Windows Form aplikacija za obračun plaće

Milošević, Antonio

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Windows Form aplikacija za obračun plaće

Završni rad

Antonio Milošević

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 17.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Antonio Milošević
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3962, 24.09.2019.
OIB studenta:	67833186579
Mentor:	Izv. prof. dr. sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Windows Form aplikacija za obračun plaće
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	17.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Antonio Milošević
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3962, 24.09.2019.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Windows Form aplikacija za obračun plaće**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja Antonio Milošević, OIB: 67833186579, student/ica Fakulteta elektrotehnike,
računarstva i informacijskih tehnologija Osijek na studiju Preddiplomski sveučilišni studij Računarstvo,
kao autor/ica ocjenskog rada pod naslovom: Windows Form aplikacija za obračun plaće**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

Osijek, 23.09.2020. _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

Sadržaj

1. Uvod

Znamo kako su prije izgledali obračuni plaća, ručno su se zbrajali odrađeni sati i općenito se cjelokupan obračun radio na papiru. Kako je vrijeme prolazilo i tehnologije su napredovale, došlo je do pojave aplikacija za računala koje same računaju iznos koji poslodavci trebaju isplatiti na kraju mjeseca za svakog radnika.

Koristeći aplikaciju za računanje obračuna plaće se znatno štedi vrijeme koje se inače gubilo na računanje svih potrebnih izdataka. Za izradu gore navedenog programa je potrebno odabrati razvojno okruženje i programski jezik u kojemu će se pisati. Odabrano razvojno okruženje je Microsoft Visual Studio jer ima ugrađen dodatak za programiranje Windows Form aplikacija u C# programskom jeziku.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada jest razvoj Windows Form aplikacije koja će omogućiti unos tvrtki i djelatnika u njoj. Potrebno je unijeti sve podatke o djelatnicima koji su nužni za obračun plaće te drugih isplata djelatnicima.

Biti će kreirana SQL baza podataka u kojoj će se nalaziti podaci o radnicima kojoj će se moći pristupati preko aplikacije. Moći će se unositi ili brisati podaci iz nje. Nakon što su svi potrebni podaci upisani i obračun je gotov, biti će omogućen ispis platne liste. Sâm program će biti pisan u Microsoft Visual Studio razvojnom okruženju uz pomoć C# programskog jezika.

2. Integrirano razvojno okruženje

U današnje vrijeme gotovo svaki programski jezik ima svoje integrirano razvojno okruženje koje omogućuje lakše pisanje kôda i općenito razvoj softverskog rješenja. Integrirano razvojno okruženje je softverska aplikacija koja pruža sveobuhvatne mogućnosti računalnim programerima za razvoj softverskih rješenja.

Razvojno okruženje se obično sastoji barem od uređivača izvornog kôda, alata za automatizaciju gradnje i programa za ispravljanje grešaka. Neka razvojna okruženja, kao što su „NetBeans“ i „Eclipse“, uz kompajler sadrže i tumač programskog jezika.

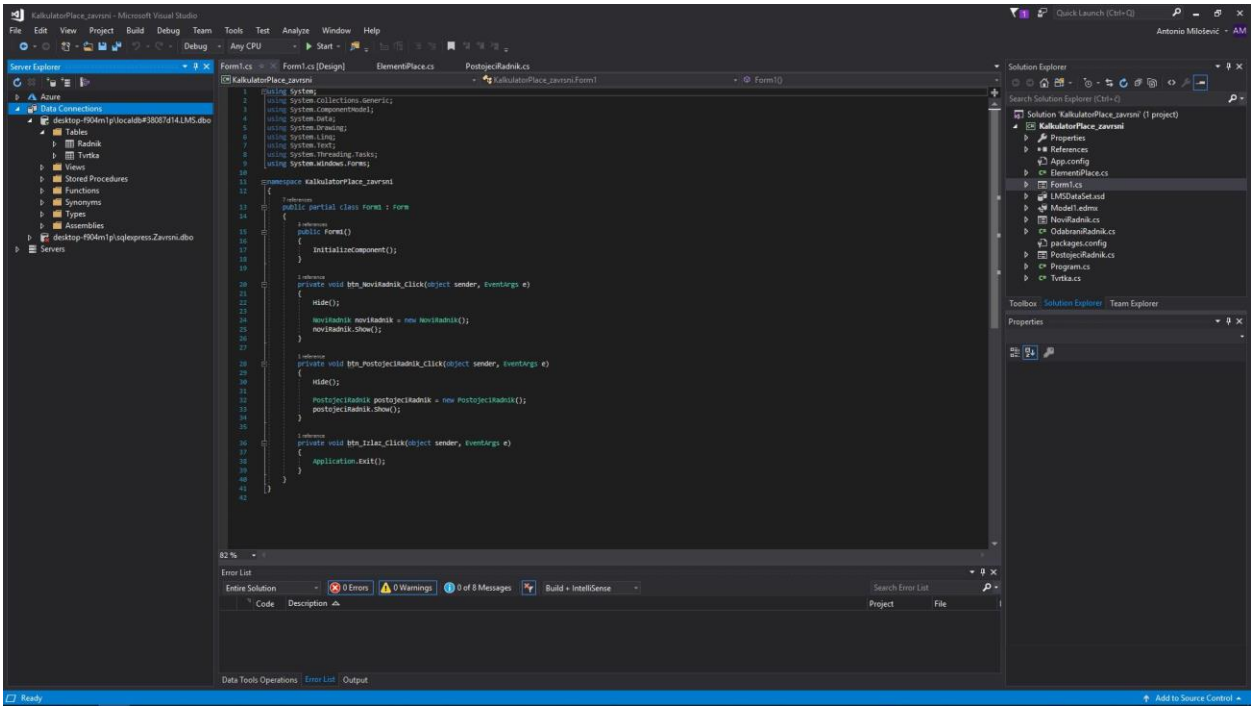
Mnoga moderna razvojna okruženja također imaju preglednik klasa, objekata i hijerarhijski dijagram klasa za upotrebu u objektno orijentiranom razvoju programskih rješenja. Integrirana razvojna okruženja su osmišljena tako da maksimiziraju produktivnost programera pružajući čvrsto povezane komponente sa sličnim korisničkim sučeljima. Razvojna okruženja predstavljaju jedinstveni program u kojem se vrši sav razvoj aplikacije. Obično nude mnoge značajke za izradu, izmjenu, sastavljanje, postavljanje i uklanjanje pogrešaka.

Rani sustavi nosu mogli podržati integrirