. Prihvaća 2 argumenta, prvi iz *textBox5.Text* naziva ime, drugi iz *textBox6.Text* naziva prezime, unutar *groupBox*-a *Izbor zaposlenika.* Kod provjerava sadržaj *textBox-ova* u slučaju da je prisutno samo ime, sastavlja upit bazi podataka za sve zaposlenike tog imena, isto je s prezimenom, u slučaju prisutnosti imena i prezimena, sastavlja upit s imenom i prezimenom. Na taj način se izbjegava popunjavanje *comboBox-a* ogromnim brojem unosa. Izgled funkcije:

```
private string dohvatiZaposlene(string ime, string prezime)
{
   string zapovijed;
   if (ime != "" && prezime == "")
   {
      zapovijed = "SELECT * FROM zaposlenici WHERE ime = '" + ime + "'";
   }
   else if (ime == "" && prezime != "")
```

```
    {
        zapovijed = "SELECT * FROM zaposlenici WHERE prezime = '" + prezime + "'";
    }
    else if (ime != "" && prezime != "")
    {
        zapovijed = "SELECT * FROM zaposlenici WHERE ime = '" + ime + "' AND prezime = '" +
        prezime + "'";
    }
    else
    {
        zapovijed = "SELECT * FROM zaposlenici";
    }
    return zapovijed;
}
```

Funkcija za sastavljanje upita na osnovu *izbornika datuma i vremena*

Funkcija naziva *dateTimeDohvati* služi povlačenju podataka o radnom vremenu zaposlenika iz baze podataka iz bilo kojeg vremenskog intervala. U *groupBoxu* naziva *Izbor intervala za dohvaćanje podataka o radnom vremenu zaposlenika* prisutna su dva *dateTimePicker-a* označenih sa *od* i *do* na slici 3.1. U ovim kontrolama se izaberu datumi od kojega i do kojega se žele povući podatci, te se označi kvačica u *checkBox-u* pored, kraj kojega piše *povuci podatke iz odabranog intervala*. Funkcija slaže upit bazi podataka s obzirom na vrijednosti u *dateTimePicker-ima.* Funkcija vraća podatak tipa *string*, prihvaća tri argumenta, prvi datum od, drugi datum do i treći OIB zaposlenika. Izgled funkcije:

```
public string dateTimeDohvati(string od, string doo, string oib)
{
    string zapovijed;
    zapovijed = "SELECT * FROM OIB" + oib + " WHERE datum BETWEEN '" + od + "' AND '" +
    doo + "'";
    return zapovijed;
}
```

Funkcija za sastavljanje upita na osnovu unosa godine, mjeseca i datuma

Funkcija naziva *godinaMjesecDohvati* služi za dohvaćanje podataka o radnom vremenu zaposlenika za godinu mjesec ili dan. Funkcija vraća podatak tipa *string*, koji služi kao upit bazi podataka. Prihvaća tri argumenta tipa *string*. Prvi je godina iz *tekstBox7.Text-a* označen sa *godina* na slici 3.1., drugi je mjesec iz *textBox8.Text-a* označen sa *mjesec* na slici 3.1., treći je dan iz textBox9.Text označen sa *dan* na slici 3.1. Ako je unesena samo godina, bit će povučeni podatci za unesenu godinu. Ako su mjesec i godina uneseni bit će povučeni podatci za uneseni mjesec u unesenoj godini, na isti način i dan. Ako je unesen dan i/ili mjesec bez godine. Iskočit će *skočni prozor*, sa upozorenjem da mora biti unesena godina, da bi se mogao izabrati mjesec, isto vrijedi i za dan. Funkcija provjerava vrijednost varijable mjesec, ako je 4, 6, 9 ili 11 varijabli dan pridružuje vrijednost 30. Ako je mjesec dva i ostatak dijeljenja godine s četiri je jednako

nula, varijabli dan pridružuje vrijednost 29. Ako je mjesec dva i ostatak dijeljenja godine s četiri je različito od nula, varijabli dan pridružuje vrijednost 28. Nakon toga sastavlja upit bazi podataka za uneseni mjesec, od datuma prvog do 28, 29, 30 ili 31, u ovisnosti o unesenom mjesecu u godini i tome dali je godina prijestupna. Prikaz funkcije:

```
private string godinaMjesecDohvati(string godina, string mjesec, string dan, string oib)
{
  string zapovijed;

  if (godina != "" && mjesec == "" && dan == "")
  {
    zapovijed = "SELECT * FROM OIB" + oib + " WHERE datum BETWEEN '" + godina + "-1-1' AND
    '" + godina + "-12-31'";
  }
  else if (godina != "" && mjesec != "" && dan == "")
  {
    string dann;
    int g = Convert.ToInt32(godina);
    if (mjesec == "4" || mjesec == "6" || mjesec == "9" || mjesec == "11")
    {
      dann = "30";
    }
    else if (mjesec == "2" && (g % 4 == 0))
    {
      dann = "29";
    }
    else if (mjesec == "2" && (g % 4 != 0))
    {
      dann = "28";
    }
    else
    {
      dann = "31";
    }
    zapovijed = "SELECT * FROM OIB" + oib + " WHERE datum BETWEEN '" + godina + "-" +
    mjesec + "-1' AND '" + godina + "-" + mjesec + "-" + dann + "'";
  }
  else if (godina != "" && mjesec != "" && dan != "")
  {
    zapovijed = "SELECT * FROM OIB" + oib + " WHERE datum BETWEEN '" + godina + "-" +
    mjesec + "-" + dan + "' AND '" + godina + "-" + mjesec + "-" + dan + "'";
  }
  else
  {
    zapovijed = "SELECT * FROM OIB" + oib;
  }
  return zapovijed;
}
```

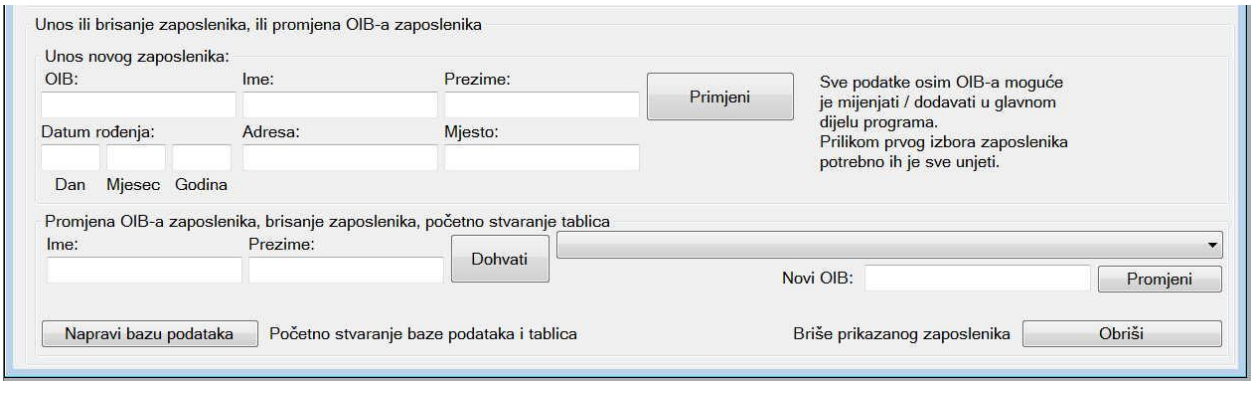
Funkcija za promjenu OIB-a zaposlenika

Funkcija naziva *promijeniOib* služi, naravno, promjeni OIB-a. Unutar funkcije se vadi OIB iz *comboBox-a2* prikazanog na slici 3.2., unutar *groupBox-a* naziva *Promjena OIB-a zaposlenika, brisanje zaposlenika, početno stvaranje tablica*, u kojem je prikazan zaposlenik za kojega se želi povući podatke iz baze podataka. Koji služi za upite u bazu podataka. OIB se vadi iz *comboBox-*

a uz pomoć metode *LastIndexOf* koja vraća cjelobrojnu vrijednost *int* mjesta znaka u nizu znakova koji je proslijeđen kao argument funkciji. Nakon toga koristi se funkcija *remove* koja prihvaća dva argumenta. Funkcija briše niz znakova od mjesta u nizu na broju prvog argumenta do mjesta u nizu na broju drugog argumenta. Ovaj OIB se zamjenjuje sa OIB-om upisanim u *textBox14.Text* stvaranjem objekta klase *PromijeniOIB* ranije opisane. Prikaz funkcije:

```
private void promijeniOib()
{
   int index;
   index = comboBox2.Text.LastIndexOf(":");

   string oibStari;
   oibStari = comboBox2.Text.Remove(0, index + 2);

   string oibNovi;
   oibNovi = textBox14.Text;

   try
   {
      PromjeniOIB promjenaOiba = new PromjeniOIB();
      promjenaOiba.promjenaOiba(oibStari, oibNovi, spojni_string);
   }
      catch (Exception ex)
   {
      MessageBox.Show("Dogodila se greška prilikom spajanja na bazu podataka ili stvaranja
      tablice!\n" + ex.Message, "Upozorenje!");
   }

   MessageBox.Show("OIB zaposlenika promijenjen uspješno!\n", "Obavijest!");
}
```

Stvaranje objekta klase *PromeniOIB* i pozivanje metode *promjenaOiba* izvedeno je u bloku *try* u slučaju ne uspjeha izvršenja u bloku *catch* otvara se skočni prozor, javlja obavijest o ne uspjehu i prosljeđuje greška od baze podataka. U slučaju uspjeha pojavljuje se skočni prozor koji obavještava korisnika o uspješno izvedenoj aktivnosti.



Sl. 3.2. *Donji dio obrasca izbor i obrada podataka zaposlenika.*

Funkcija za brisanje zaposlenika

Funkcija naziva *obrisiZaposlenika*, pretpostavljate, briše zaposlenika iz baze podataka. Na sličan način kao i prethodno opisana funkcija, vadi OIB iz niza znakova *comboBox-a* te ga upotrebljava za pronalaženje i brisanje zaposlenika. Kao i prethodna funkcija ima *try* i *catch* blokove u kojima stvara objekt klase *BrisiZaposlenog*, ranije opisano, i skočne prozore koji obavještavaju korisnika o uspjehu ili ne uspjehu.

```
private void obrisiZaposlenika()
{
   int index;
   index = comboBox2.Text.LastIndexOf(":");

   string oib3;
   oib3 = comboBox2.Text.Remove(0, index + 2);

   try
   {
      BrisiZaposlenog brisanje = new BrisiZaposlenog();
      brisanje.Obrisi(oib3, spojni_string);
   }
      catch (Exception ex)
   {
      MessageBox.Show("Dogodila se greška prilikom spajanja na bazu podataka ili brisanja
      zaposlenika!\n" + ex.Message, "Upozorenje!");
   }

   MessageBox.Show("Zaposlenik je uspješno obrisan.",
   "Obavijest");
}
```

Funkcija za izrada početnih tablica

Funkcija naziva *napraviTablice*, izrađuje bazu podataka i pripadajuće tablice. Ne vraća ništa, prihvaća jedan argument tipa *string* u kojem su podatci za spajanje na bazu podataka. Stvara objekt klase *StvoriTablice*, koji odrađuje posao. Ima skočne prozore koji javljaju uspjeh ili ne uspjeh.

```
private void napraviTablice(string spojni_string2)
{
   try
   {
      StvoriTablice stvaranje = new StvoriTablice();
      stvaranje.napraviTablice(spojni_string, spojni_string2);

      MessageBox.Show("Baza podataka i tablice su uspješno napravljene.",
      "Upozorenje!");
   }
      catch (Exception ex)
   {
      MessageBox.Show("Greška prilikom stvaranja baze podataka i/ili tablica.\n" +
      ex.Message, "Upozorenje!");
   }}
```

Funkcija za dodavanje zaposlenika u bazu podataka

Funkcija naziva *dodajZaposlenika*, dodaje novog zaposlenika u bazu podataka, sa pripadajućom tablicom i unosima u postojeće tablice. Sva polja *groupBoxa* naziva *unos novog zaposlenika* sa slike 3.2. moraju biti popunjena, inače se pojavljuje skočni prozor sa upozorenjem o nepotpunim podatcima. Ostatak podatka se popunjava kasnije. Podatci o datumu rođenja se pretvaraju u Američki način zapisa mjesec, dan godina i kao takav šalje bazi podataka. U slučaju uspjeha brišu se polja za unos podataka na korisničkom sučelju. Skočni prozor javlja o uspjehu ili ne uspjehu.

```
private void dodajZaposlenika()
{
   UnosZaposlenika UnesiZaposlenika = new UnosZaposlenika();

   try
   {
      string rodjenje = textBox19.Text + "/" + textBox18.Text + "/" + textBox15.Text;

      UnesiZaposlenika.spojiSe(spojni_string, textBox11.Text, textBox10.Text,
      textBox4.Text,
      rodjenje, textBox16.Text, textBox17.Text);

      textBox11.Text = textBox10.Text = textBox4.Text = textBox15.Text = "";
      textBox18.Text = textBox19.Text = textBox16.Text = textBox17.Text = "";

      MessageBox.Show("Zaposlenik je uspješno upisan u bazu podataka!", "Upozorenje!");
   }
   catch (Exception ex)
   {
      MessageBox.Show("Dogodila se greška prilikom spajanja na bazu podataka ili stvaranja
      tablice!\n" + ex.Message, "Upozorenje!");
   }
}
```

3.3.9. Tijelo obrasca *izbor i obrada podataka zaposlenika*

Tijelo je podijeljeno blokovima koda koji se izvode prilikom učitavanja obrasca ili pritiskom na dugme. Na početku koda samog obrasca deklarirane su varijable i objekti dostupni svim blokovima koda. Prikaz:

```
string zapovijed;
string oib;
DataSet evidencija_tabla;
List<Zaposlenik> zaposleni;
```

Nakon toga ide blok koda koji se izvršava prilikom učitavanja obrasca. *Groupbox4* je postavljen na ne vidljiv. Pritiskom na dugme *Prikaži kontrole,* od strane korisnika, postaje vidljiv. *CheckBox3* je postavljen na *checked,* ovo služi za korištenje istih koeficijenata radnih sati za sve zaposlenika. Ako to nije željeno, korisnik može maknuti oznaku na korisničkom sučelju. Prikaz:

```
private void Form2_Load(object sender, EventArgs e)
{
   groupBox4.Visible = false;
   checkBox3.Checked = true;
}
```

Dugme zaposlenici

Dugme *Zaposlenici* unutar *groupBox-a* naziva *Izbor zaposlenika* sadržava kod za povlačenje liste zaposlenika iz baze podataka. Prvo se obavlja provjera ispunjenosti potrebnih podataka za spajanje na server. Ako je polje naziva (adrese) servera različito od prazno i *checkBox1* sa slike 3.1. *Označi ako se radi o windows acc-u na korisničkom računalu,* označen varijabli zapovijed pridružuju se podatci za spajanje na bazu podataka sa Windows računom. Ako *checkBox1* nije označen i polja *Server, Korisničko ime* i *Zaporka* su različiti od prazno, varijabli zapovijed pridružuju se podatci za spajanje na bazu podataka sastavljeni od unesenih podataka. U suprotnom se prikazuje upozorenje o nepotpunim podatcima. Stvara se objekt klase *Komunikator* komu se kao argumenti predaju podatci za spajanje i upit bazi podataka. Stvara se generička lista objekata klase *Zaposlenik* kojoj se predaju rezultati upita bazi podataka. Ujedno se ovi rezultati prosljeđuju *combobox-u* zaposlenici uz pomoć prepisane funkcije *ToString* u kojoj je definiran format ispisa. Prikaz:

```
private void button1_Click(object sender, EventArgs e)
{
   if (textBox1.Text == "" && checkBox1.Checked == true) //-- u slučaju da je
   {
      MessageBox.Show("Ime servera na kojem je baza podataka nije upisano.",
      "Upozorenje!");
   }
   else if ((textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "") &&
   checkBox1.Checked == false)
   {
      MessageBox.Show("Jedno ili više potrebnih polja (ime servera, korisničko ime ili
      lozinka) je prazno!",
      "Upozorenje!");
   }
   else
   {
      if (checkBox1.Checked == true)
      {
         spojni_string = spojWinAcc();
      }
     else
      {
         spojni_string = spojUserPass();
      }
      try
      {
         zapovijed = dohvatiZaposlene(textBox5.Text, textBox6.Text);
         Komunikator zapos = new Komunikator();
         zaposleni = new List<Zaposlenik>();
         comboBox1.DataSource = zaposleni = zapos.VratiListu(spojni_string, zapovijed);
```

```
            comboBox1.DisplayMember = zapos.ToString();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Dogodila se greška prilikom spajanja na bazu podataka ili
            učitavanja podataka! \n" + ex.Message, "Izvor podataka nedostupan!");
        }
    }
}
```

Dugme za dohvaćanje podatka o radnom vremenu zaposlenika

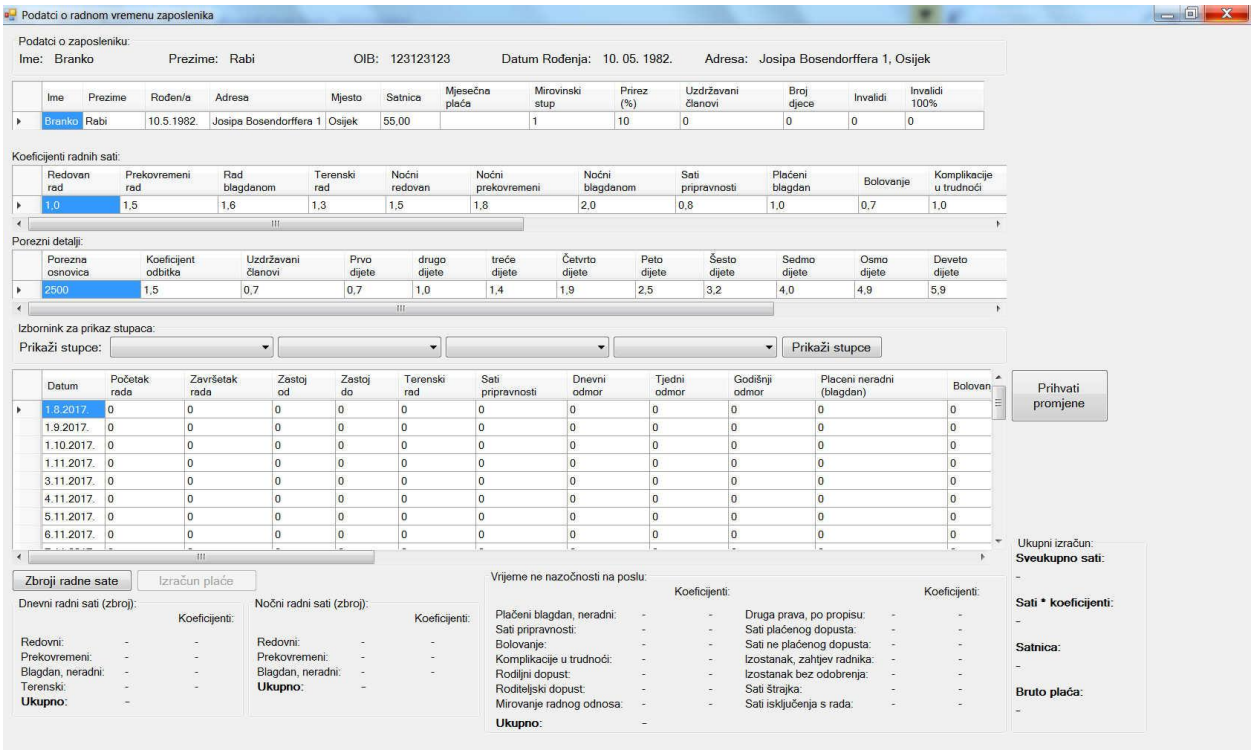
Prilikom pritiska na dugme *Dohvati podatke zaposlenika* sa slike 3.1. provjerava se stanje kontrola na grafičkom korisničkom sučelju. Ako je *combobox* u kojem treba biti prikazan željeni zaposlenik prazan, prikazuje se upozorenje. Provjeravaju se vrijednosti *textBox-ova godina, mjesec i dan* i *checkBox-ovi Koristi iste koeficijente za sve zaposlenike* i *Povuci podatke iz izabranog intervala.* Na osnovu vrijednosti u ovim kontrolama slaže se upit bazi podataka.

Ako su *textBox-ovi godina, mjesec i dan* prazni, *combobox-ovi Koristi iste koeficijente za sve zaposlenike* i *Povuci podatke iz izabranog intervala* ne označeni povlače se svi podatci o radnom vremenu zaposlenika iz tablice baze podataka. Ako je upisana godina, onda za upisanu godina, ako godina i mjesec, onda za cijeli mjesec u godini, te isto za dan.

Ako je označen *comboBox Povuci podatke iz izabranog intervala*, kao upit šalju se vrijednosti iz *timeDatePickera.*

U slučaju označenosti *comboBox-a Koristi iste koeficijente za sve zaposlenike* kao upit tablici koeficijenti umjesto OIB-a zaposlenika šalje se upit *osnova* koji služi kao općeniti unos za sve zaposlenike.

Nakon slanja upita i prihvaćanja podataka u *DataSet* izrađuje se slijedeći obrazac koji sadrži kontrole (*dataGridView-i i labele*) u koje će biti upisani podatci vraćeni od baze podataka. Izgled obrasca *podatci o radnom vremenu zaposlenika.*



Sl. 3.3. *Podatci o radnom vremenu zaposlenika.*

Nakon izrade obrasca, podatci se iz *DataSeta-a* koji je skup tablica vraćenih iz baze podataka, prosljeđuju u pripadajuće *dataGridView-e.* Podatci o zaposleniku ispisuju se na vrhu obrasca. Zaglavljima *dataGridView-a* daju se imena, tj. prepisuju se imena prenesena iz baze podataka. Na ovaj način zaglavlja mogu biti nazvana korištenjem znakova Hrvatske abecede i mogu sadržavati razmake. Prikaz:

```
private void button2_Click(object sender, EventArgs e)
{
  if (comboBox1.Text == "") // Provjera je li zaposlenik izabran.
  {
    MessageBox.Show("Zaposlenik nije izabran!", "Upozorenje!"); // Upozorenje ako nije.
  }
    else
    {
      if ((textBox7.Text == "" && (textBox8.Text != "" || textBox9.Text != "")) &&
      checkBox2.Checked == false) // Provjera jesu li potpuni podatci o intervalu za
      {                           // dohvaćanje podataka i upozorenje ako nisu.
        MessageBox.Show("Za dohvaćanje podataka iz nekog mjeseca godina mora biti
        unesena.\nZa dohvaćanje dana, mjesec i godina moraju biti uneseni.","Upozorenje!");
      }
      else
      {
        int indeks; //-Varijabla za spremanje pozicije znaka u nizu.
        indeks = comboBox1.Text.LastIndexOf(":"); // Dohvaća poziciju znaka u nizu
        (argument)
        oib = comboBox1.Text.Remove(0, indeks + 2); // Vraća OIB brišući ostatak niza
          if (checkBox2.Checked == true) // provjera označenosti checkBox-a za korištenje
          {                              // dateTimePickera
            zapovijed = dateTimeDohvati(dateTimePicker1.Value.ToString("yyyy-MM-dd"),
            dateTimePicker2.Value.ToString("yyyy-MM-dd"), oib); // dateTimedohvati funkcija
```

```
  // Koristi vrijednosti iz dateTimePickera, u argumentu je oblik u kojem se
  datum ispisuje, tj. šalje bazi podataka.
}
else // Ako nije označeno korištenje dateTimePickera koriste se unosi u textBox
{
  zapovijed = godinaMjesecDohvati(textBox7.Text, textBox8.Text, textBox9.Text,
  oib); // Zapovijed se slaže od vrijednosti u textBox-ovima.
}

bool koef;
if (checkBox3.Checked == true) // Ako je checkBox za iste koeficijente označen.
{
  koef = true; // Varijabla se postavlja na istina.
}
else
{
  koef = false; // Ako nije, ne istina.
}

try // pokušava stvoriti objekt i povući podatke iz baze podataka.
{
  Evidencija evidencija = new Evidencija(spojni_string, oib, zapovijed, koef);
  evidencija_tabla = evidencija.VratiEvidenciju(spojni_string, oib);
}
catch (Exception ex) // Ako ne uspije javlja grešku.
{
  MessageBox.Show("Povlačenje tablice s podatcima za izabranog zaposlenika nije
  uspjelo!\n" + ex.Message, "Upozorenje!"); }
Form3 obj = new Form3(); // Stvara novi obrazac.
// Punjenje dataGridView-a na obrascu 3 naziva obj podatcima. Naziv tablice u
DataSetu dan je u uglatim zagradama.
obj.dataGridView1.DataSource = evidencija_tabla.Tables["OIB" + oib];
obj.dataGridView2.DataSource = evidencija_tabla.Tables["koeficijenti"];
obj.dataGridView3.DataSource = evidencija_tabla.Tables["porez"];
obj.dataGridView4.DataSource = evidencija_tabla.Tables["zaposlenici"];

obj.evidencija_tabla = evidencija_tabla; // Prebacivanje vrijednosti varijabli na
obj.spojni_string = spojni_string;       // slijedeći obrazac.
obj.oib = oib;
obj.zapovijed = zapovijed;
```

Slijedeći blok koda provjerava svaku vrijednost ćelije u *dataGridView-u* i ako je vrijednost *null* pretvara u nulu. Razlog ovome bit će objašnjen prilikom prikaza vraćanja promijenjenih podataka u bazu podataka.

```
foreach (DataGridViewRow row in obj.dataGridView1.Rows)
{
  foreach (DataGridViewCell cell in row.Cells)
  {
    if (cell.Value == DBNull.Value)
    {
      cell.Value = 0;
    }
  }
}
```

Slijedeći kod je pridruživanje imena zaglavljima *dataGridView-a*. Isto je napravljeno i za preostala tri *dataGridView-a*.

```
// Pridruživanje imena zaglavljima dataGridView-a.
obj.dataGridView1.Columns[0].HeaderText = "Datum";
obj.dataGridView1.Columns[1].HeaderText = "Početak rada";
obj.dataGridView1.Columns[2].HeaderText = "Završetak rada";
obj.dataGridView1.Columns[3].HeaderText = "Zastoj od";
obj.dataGridView1.Columns[4].HeaderText = "Zastoj do";
obj.dataGridView1.Columns[5].HeaderText = "Terenski rad";
obj.dataGridView1.Columns[6].HeaderText = "Sati pripravnosti";
obj.dataGridView1.Columns[7].HeaderText = "Dnevni odmor";
obj.dataGridView1.Columns[8].HeaderText = "Tjedni odmor";
obj.dataGridView1.Columns[9].HeaderText = "Godišnji odmor";
obj.dataGridView1.Columns[10].HeaderText = "Placeni neradni (blagdan)";
obj.dataGridView1.Columns[11].HeaderText = "Bolovanje";
obj.dataGridView1.Columns[12].HeaderText = "Komplikacije u trudnoći";
obj.dataGridView1.Columns[13].HeaderText = "Rodiljni dopust";
obj.dataGridView1.Columns[14].HeaderText = "Roditeljski dopust";
obj.dataGridView1.Columns[15].HeaderText = "Mirovanje radnog odnosa";
obj.dataGridView1.Columns[16].HeaderText = "Druga prava (pravilnik)";
obj.dataGridView1.Columns[17].HeaderText = "Plaćeni dopust";
obj.dataGridView1.Columns[18].HeaderText = "Ne plaćeni dopust";
obj.dataGridView1.Columns[19].HeaderText = "Odobreni izostanak";
obj.dataGridView1.Columns[20].HeaderText = "Izostanak bez odobrenja";
obj.dataGridView1.Columns[21].HeaderText = "Štrajk";
obj.dataGridView1.Columns[22].HeaderText = "Isključenje s rada";
obj.dataGridView1.Columns[23].HeaderText = "Noćni redovni";
obj.dataGridView1.Columns[24].HeaderText = "Noćni blagdanom";
obj.dataGridView1.Columns[25].HeaderText = "Noćni prekovremeni";
obj.dataGridView1.Columns[26].HeaderText = "Redovan rad";
obj.dataGridView1.Columns[27].HeaderText = "Rad blagdanom";
obj.dataGridView1.Columns[28].HeaderText = "Redovni prekovremeni";
```

Slijedeći kod prolazi kroz listu zaposlenika, uspoređuje OIB, pronalazi izabranog i ispisuje podatke na vrhu obrasca.

```
foreach (Zaposlenik item in zaposleni)
{
  if (item.Oib == oib)
  {
    obj.label9.Text = item.Ime;
    obj.label11.Text = item.Prezime;
    obj.label13.Text = item.Rodjenje.ToString("dd. MM. yyyy.");
    obj.label15.Text = item.Oib;
    obj.label17.Text = item.Adresa + ", " + item.Mjesto;
  }
}

obj.koef = koef; // Prosljeđuje vrijednost varijable tipa bool slijedećem obrascu

obj.Show(); // Prikazuje slijedeći obrazac.
```

Dugme za dodavanje novog zaposlenika

Prilikom pritiska na dugme *Primjeni* unutar *groupBox-a Unos novog zaposlenika* sa slike 3.2. provjerava se prisutnost podataka u svim poljima (*OIB, Ime, Prezime, Datum rođenja, Adresa* i *Mjesto*). Ako su svi podatci prisutni poziva se funkcija *dodajZaposlenika*, ranije opisana. Ako koji podatak fali prikazuje se upozorenje. Prikaz:

```
private void button3_Click(object sender, EventArgs e)
{

  // Provjerava jesu li sva potrebna polja popunjena
  if (textBox1.Text != "" && checkBox1.Checked == true && textBox11.Text != "" &&
  textBox10.Text != "" && textBox4.Text != "" && textBox15.Text != "" && textBox16.Text
  != "" && textBox17.Text != "")
  {
    spojni_string = spojWinAcc();
    dodajZaposlenika();
  }

  // Način spajanja sa korisničkim imenom i zaporkom.
  else if (checkBox1.Checked == false && textBox1.Text != "" && textBox2.Text != ""
  && textBox3.Text != "" && textBox11.Text != "" && textBox10.Text != ""
  && textBox4.Text != "" && textBox15.Text != "" && textBox16.Text != ""
  && textBox17.Text != "")
  {
    spojni_string = spojUserPass();
    dodajZaposlenika();
  }
  else
  {
    MessageBox.Show("Provjerite dali je uneseno ime servera i svi potrebni podatci o
    zaposleniku.", "Upozorenje!");
} }
```

Dugme za promjenu OIB-a zaposlenika

Prilikom pritiska na dugme *Promjeni* unutar *groupBox-a* naziva *Promjena OIB-a zaposlenika, brisanje zaposlenika, početno stvaranje tablica* sa slike 3.2. vrši se promjena OIB-a zaposlenika. Prvo se vrši provjera prisutnosti svih informacija u potrebnim poljima te javlja greška u slučaju ne potpunosti. Nakon toga se poziva funkcija *promijeniOib* ranije opisana. Prikaz:

```
private void button6_Click(object sender, EventArgs e)
{
  if (textBox1.Text != "" && checkBox1.Checked == true && textBox14.Text != "")
  {
    spojni_string = spojWinAcc();
    promijeniOib();
  }
  else if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != ""
  && checkBox1.Checked == false && textBox14.Text != "")
  {
    spojni_string = spojUserPass();
    promijeniOib();
  }
  else
  {
    MessageBox.Show("Provjerite dali su svi potrebni podatci upisani.",
    "Upozorenje!");
  }
}
```

Dugme za brisanje izabranog zaposlenika

Prilikom pritiska na dugme *obriši* iz *groupBox-a* naziva *Promjena OIB-a zaposlenika, brisanje zaposlenika, početno stvaranje tablica* sa slike 3.2. briše se izabrani zaposlenik. Prvo se radi provjera dali je *comboBox* prazan, ako je prikazuje se upozorenje. Ako nije radi se provjera načina spajanja na bazu podataka, te se poziva funkcija *obrisiZaposlenika*. Prikaz:

```
private void button4_Click(object sender, EventArgs e)
{
  if (comboBox2.Text == "")
  {
    MessageBox.Show("Zaposlenik nije izabran!",
    "Upozorenje!");
  }
  else if (textBox1.Text != "" && checkBox1.Checked == true && comboBox2.Text != "")
  {
    spojni_string = spojWinAcc();
    obrisiZaposlenika();
  }
  else if (textBox1.Text != "" && checkBox1.Checked == false && textBox2.Text != ""
  && textBox3.Text != "" && comboBox2.Text != "")
  {
    spojni_string = spojUserPass();
    obrisiZaposlenika();
  }
}
```

Dugme za pravljenje baze podataka i početnih tablica

Dugme naziva *Napravi bazu podataka* nalazi se u *groupBoxu* naziva *Promjena OIB-a zaposlenika, brisanje zaposlenika, početno stvaranje*. Prilikom pritiska provjerava način spajanja na bazu podataka. Stvara dodatni podatak tipa *string* za spajanje na bazu podataka master, na kojoj se izrađuje baza evidencija. Poziva funkciju kojoj prosljeđuje dodatni *string*. Prikaz:

```
private void button7_Click(object sender, EventArgs e)
{
  if (textBox1.Text != "" && checkBox1.Checked == true)
  {
    spojni_string = spojWinAcc();

    string spojni_string2 = "Data Source=" + textBox1.Text +
    ";Initial Catalog=master;Integrated Security=True";

    napraviTablice(spojni_string2);
  }
  else if (textBox1.Text != "" && checkBox1.Checked == false && textBox2.Text != "" &&
  textBox3.Text != "")
  {
    spojni_string = spojWinAcc();

    string spojni_string2 = "Data Source=" + textBox1.Text + ";Initial Catalog=master;User
    ID=" +
    textBox2.Text + ";Password=" + textBox3.Text;
```

```
    napraviTablice(spojni_string2);
  }
  else
  {
    MessageBox.Show("Provjerite jesu li sva potrebna polja za spajanje na bazu podataka
    ispunjena.", "Upozorenje!");
  }
}
```

Dugme za prikaz i sakrivanje donjeg dijela obrasca

Jednostavno mijenja vidljivost *groupBox-a.*

```
private void button8_Click(object sender, EventArgs e)
{
  if (groupBox4.Visible == true)
  {
    groupBox4.Visible = false;
    button8.Text = "Prikaži";
  }
  else
  {
    groupBox4.Visible = true;
    button8.Text = "Sakrij";
  }
}
```

S ovime je završen kod obrasca.

3.3.10. Tijelo obrasca podatci o radnom vremenu zaposlenika

Zaglavlje obrasca nije dirano te izgleda jednako kao kod prijašnjeg obrasca. Na početku definirane su *informacije kulture* koje su postavljene na hrvatski. U informacijama kulture su između ostaloga informacije koji znak služi za odvajanje tisuće u broju, a koji označava decimale.

Nakon toga četiri vektora varijabli tipa *string* koje služe za punjenje izbornika mogućnostima koje korisnik može izabrati. Slijedi popis varijabli i objekata dostupnih svim blokovima koda obrasca.

```
public partial class Form3 : Form
{
CultureInfo kultura = CultureInfo.CreateSpecificCulture("hr-HR");

string[] stupci1 = { "", "Redovan_rad", "Redovni_prekovremeni", "Nocni_redovni",
"Redovni_blagdanom", "Terenski_rad", "Nocni_prekovremeni", "Nocni_blagdani", "Zastoj_od",
"Zastoj_do", "Dnevni_odmor", "Tjedni_odmor", "Godisnji_odmor", "Placeni_neradni",
"Bolovanje", "Sati_pripravnosti", "Placeni_dopust", "Rodiljni_dopust",
"Roditeljski_dopust", "Trudnoca_komplikacije", "Mirovanje_rad_odnosa",
"Dopusteni_izostanak", "Ne_dopusteni_izostanak", "Ne_placeni_dopust",
"Iskljucenje_s_rada", "strajk", "Druga_prava" };

string[] stupci2 = { "", "Redovan_rad", "Redovni_prekovremeni", "Nocni_redovni",
"Redovni_blagdanom", "Terenski_rad", "Nocni_prekovremeni", "Nocni_blagdani", "Zastoj_od",
```

```
"Zastoj_do", "Dnevni_odmor", "Tjedni_odmor", "Godisnji_odmor", "Placeni_neradni",
"Bolovanje", "Sati_pripravnosti", "Placeni_dopust", "Rodiljni_dopust",
"Roditeljski_dopust", "Trudnoca_komplikacije", "Mirovanje_rad_odnosa",
"Dopusteni_izostanak", "Ne_dopusteni_izostanak", "Ne_placeni_dopust",
"Iskljucenje_s_rada", "strajk", "Druga_prava" };

string[] stupci3 = { "", "Redovan_rad", "Redovni_prekovremeni", "Nocni_redovni",
"Redovni_blagdanom", "Terenski_rad", "Nocni_prekovremeni", "Nocni_blagdani", "Zastoj_od",
"Zastoj_do", "Dnevni_odmor", "Tjedni_odmor", "Godisnji_odmor", "Placeni_neradni",
"Bolovanje", "Sati_pripravnosti", "Placeni_dopust", "Rodiljni_dopust",
"Roditeljski_dopust", "Trudnoca_komplikacije", "Mirovanje_rad_odnosa",
"Dopusteni_izostanak", "Ne_dopusteni_izostanak", "Ne_placeni_dopust",
"Iskljucenje_s_rada", "strajk", "Druga_prava" };

string[] stupci4 = { "", "Redovan_rad", "Redovni_prekovremeni", "Nocni_redovni",
"Redovni_blagdanom", "Terenski_rad", "Nocni_prekovremeni", "Nocni_blagdani", "Zastoj_od",
"Zastoj_do", "Dnevni_odmor", "Tjedni_odmor", "Godisnji_odmor", "Placeni_neradni",
"Bolovanje", "Sati_pripravnosti", "Placeni_dopust", "Rodiljni_dopust",
"Roditeljski_dopust", "Trudnoca_komplikacije", "Mirovanje_rad_odnosa",
"Dopusteni_izostanak", "Ne_dopusteni_izostanak", "Ne_placeni_dopust",
"Iskljucenje_s_rada", "strajk", "Druga_prava" };

public string spojni_string;
public string oib;
public string zapovijed;
public DataSet evidencija_tabla;
```

Definirana su četiri ista vektora iz razloga što ako se definira jedan vektor čije se vrijednosti pridruže četirima različitim padajućim menijima, dogodi se da prilikom izbora vrijednosti u jednom, ista vrijednost se prikaže u svima.

```
// Varijable sati
object sumaRedRad;
object sumPrekovremeni;
object sumRedovniBlagdan;
object sumTerenski;
object sumNocniRedovni;
object sumNocniPrekovremeni;
object sumNocniBlagdanom;
object sumPlaceniNeradni;
object sumBolovanje;
object sumKompliUTrudnoci;
object sumRodiljniDopust;
object sumRoditeljskiDopust;
object sumMirovanjeRadOdnosa;
object sumDrugaPrava;
object sumNePlaceniDopust;
object sumIzostanakZahtjev;
object sumPlaceniDopust;
object sumIzostanakBezOdobr;
object sumStrajk;
object sumIskljucenjeRad;
object sumSatiPripravnosti;

// Varijable koeficijenata.
object redovan_rad;
object prekovremeni_rad;
object rad_blagdanom;
object terenski_rad;
object nocni_redovni;
```

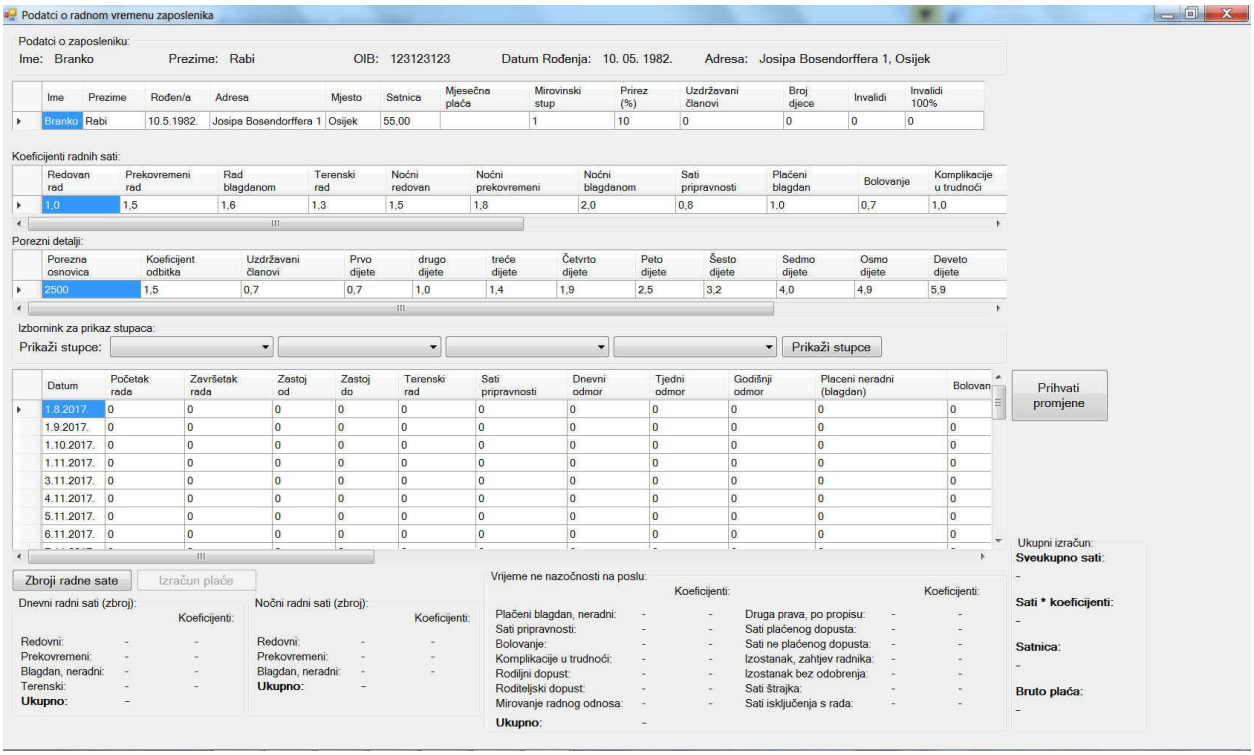
```
object nocni_prekovremeni;
object nocni_blagdan;
object placeni_blagdan;
object sati_pripravnosti;
object bolovanje;
object kompl_u_trud;
object rodiljni_dopust;
object roditeljski_dopust;
object mirovanje_rada;
object druga_prava;
object placeni_dopust;
object ne_placeni_dopust;
object izostanak_zahtjevani;
object izostanak_ne_opravdani;
object strajk;
object iskljucenje_s_rada;
object satnica;

double bruto_placa;
double suma_produkt;

        // Sume dnevnih, noćnih i sati ne nazočnosti.
double sumNocniSati;
double sveukupno;
double sumDnevniSati;
double SumSatiNeNazocnosti;

public bool koef;
```

Deklarirane su varijable tipa *object* jer se jedino u njih mogu prepisati podatci iz *dataGridView-a.*

Prilikom samog učitavanja forme, gore prikazani vektori tipa *string* pridružuju se padajućim izbornicima. Koji služe izboru stupaca *dataGridView-a* koje korisnik želi prikazane.

```
private void Form3_Load(object sender, EventArgs e)
{
  comboBox2.DataSource = stupci1;
  comboBox3.DataSource = stupci2;
  comboBox4.DataSource = stupci3;
  comboBox1.DataSource = stupci4;
}
```

Izgled obrasca prikazan je na sljedećoj slici.



Sl. 3.3. *Izgled obrasca podatci o radnom vremenu zaposlenika.*

Na samom vrhu su osnovni podatci o zaposleniku čiji se podatci obrađuju na slici 3.3. Ispod u *dataGridView-u* su osnovni podatci ali i podatci potrebni za izračun plaće, doprinosa, poreza i prireza. U *dataGridView-u* moguća je promjena podataka. Sinkronizacija sa bazom podatka se obavlja pritiskom na dugme *Prihvati promjene* sa slike 3.3.

DataGridView ispod označen sa koeficijenti radnih sati, slika 3.3. Pokazuje koeficijente vrijednosti određene vrste radnih sati. Moguće je napraviti promjene i spremiti ih u bazu podataka pritiskom na dugme *Prihvati promjene.*

Za *dataGridView* ispod označen sa porezni detalji slika 3.3., vrijedi sve isto kao za prethodna dva.

Za ova tri dataGridView-a onemogućeno je dodavanje unosa i brisanje unosa. Moguće su samo promjene.

Četvrti *dataGridView* pokazuje podatke o radnom vremenu zaposlenika. Ima iznad sebe pridružen *izbornik za prikaz stupaca* slika 3.3. Pomoću ovoga izbornika moguće je izabrati do četiri stupca koji će se prikazati. Stupaca ima 29, što može biti nezgodno prilikom unosa podataka od strane korisnika. U ovom *dataGridView-u* moguće su promjene, ubacivanje i brisanje podataka. Promjene se spremaju na bazu podataka prilikom pritiska na dugme *Prihvati promjene.*

Ispod četvrtog *dataGridView-a* nalaze se dva dugmeta naziva *Zbroji radne sate* i *Izračun plaće,* slika 3.3. Prilikom pritiska na dugme *Zbroji radne sate* zbrajaju se vrijednosti svakog stupca *dataGridView-a* u kojemu su prikazani radni sati. Ove sume množe se sa pripadajućim koeficijentima, zbrajaju, množe sa satnicom kako bi se dobila bruto plaća te se sve to prikazuje ispod *dataGridView-a,* slika 3.3. Nakon toga dugme *Izračun plaće* postaje omogućeno.

Pritiskom na dugme *Izračun plaće* otvara se slijedeći obrazac u kojem je izračun plaće, doprinosa, poreza i prireza.

Dugme za prikazivanje željenih stupaca

U padajućim izbornicima označenim sa *Prikaži stupce* slika 3.3. izaberu se redom, jedan ili više, stupaca koje korisnik želi prikazane te se pritisne dugme *Prikaži stupce.* Kod radi provjeru, ako je prvi padajući izbornik prazan svi stupci biti će prikazani. Ako nije bit će prikazan izabrani stupac. Ako je više stupaca izabrano, bit će prikazani izabrani stupci. Prva tri stupca *Datum, Početak rada* i *Završetak rada* uvijek su prikazani. Ako korisnik želi opet vidjeti sve stupce, dovoljno je da samo u prvom padajućem izborniku izabere *prazno,* i stisne dugme *Prikaži stupce.* Zahtjev je da padajući izbornici budu popunjavani redom, u suprotnom ispisuje se poruka koja to i kaže. Prikaz koda:

```
private void button4_Click(object sender, EventArgs e)
{
  if (comboBox2.Text == "") // Ako je prvi padajući izbornik prazan prikaži sve stupce.
  {
    for (int i = 3; i < 29; i++) // Petlja prolazi kroz sve stupce i čini ih vidljivima.
    {
      dataGridView1.Columns[i].Visible = true;
    }
  }

  else // ako prvi padajući izbornik nije prazna, svi stupci se učine ne vidljivima.
  {
    for (int i = 3; i < 29; i++)
    {
      dataGridView1.Columns[i].Visible = false;
    }

    if (comboBox2.Text != "" && comboBox3.Text == "" && comboBox4.Text == "" &&
    comboBox1.Text == "") // Ako je samo u prvom padajućem izborniku prisutna informacija
    {                     // samo jedan stupac se učini vidljivim.
      dataGridView1.Columns[comboBox2.Text].Visible = true;
    }
    else if (comboBox2.Text != "" && comboBox3.Text != "" && comboBox4.Text == "" &&
    comboBox1.Text == "") // ako je u prva dva prisutna informacija dva se učine
    {                     // vidljivima itd.
      dataGridView1.Columns[comboBox2.Text].Visible = true;
      dataGridView1.Columns[comboBox3.Text].Visible = true;
    }
```

```
    else if (comboBox2.Text != "" && comboBox3.Text != "" && comboBox4.Text != "" &&
    comboBox1.Text == "")
    {
      dataGridView1.Columns[comboBox2.Text].Visible = true;
      dataGridView1.Columns[comboBox3.Text].Visible = true;
      dataGridView1.Columns[comboBox4.Text].Visible = true;
    }
    else if (comboBox2.Text != "" && comboBox3.Text != "" && comboBox4.Text != "" &&
    comboBox1.Text != "")
    {
      dataGridView1.Columns[comboBox2.Text].Visible = true;
      dataGridView1.Columns[comboBox3.Text].Visible = true;
      dataGridView1.Columns[comboBox4.Text].Visible = true;
      dataGridView1.Columns[comboBox1.Text].Visible = true;
    }
    else
    { // Ako ni jedan uvjet nije ispunjen ispisuje se poruka.
      MessageBox.Show("Unosite redom stupce koje želite vidjeti!",
      "Upozorenje!");

      for (int i = 3; i < 29; i++)
      {
        dataGridView1.Columns[i].Visible = true;
      }
    }
  }
}
```

Dugme prihvati promjene

Pritiskom na dugme *Prihvati promjene* spremaju se promjene u bazu podataka. Kod prvo radi provjeru dali je podatak mirovinski stup jedan ili dva, ako nije javlja da mirovinski stup može biti samo jedan ili dva. Nakon toga stvara objekt klase Evidencija, kao argumente daje podatke za spoj na bazu podataka, OIB, naredba bazi podataka, jesu li su koeficijenti opći ili samo za pojedinog zaposlenika.

Nakon toga ide petlja koja u *dataGridView-u* na sva mjesta u kojima je vrijednost null, stavlja nulu. Međutim ovo ne radi uvijek kako treba. Te je iz tog razloga prilikom učitavanja *dataGridView-a* na obrascu dva ista petlja napravljena. Nakon toga u bloku *try* pozivaju se metode klase Evidencija za obnovu podataka te na kraju naredba *AcceptChanges* na *DataSet*, koji označava da nema više promijenjenih podataka u *DataSetu.* S obzirom na uspješnost ili ne uspješnost obavještava se korisnik prikladnim skočnim prozorom. Prikaz:

```
private void button3_Click(object sender, EventArgs e)
{
  object mirovinski_stup = dataGridView4[8, 0].Value;
  int mir_stup = 0;
  mir_stup = Convert.ToInt16(mirovinski_stup);

  if (mir_stup < 1 || mir_stup > 2 )
  {
    MessageBox.Show("Mirovinski stup može biti samo 1 ili 2",
```

```
    "Upozorenje!");
  }
  else
  {
    Evidencija evid = new Evidencija(spojni_string, oib, zapovijed, koef);

    foreach (DataGridViewRow row in dataGridView1.Rows)
    {
      foreach (DataGridViewCell cell in row.Cells)
      {
        if (cell.Value == DBNull.Value)
        {
          cell.Value = 0;
        }
      }
    }

  try
  {
    evid.ObnoviPodatkeKoef(evidencija_tabla.GetChanges());
    evid.ObnoviZaposlenika(evidencija_tabla.GetChanges());
    evid.obnoviPorez(evidencija_tabla.GetChanges());
    evid.ObnoviPodatke(evidencija_tabla.GetChanges(), oib);
    evidencija_tabla.AcceptChanges();

    MessageBox.Show("Promjene uspješno pohranjene u bazu podataka.",
    "Obavijest!");
  }
  catch (Exception ex)
  {
    MessageBox.Show("Došlo je do pogreške prilikom upisivanja podataka u bazu
    podataka!!!\n" +
    ex.Message, "Upozorenje!");
  }
  }
}
```

Dugme zbroji radne sate

Dugme zbraja radne sate zaposlenika, množi sa pripadajućim koeficijentima pa sa satnicom da dobije bruto plaću te sve to ispisuje u područje ispod *dataGridView-a* za prikaz radnih sati zaposlenika slika 3.3.

Kod prvo provjerava prisutnost svih potrebnih podataka u *dataGridView-u* koji prikazuje podatke o zaposleniku, slika 3.3. Ako nisu svi podatci prisutni izbacuje poruku korisniku da ih popuni.

```
private void button5_Click(object sender, EventArgs e)
{
  if (dataGridView4[11, 0].Value == DBNull.Value || dataGridView4[12, 0].Value ==
  DBNull.Value || dataGridView4[13, 0].Value == DBNull.Value || dataGridView4[9, 0].Value
  == DBNull.Value || dataGridView4[8, 0].Value == DBNull.Value)
  {
    MessageBox.Show("Jedna ili više potrebnih informacija nije dostupna (satnica, broj
    uzdržavane\ndjece, uzdržavani članovi obitelji, broj uzdržavanih invalida).\nMolim vas
    unesite potrebne informacije.", "Upozorenje!");
  }
```

```
else
{
  sumaRedRad = evidencija_tabla.Tables["OIB" + oib].Compute("Sum(Redovan_rad)", "");
  label24.Text = sumaRedRad.ToString();
```

Zadnje dvije linije prikazuju zbrajanje svih vrijednosti stupca u *DataSetu*. U uglatim zagradama dan je naziv tablice u datasetu za koju se radi zbrajanje. Zatim se poziva metoda *compute* objekta klase *DataSet* kojoj se kao argument daje naredba sum i u zagradi naziv stupca za koji se radi zbrajanje. Rezultat se prosljeđuje varijabli. Ova naredba napravljena je za svaki stupac koji sadržava podatak o radnim satima zaposlenika.

```
  sumDnevniSati = Convert.ToDouble(sumaRedRad) + Convert.ToDouble(sumPrekovremeni) +
  Convert.ToDouble(sumRedovniBlagdan) + Convert.ToDouble(sumTerenski);
  label23.Text = sumDnevniSati.ToString();
```

Gornje linije prikazuju sumiranje svih dnevnih sati (redovni, prekovremeni, terenski). Vrši se konverzija u tip *double*, jer sa varijablama tipa *object* nije moguće vršiti računske operacije. Isto se radi za noćne radne sate i sate ne nazočnosti na poslu.

```
  redovan_rad = dataGridView2[1, 0].Value;
  label25.Text = Convert.ToString(redovan_rad);
```

Gornje linije prikazuju dohvaćanje koeficijenata vrijednosti radnih sati i njihovo upisivanje u kontrole na području ispod *dataGridView-a* za prikaz radnih sati zaposlenika, slika 3.3. Ovo je napravljeno za sve koeficijente.

```
suma_produkt = Convert.ToDouble(sumaRedRad) * Convert.ToDouble(redovan_rad) +
  Convert.ToDouble(sumPrekovremeni) * Convert.ToDouble(prekovremeni_rad) +
  Convert.ToDouble(sumRedovniBlagdan) * Convert.ToDouble(rad_blagdanom) +
  Convert.ToDouble(sumTerenski) * Convert.ToDouble(terenski_rad) +
  Convert.ToDouble(sumNocniRedovni) * Convert.ToDouble(nocni_redovni) +
  Convert.ToDouble(sumNocniPrekovremeni) * Convert.ToDouble(nocni_prekovremeni) +
  Convert.ToDouble(sumNocniBlagdanom) * Convert.ToDouble(nocni_blagdan) +
  Convert.ToDouble(sumPlaceniNeradni) * Convert.ToDouble(placeni_blagdan) +
  Convert.ToDouble(sumSatiPripravnosti) * Convert.ToDouble(sati_pripravnosti) +
  Convert.ToDouble(sumBolovanje) * Convert.ToDouble(bolovanje) +
  Convert.ToDouble(sumKompliUTrudnoci) * Convert.ToDouble(kompl_u_trud) +
  Convert.ToDouble(sumRodiljniDopust) * Convert.ToDouble(rodiljni_dopust) +
  Convert.ToDouble(sumRoditeljskiDopust) * Convert.ToDouble(roditeljski_dopust) +
  Convert.ToDouble(sumMirovanjeRadOdnosa) * Convert.ToDouble(mirovanje_rada) +
  Convert.ToDouble(sumDrugaPrava) * Convert.ToDouble(druga_prava) +
  Convert.ToDouble(sumPlaceniDopust) * Convert.ToDouble(placeni_dopust) +
  Convert.ToDouble(sumNePlaceniDopust) * Convert.ToDouble(ne_placeni_dopust) +
  Convert.ToDouble(sumIzostanakZahtjev) * Convert.ToDouble(izostanak_zahtjevani) +
  Convert.ToDouble(sumIzostanakBezOdobr) * Convert.ToDouble(izostanak_ne_opravdani) +
  Convert.ToDouble(sumStrajk) * Convert.ToDouble(strajk) +
  Convert.ToDouble(sumIskljucenjeRad) * Convert.ToDouble(iskljucenje_s_rada);
```

Kod iznad prikazuje množenje pojedine vrste radnih sati sa pripadajućim koeficijentima te međusobno zbrajanje. Vrši se konverzija u tip *double*.

```
label47.Text = Convert.ToString(suma_produkt);

satnica = dataGridView4[6, 0].Value;
double satnica1 = Convert.ToDouble(satnica);
label50.Text = satnica1.ToString("N2") + " kn";

bruto_placa = Math.Round(suma_produkt * satnica1);
label4.Text = bruto_placa.ToString("N2") + "kn";

button1.Enabled = true;
```

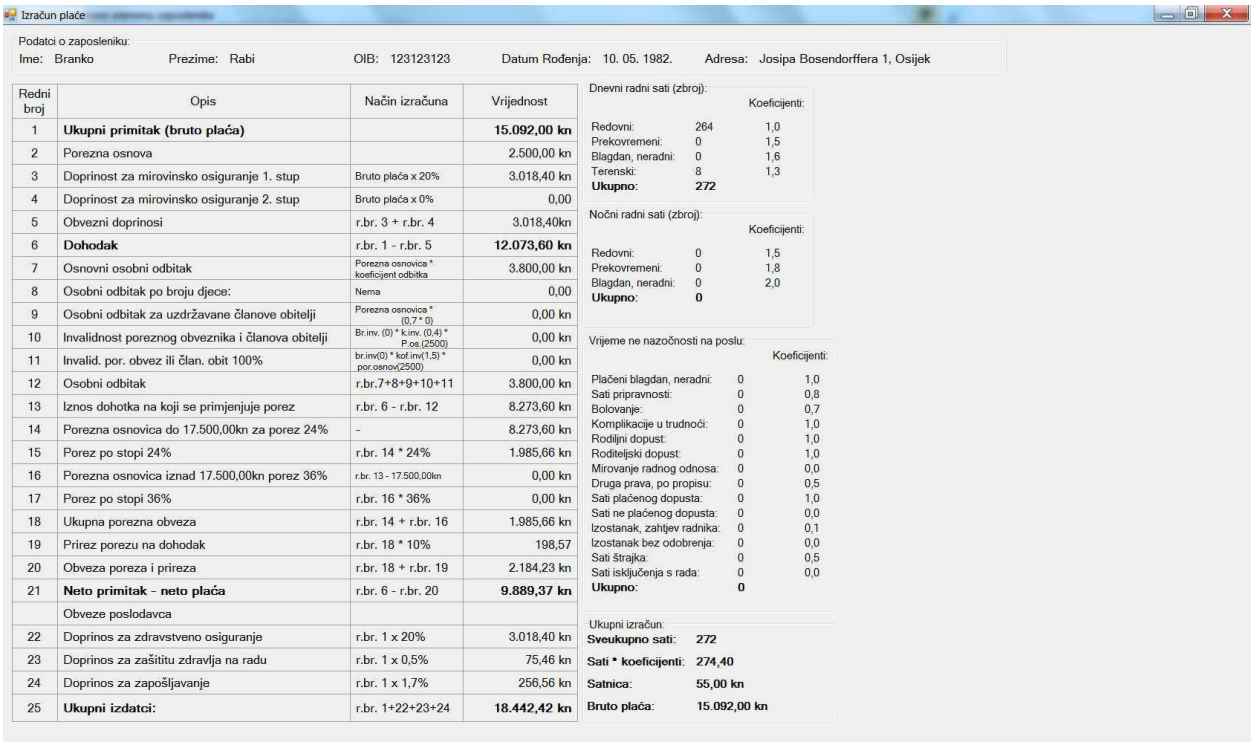
Zatim se ispisuje zbroj umnožaka u obrazac. Dohvaća se satnica, pretvara u tip *double*, ali se stvara nova varijabla. Računa se plaća množenjem satnice sa zbrojem umnožaka radnih sati i pripadajućih koeficijenata. Ispis satnice i bruto plaće radi se uz pomoć funkcije *ToString* kojoj se kao argumenti šalju *„N2“* što zapravo znači da ispis bude prema lokalnim pravilima, tisućica se odvaja točkom, decimale zarezom i zaokružuje se na dva decimalna mjesta. Na kraju se omogućava dugme *Izračun plaće*.

Dugme izračun plaće

Dugme *Izračun plaće* izrađuje sljedeći obrazac naziva *Izračun plaće*, radi sve izračune, prenosi ih na sljedeći obrazac i prikazuje ga.

```
private void button1_Click(object sender, EventArgs e)
{
   Form4 form4 = new Form4();

   form4.label9.Text = label9.Text;
   form4.label11.Text = label11.Text;
   form4.label15.Text = label15.Text;
   form4.label13.Text = label13.Text;
   form4.label17.Text = label17.Text;

   form4.label24.Text = sumaRedRad.ToString();
   form4.label20.Text = sumPrekovremeni.ToString();
   form4.label21.Text = sumRedovniBlagdan.ToString();
   form4.label22.Text = sumTerenski.ToString();
```

Kod iznad prikazuje prepisivanje podatka iz trenutnog obrasca u slijedeći. Ovo je samo dio. Isto je napravljeno za sve varijable sa trenutnog obrasca na slijedeći. Sve informacije sa dna trenutnog obrasca prepisane su na sljedeći te se nalaze s lijeve strane, slika 3.4.



Sl. 3.4. *Obrazac izračun plaće.*

Na lijevoj strani slike 3.4. su prikazani prepisani podatci sa obrasca *Podatci o radnom vremenu zaposlenika.*

```
if (form4.mirovinski_stup == 1)
{
  form4.label97.Text = "Bruto plaća x 20%";
  form4.dop_za_mir1 = bruto_placa * 0.2;
  form4.label103.Text = form4.dop_za_mir1.ToString("N2") + " kn";
  form4.label102.Text = "Bruto plaća x 0%";
  form4.label104.Text = "0,00";
  form4.dop_za_mir2 = 0;
}
else if (form4.mirovinski_stup == 2)
{
  form4.label97.Text = "Bruto plaća x 15%";
  form4.dop_za_mir1 = bruto_placa * 0.15;
  form4.label103.Text = form4.dop_za_mir1.ToString("N2") + " kn";
  form4.label102.Text = "Bruto plaća x 5%";
  form4.dop_za_mir2 = bruto_placa * 0.05;
  form4.label104.Text = form4.dop_za_mir2.ToString("N2") + " kn";
}
else
{
  MessageBox.Show("Greška u podatku mirovinskog osiguranja.",
  "Upozorenje!");
}
```

Kod iznad provjerava vrijednost varijable mirovinski stup. Ako je jedan, upisuje u tablicu na slici 3.4. pod rednim brojem tri, u stupac način izračuna *Bruto plaća x 20%,* te u stupac

vrijednost izračun (20% od bruto plaće). Svi ispisi vrijednosti se zaokružuju na dvije decimale i tisućice su odvojene točkom.

Ako je vrijednost mirovinskog stupa dva u redak s rednim brojem tri, u stupac način izračuna upisuje se *bruto plaća x 15%,* a u isti stupac sa rednim brojem četiri *bruto plaća x 5%.* Izračunavaju se vrijednosti i upisuju u stupac *vrijednost* pripadajućeg retka.

```
form4.label108.Text = (form4.dop_za_mir1 + form4.dop_za_mir2).ToString("N2") + "kn";
```

Gornja linija izvršava zbrajanje doprinosa za prvi i drugi stup, i prikazuje ih u tablici na slici 3.4. u retku pod brojem pet u stupcu *vrijednost.*

```
form4.dohodak = form4.bruto_placa - form4.dop_za_mir1 - form4.dop_za_mir2;
form4.label112.Text = form4.dohodak.ToString("N2") + " kn";
```

Izračunava se dohodak, oduzimanjem doprinosa od bruto plaće i prikazuje u tablici u retku broj šest, u stupcu *vrijednost.*

```
form4.osnovni_os_odbitak = Convert.ToDouble(Math.Round(form4.por_osnovica *
form4.koef_odbitka / 100) * 100);
form4.label116.Text = form4.osnovni_os_odbitak.ToString("N2") + " kn";
```

Izračunava se osnovni osobni odbitak, zaokružen na 100. Množenjem osnovice osobnog odbitka sa koeficijentom osobnog odbitka. Dijeli se sa 100, zaokružuje na cjelobrojnu vrijednost te množi sa sto.

Slijedeći blok koda računa odbitak ostvaren na osnovu uzdržavanog broja djece. Koeficijenti se ne računaju nego uzimaju iz tablice *porez.*

```
if (form4.br_djece == 0)
{
  form4.label187.Text = "Nema";
  form4.odbitak_djece = 0;
  form4.label192.Text = "0,00";
}
else if (form4.br_djece == 1)
{
  form4.label187.Text = "Porezna osnovica * " + form4.prvo_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.prvo_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 2)
{
  form4.label187.Text = "Porezna osnovica * " + form4.drugo_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.drugo_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 3)
{
  form4.label187.Text = "Porezna osnovica * " + form4.trece_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.trece_dijete;
```

```
  form4.label192.Text = form4.odbitak_djece.ToString("N") + " kn";
}
else if (form4.br_djece == 4)
{
  form4.label187.Text = "Porezna osnovica * " + form4.cetvrto_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.cetvrto_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 5)
{
  form4.label187.Text = "Porezna osnovica * " + form4.peto_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.peto_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 6)
{
  form4.label187.Text = "Porezna osnovica * " + form4.sesto_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.sesto_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 7)
{
  form4.label187.Text = "Porezna osnovica * " + form4.sedmo_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.sedmo_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 8)
{
  form4.label187.Text = "Porezna osnovica * " + form4.osmo_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.osmo_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
else if (form4.br_djece == 9)
{
  form4.label187.Text = "Porezna osnovica * " + form4.deveto_dijete;
  form4.odbitak_djece = form4.por_osnovica * form4.deveto_dijete;
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
```

Kod iznad provjerava vrijednost varijable *br_djece* u kojoj je broj uzdržavane djece zaposlenika. Na osnovu ovog broja izabire pripadajući koeficijent i množi s osnovicom osobnog odbitka. U redak 8 tablice sa slike 3.4. stupac način izračuna upisuje *nema* ako nema uzdržavane djece. Ako ima upisuje *porezna osnovica * vrijednost pripadajućeg koeficijenta.* A u stupac vrijednost umnožak tog dvoje.

```
else if (form4.br_djece > 9)
{
  form4.label187.Text = "Porezna osnovica *\n(" + form4.deveto_dijete +
  " + (" + (form4.br_djece - 9) + " * " + form4.slijedece_dijete + "))";

  form4.odbitak_djece = form4.por_osnovica * (form4.deveto_dijete +
  ((form4.br_djece - 9) * form4.slijedece_dijete));
  form4.label192.Text = form4.odbitak_djece.ToString("N2") + " kn";
}
```

Ako zaposlenik ima više od devet djece onda se koeficijent uvećava za 1,1 za svako slijedeće dijete. Formula:

$$P_{OD} * (K_9 + (B_D - 9) * K_{>9}) \quad (3\text{-}1)$$

P_{OD} – Osnovica osobnog odbitka

K_9 – Koeficijent za devet uzdržavane djece

B_D – Broj uzdržavane djece

$K_{>9}$ – Koeficijent za svako slijedeće dijete – iznosi 1,1

Slijedi odbitak za uzdržavane članove

```
form4.label119.Text = "Porezna osnovica *\n" + "(" + form4.uzdrzavan_obitelj +
" * " + form4.uzdrzavani_clanovi + ")";
form4.odbitak_uzdrzavani = (form4.por_osnovica * form4.uzdrzavan_obitelj *
form4.uzdrzavani_clanovi);
form4.label120.Text = form4.odbitak_uzdrzavani.ToString("N2") + " kn";
```

U tablicu sa slike 3.4. redak devet, stupac *Način izračuna* upisuje se *Porezna osnovica* * vrijednost varijable *broj uzdržavanih članova* * vrijednost *koeficijenta.* U stupac *Vrijednost* upisuje se izračun.

Odbitak u slučaju invalidnosti poreznog obveznika ili uzdržavanih članova

```
form4.odbitak_invalidi = (form4.invalidi * form4.invalidnost * form4.por_osnovica);
form4.label123.Text = "Br.inv. (" + form4.invalidi + ") * k.inv. (" + form4.invalidnost +
") * P.os.(" + form4.por_osnovica + ")";
form4.label124.Text = form4.odbitak_invalidi.ToString("N2") + " kn";
```

Opet u redak 10 sa slike 3.4. upisuje se način izračuna, sa konkretnim vrijednostima, te izračunata vrijednost.

U slučaju 100%-tne invalidnosti

```
form4.odbitak_invalidi_100 = (form4.invalidnost_100 * form4.invalidi_100 *
form4.por_osnovica);
form4.label155.Text = "br.inv(" + form4.invalidi_100 + ") * kof.inv(" +
form4.invalidnost_100 + ") *\n por.osnov(" + form4.por_osnovica + ")";
form4.label156.Text = form4.odbitak_invalidi_100.ToString("N") + " kn";
```

Popunjava se redak 11 po istom principu.

Osobni odbitak predstavlja zbroj svih gore navedenih odbitaka. U ovom slučaju se stupac način izračuna ne popunjava. Nego stoji popunjen te u njemu piše koje stavke iz tablice se zbrajaju. U stupac vrijednost retka 12, upisuje se zbroj.

```
form4.osobni_odbitak = (form4.osnovni_os_odbitak + form4.odbitak_djece +
form4.odbitak_uzdrzavani + form4.odbitak_invalidi + form4.odbitak_invalidi_100);
form4.label128.Text = form4.osobni_odbitak.ToString("N2") + " kn";
```

Slijedi izračun iznosa za poreziti. Dobije se oduzimanjem ostvarenih odbitaka od dohotka.

```
form4.iznos_poreziti = (form4.dohodak - form4.osobni_odbitak);
if (form4.iznos_poreziti < 0)
{
form4.iznos_poreziti = 0;
}
form4.label132.Text = form4.iznos_poreziti.ToString("N2") + " kn";
```

Kao u prethodnom slučaju upisuje se samo vrijednost. Način izračuna stoji upisan u tablici.

Slijedi izračun iznosa koji će se poreziti.

```
form4.iznos_poreziti = (form4.dohodak - form4.osobni_odbitak);
if (form4.iznos_poreziti < 0)
{
form4.iznos_poreziti = 0;
}
form4.label132.Text = form4.iznos_poreziti.ToString("N2") + " kn";
```

Od dohotka oduzima se odbitak. Ako je rezultat manji od nula, rezultat se postavlja na nula.

Izračun iznosa poreza:

```
double porez_po_24;
double porez_po_36;
if (form4.iznos_poreziti > 17500)
{
  porez_po_24 = 17500;
  porez_po_36 = form4.iznos_poreziti - 17500;
  form4.label136.Text = porez_po_24.ToString("N2") + " kn";
  form4.label144.Text = porez_po_36.ToString("N2") + " kn";
}
else
{
  porez_po_24 = form4.iznos_poreziti;
  porez_po_36 = 0;
  form4.label136.Text = porez_po_24.ToString("N2") + " kn";
  form4.label144.Text = porez_po_36.ToString("N2") + " kn";
}
```

Ako je iznos za poreziti veći od 17500 varijabli *porez_po_24* pridružuje se vrijednost 17500. A varijabli *porez_po_36* pridružuje iznos – 17500. U suprotnom varijabli *porez_po_24* pridružuje se cijeli iznos, a varijabli *porez_po_36* pridružuje se nula.

Slijedi sam izračun iznosa poreza.

```
double iznos_por_24;
iznos_por_24 = porez_po_24 * 0.24;
form4.label140.Text = iznos_por_24.ToString("N2") + " kn";

double iznos_por_36;
iznos_por_36 = porez_po_36 * 0.36;
form4.label148.Text = iznos_por_36.ToString("N2") + " kn";
```

Vrijednosti se upisuju u tablicu.

Slijedi izračun ukupne porezne obveze, koji je jednostavan zbroj ova dva poreza.

```
double ukupna_por_obveza;
ukupna_por_obveza = iznos_por_24 + iznos_por_36;
form4.label152.Text = ukupna_por_obveza.ToString("N2") + " kn";
```

Prirez porezu na dohodak:

```
double prirez_rezultat;
prirez_rezultat = ukupna_por_obveza * form4.prirez / 100;
form4.label163.Text = "r.br. 18 * " + form4.prirez + "%";
form4.label164.Text = prirez_rezultat.ToString("N2");
```

Prirez je postotak od iznosa poreza. Te je umnožak poreza i vrijednosti prireza. Ovdje se opet upisuje *način izračuna* jer prirez ovisi o mjestu stanovanja zaposlenika. Pa je prikazano koliki je točan postotak prireza.

Zbroj poreza i prireza:

```
double porez_i_prirez;
porez_i_prirez = ukupna_por_obveza + prirez_rezultat;
form4.label168.Text = porez_i_prirez.ToString("N2") + " kn";
```

Slijedi izračun neto plaće koja je dohodak umanjen za porez i prirez.

```
double neto_placa;
neto_placa = form4.dohodak - porez_i_prirez;
form4.label172.Text = neto_placa.ToString("N2") + " kn";
```

Slijedi izračun doprinosa za zdravstveno osiguranje. Iznosi 20% bruto plaće. Spada pod obveze poslodavca.

```
double zdravstveno_osiguranje;
zdravstveno_osiguranje = bruto_placa * 0.20;
form4.label177.Text = zdravstveno_osiguranje.ToString("N2") + " kn";
```

Izračun iznosa za zaštitu zdravlja na radu:

```
double zastita_zdravlja;
zastita_zdravlja = bruto_placa * 0.005;
form4.label181.Text = zastita_zdravlja.ToString(„N2“) + „ kn“;
```

Izračun doprinosa za zapošljavanje.

```
double zaposljavanje;
zaposljavanje = bruto_placa * 0.017;
form4.label185.Text = zaposljavanje.ToString("N2") + " kn";
```

Izračun potpunih izdataka poslodavca:

```
double izdatci;
izdatci = bruto_placa + zdravstveno_osiguranje + zastita_zdravlja + zaposljavanje;
form4.label189.Text = izdatci.ToString("N2") + " kn";
```

Na kraju se prikazuje forma *Izračun plaće.*

```
form4.Show();
```

3.3.11. Obrazac Izračun plaće

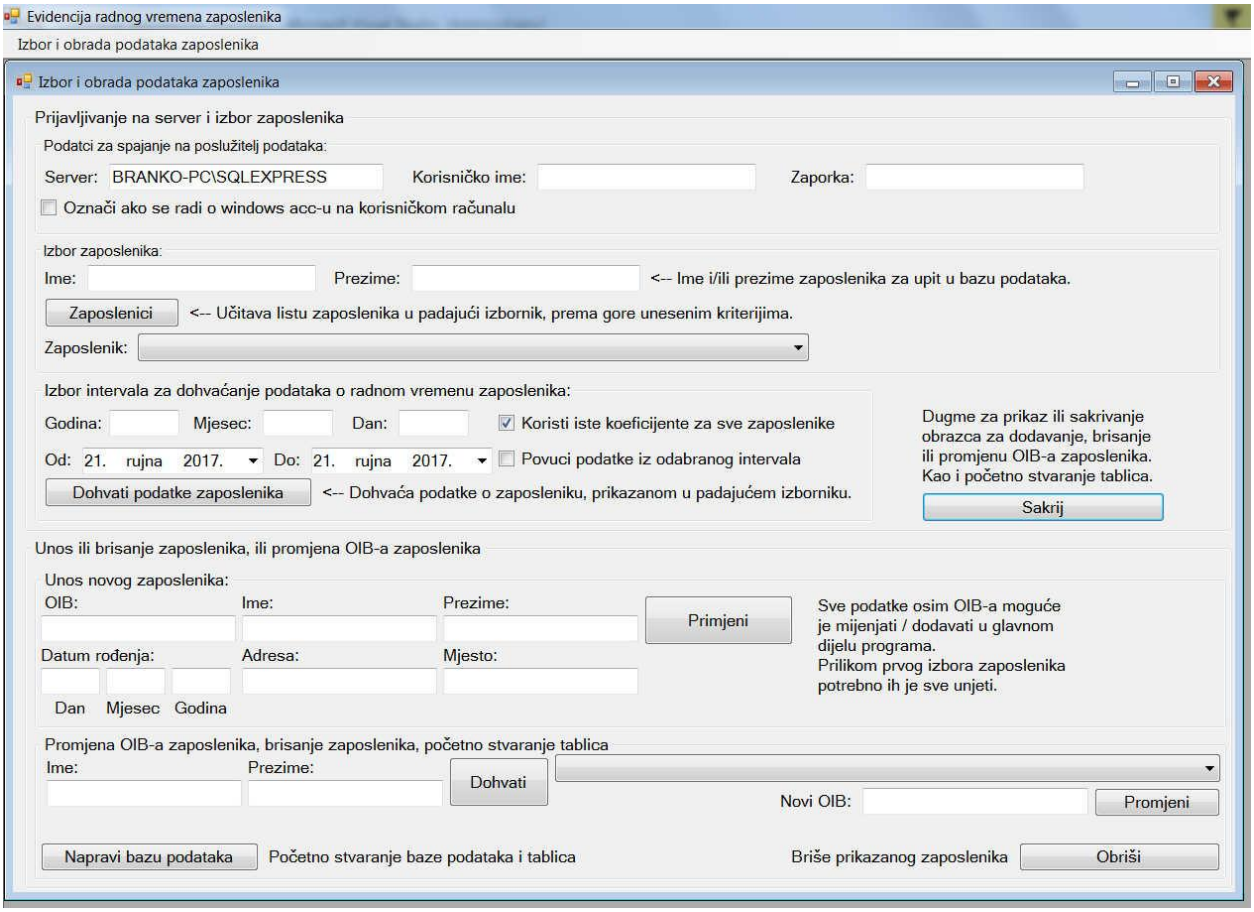
Ovaj obrazac sastoji se samo od grafičkog sučelja i varijabli. Ne događaju se nikakvi izračuni. Izračuni su napravljeni i proslijeđeni od ranijeg obrasca. Izgled:

```
public partial class Form4 : Form
{
public Form4()
{
InitializeComponent();
}
CultureInfo kultura = CultureInfo.CreateSpecificCulture("hr-HR");

public double bruto_placa;

// Porezni koeficijenti
public int por_osnovica;
public int osnovni_odbitak;
public double koef_odbitka;
public double uzdrzavan_obitelj;
public double prvo_dijete;
public double drugo_dijete;
public double trece_dijete;
public double cetvrto_dijete;
public double peto_dijete;
public double sesto_dijete;
public double sedmo_dijete;
public double osmo_dijete;
public double deveto_dijete;
public double slijedece_dijete;
public double invalidnost;
public double invalidnost_100;
public int prirez;

// Zaposlenik
public int uzdrzavani_clanovi;
public int br_djece;
public int invalidi;
public int invalidi_100;
public int mirovinski_stup;

// Rezultati
public double dop_za_mir1;
public double dop_za_mir2;
public double dohodak;
public double osnovni_os_odbitak;
public double odbitak_djece;
public double odbitak_uzdrzavani;
public double odbitak_invalidi;
public double odbitak_invalidi_100;
public double osobni_odbitak;
public double iznos_poreziti;

private void Form4_Load(object sender, EventArgs e)
{
}}
```

3.4. Način korištenja aplikacije

Prilikom pokretanja aplikacije pojavljuje se obrazac naziva *Izbor i obrada podataka zaposlenika*. Obrazac je podijeljen na dva glavna dijela. Gornji za unos podataka za spajanje na bazu podataka, izbor zaposlenika i izbor vremenskog intervala za koji će se podatci o zaposleniku dohvatiti.

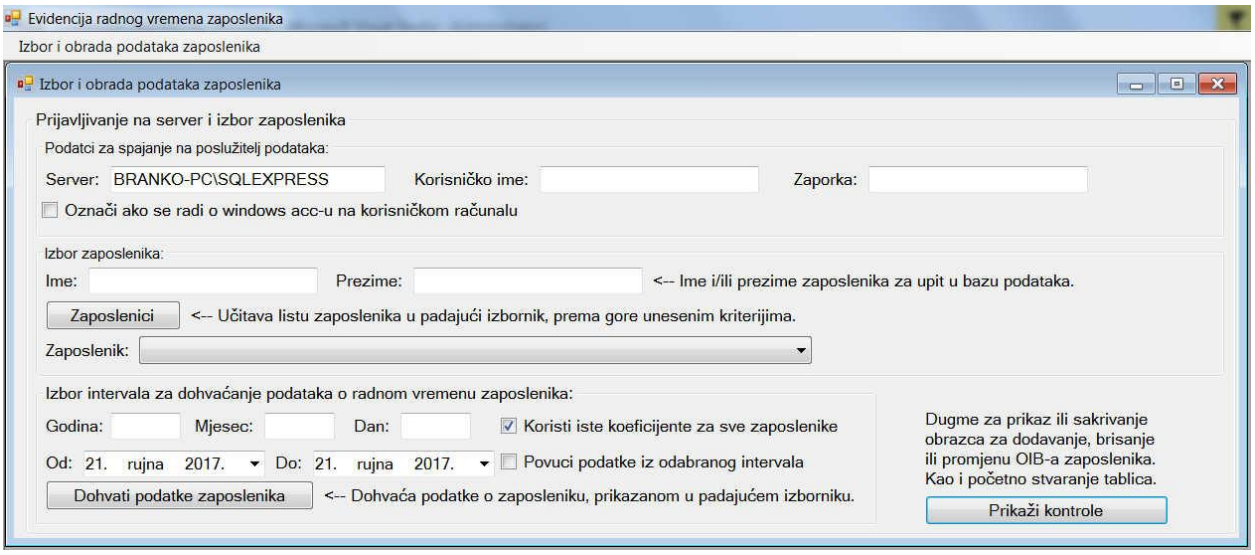
Donji dio za izradu baze podataka, dodavanje ili brisanje zaposlenika i promjenu OIB-a zaposlenika. Donji dio nije vidljiv prilikom pokretanja aplikacije. Slika grafičkog sučelja:



Sl. 3.5. *Obrazac Izbor i obrada podataka zaposlenika.*

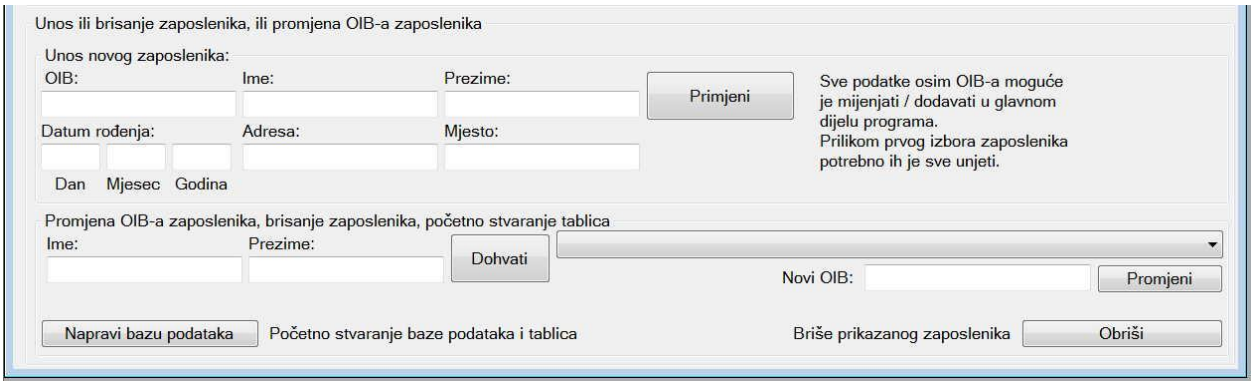
Nakon pokretanja aplikacije potrebnog je unijeti ime (adresu) servera (Baze podataka). Ako se koristi baza podataka na korisničkom računalu, koja koristi windows račun, nije potrebno unositi korisničko ime i zaporku, potrebno je označiti *checkBox* pored kojeg piše *Označi ako se radi o windows acc-u na korisničkom računalu.* U suprotnom potrebno je unijeti korisničko ime i zaporku.

Ako baza podataka ne postoji, potrebno je prvo pritisnuti dugme *Prikaži kontrole* sa slike 3.6. (ispod). Nakon toga će se prikazati i donji dio obrasca.



Sl. 3.6 *Gornji dio obrasca (kada je donji sakriven).*

Zatim pritisnuti tipku *Napravi bazu podataka*, slika 3.7.(ispod). Pojavit će se skočni prozor koji javlja uspjeh ili ne uspjeh.



Sl. 3.7. *Donji dio obrasca.*

Zatim se unesu podatci o zaposleniku u *groupBox* naziva *Unos ili brisanje zaposlenika, ili promjena OIB-a zaposlenika*, slika 3.7. Svi podatci moraju biti prisutni (*OIB, Ime, Prezime, Datum rođenja, Adresa i Mjesto*). Ostali potrebni podatci unose se kasnije. Pritiskom na dugme *Primjeni* unosi se zaposlenik u bazu podataka. Skočni prozor javlja uspjeh ili ne uspjeh.

Nudi se mogućnost brisanja zaposlenika, to se izvodi pritiskom na dugme *Dohvati*, slika 3.7. koje dohvaća listu zaposlenika u padajući izbornik. U slučaju velikog broja zaposlenika poželjno je upisati ime i/ili prezime. Izabrati zaposlenika, pritiskom na dugme *Briši* zaposlenik prikazan u padajućem izborniku biti će obrisan.

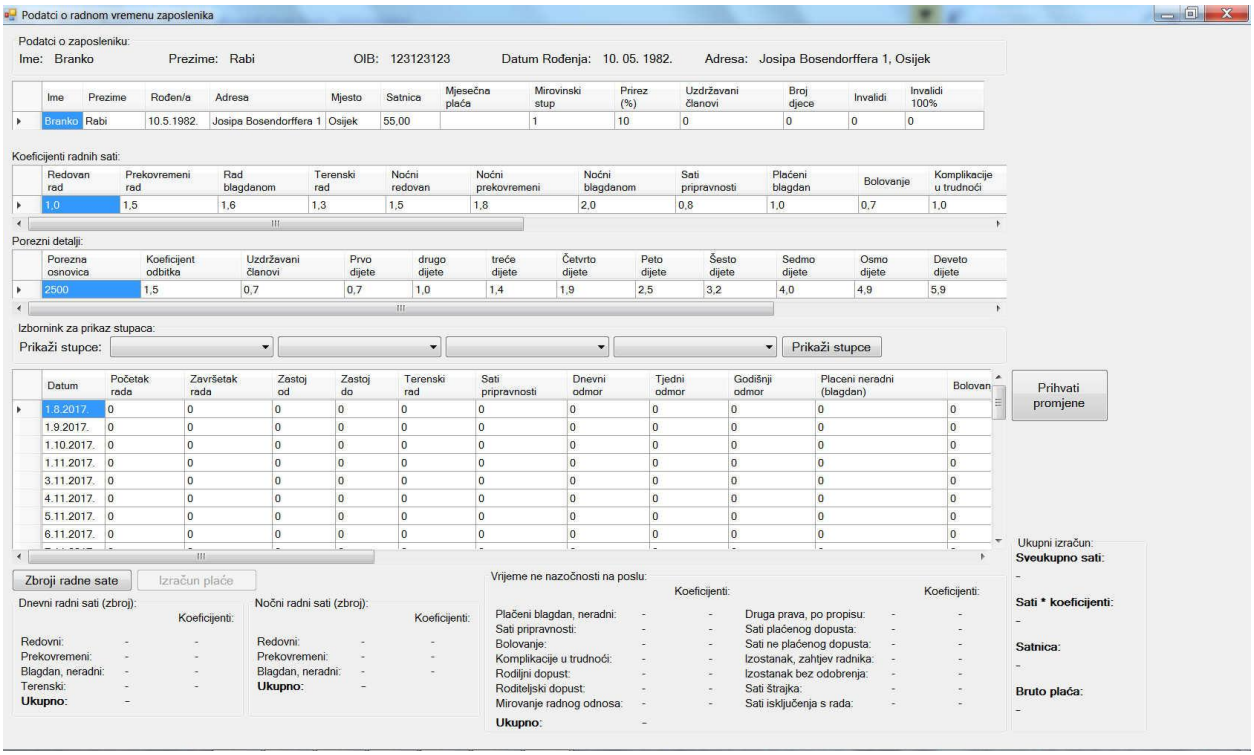
Ispod padajućeg menija je i dugme *Promjeni* te kontrola za unos teksta kraj koje piše *Novi OIB,* slika 3.7. Unosom OIB-a u ovu kontrolu i pritiskom na dugme *Promjeni* OIB prikazanog korisnika biti će promijenjen.

U gornjem dijelu vrši se izbor zaposlenika. Na način da se prvo iz baze podataka povuče lista zaposlenika u padajući izbornik pritiskom na dugme *Zaposlenici,* slika 3.6. Onda se iz padajućeg izbornika izabere zaposlenik. Obzirom da ovo može povući veliki broj zaposlenika, postoje polja *Ime* i *Prezime* unutar *groupBox-a* naziva *Izbor zaposlenika,* slika 3.6. Unosom imena i/ili prezimena, dohvatit će se zaposlenici samo toga imena i/ili prezimena. Ovo nikada ne bi trebalo vratiti veliki broj unosa.

Nakon toga iz padajućeg izbornika iza riječi označenog sa *Zaposlenik:* (Sl. 3.6.) izabere se zaposlenik.

Potrebno je još izabrati interval za koji se žele povući podatci o zaposleniku. To je moguće unošenjem godine i/ili mjeseca i/ili dana u polja *Godina, mjesec* i *dan* unutar *groupBox-a* naziva *Izbor intervala za dohvaćanje podataka o radnom vremenu zaposlenika:,* slika 3.6. ili korištenjem *timeDatePickera* i izborom početnog i završnog datuma, u koje slučaju treba u kojem slučaju treba označiti *checkBox* pored, kraj kojega piše *Povuci podatke iz odabranog intervala*, slika 3.6.

Pritiskom na dugme *Dohvati podatke zaposlenika* otvara se slijedeći obrazac naziva *Podatci o radnom vremenu zaposlenika,* slika 3.8. (ispod).



Sl. 3.8. *Obrazac podatci o radnom vremenu zaposlenika.*

U gornjem dijelu unutar *groupBox-a* naziva *Podatci o zaposleniku,* slika, 3.8.*:* prikazani su osnovi podatci o zaposleniku.

Ispod u *dataGridViev-u* prikazani su puni podatci o zaposleniku. Ako se radi o tek stvorenom zaposleniku treba popuniti podatke koji nedostaju. Podatci se predaju bazi podataka pritiskom na dugme *Prihvati promjene*, slika 3.8.

Potrebno je provjeriti valjanost svih podataka kako o zaposleniku, tako i vrijednosti koeficijenata radnih sati i koeficijenata poreza.

U donji *dataGridView, slika 3.8.* koji služi unosu podataka o radnom vremenu zaposlenika, moguće je unositi nove podatke, mijenjati stare i brisati podatke. Pritiskom na dugme *Prihvati promjene, slika 3.8.* šalju se promjene bazi podataka sa svih *dataGridView*-a. S obzirom na velik broj stupaca, zbog velikog broja vrsta radnih sati, moguće je izbirati do četiri stupca koja će biti vidljiva. Prva tri su uvijek vidljiva.

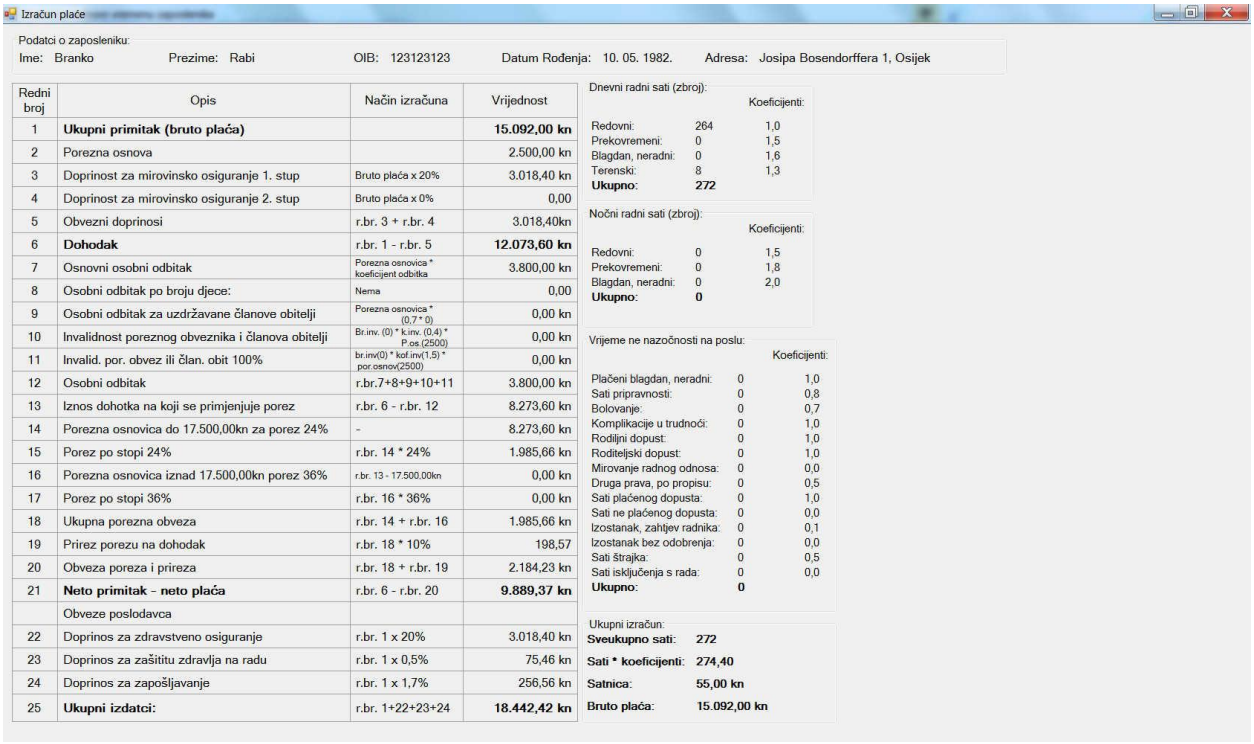
Ovo se postiže pomoću padajućih izbornika iznad samog *dataGridView-a* unutar *groupBox-a* naziva *Izbornik za prikaz stupaca,* slika 3.8. te pritiskom na dugme *Prikaži stupce.*

Pritiskom na dugme *Zbroji radne sate*, slika 3.8. zbrojit će se svi radni sati pojedine vrste za interval podataka prikazan u *dataGridView-u.* Ujedno zbrojit će se dnevni radni sati, Noćni radni

sati, i sati ne prisutnosti na poslu. Pomnožit će se sa pripadajućim koeficijentima. Prikazati će se bruto plaća. Prikaz je na dnu obrasca, slika 3.8. i omogućit će se dugme izračun plaće.

Pritiskom na dugme izračun plaće pojavit će se slijedeći obrazac. Podatci o radnom vremenu bit će prepisani na slijedeći obrazac.

Na slijedećem obrascu prikazan je izračun bruto plaće, doprinosa, poreza, prireza, odbitaka i neto plaće.



Sl. 3.9. *Obrazac izračun plaće*

Ovaj obrazac pruža pregled svih podataka koji se tiču plaće zaposlenika.

4. ZAKLJUČAK

Aplikacija uspješno izrađuje bazu podataka, potrebnog izgleda, u koju se zapisuju podatci o radnom vremenu zaposlenika, kao i ostali popratni podatci potrebni za izračune bruto plaće, dohotka, odbitaka, poreza, prireza porezu i neto plaće od strane aplikacije. Ostali podatci su koeficijenti vrijednosti pojedine vrste radnih sati. Broj uzdržavane djece i broj uzdržavanih članova uže obitelji (supružnik, roditelji), eventualna invalidnost, zaposlenika i/ili uzdržavanog člana uže obitelji. Koeficijenti odbitka za uzdržavanu djecu i članove uže obitelji te eventualnu invalidnost istih. Komunikacija sa bazom podataka se uspješno odvija. Sastavljanje upita bazi podataka na osnovu informacija unesenih u korisničko sučelje i povlačenje željenog dijela podataka radi dobro, kao i povlačenje svih podataka za jednog zaposlenika. Vraćanje promijenjenih podataka bazi podataka uglavnom radi dobro, ali ima jedan problem. Naime, promjena podatka koji je primarni ključ uz pomoć *dataGridView-a* ne funkcionira, program izbacuje *Concurrency violation.* Naravno promjena je moguća tako što se unos obriše i upiše drugi te program i dalje izvršava svoju zadaću uspješno. Ovaj problem izražen je samo u tablici radnih sati zaposlenika. Primarni ključevi ostalih tablica nisu dani korisniku na izmjenu preko *dataGridView*-a jer su veza sa zaposlenikom te se ne mijenjaju preko *dataGridView-a*, nego se mijenjaju sa promjenom OIB-a zaposlenika te samim time nemaju ovaj problem. Promjene svih ostalih podataka (koji nisu primarni ključ) nemaju ovaj problem. Izračune na podatcima aplikacija uspješno obavlja. Popratne funkcionalnosti također rade. Popratne funkcije su: unos novog zaposlenika, brisanje zaposlenika, mijenjanje OIB-a zaposlenika. Na žalost aplikacija ne podržava ispis na papir, ovo područje je jako slabo pokriveno literaturom.

LITERATURA

[1] Julijan Šribar, Boris Motik, **Demistificirani C++,** Element Zagreb, 2001

[2] Dan Clark, **Beginning C# Object-Oriented Programming,** Apress, 2011

SAŽETAK

Aplikacija za pomoć u praćenju rada privatne tvrtke.

Aplikacija omogućuje unos i spremanje podataka o radnom vremenu zaposlenika u bazu podataka. Dohvaćanje podataka od strane korisnika iz baze podataka u cijelosti ili željenom dijelu te izmjenu, brisanje i dodavanje podataka.

Uz pomoć podataka o radnom vremenu zaposlenika (broj i vrsta radnih sati, satnica) aplikacija izračunava bruto plaću. Nakon toga uz pomoć podataka o zaposleniku (broj uzdržavanih članova uže obitelji, visina prireza porezu) aplikacija izračunava doprinose, dohodak, porez, prirez i neto plaću.

Ključne riječi: radno vrijeme, plaća, bruto, neto, dohodak, doprinosi, porez, prirez

ABSTRACT

Application for support in monitoring of management of private company.

The application allows user to enter and save daily work hour's data of employee to the database. Retrieving data by the user from database partially or entirely as desired and to modify, delete and add data.

With the help of employee work hour's data (number and type of work hours, hourly rate) the application calculates the gross salary. Afterwards, with the help of employee data (the number of close family members sustained, the surtax rate) application calculates contributions, income, tax, surtax and net salary.

Key words: work hour's, salary, gross, net, contributions, income, tax, surtax

ŽIVOTOPIS

Branko Rabi, rođen 10. 05. 1982. u Osijeku. Pohađao osnovnu školu Ljudevita Gaja u Osijeku. Završio srednju Strojarsku školu u Osijeku smjer Strojarski tehničar. Trenutno zaposlenje laborant u laboratoriju za ispitivanje građevinskih materijala.

Branko Rabi

ŽIVOTOPIS

Branko Rabi, rođen 10. 05. 1982. u Osijeku. Pohađao osnovnu školu Ljudevita Gaja u Osijeku. Završio srednju Strojarsku školu u Osijeku smjer Strojarski tehničar. Trenutno zaposlenje laborant u laboratoriju za ispitivanje građevinskih materijala.

B. Rabi

Branko Rabi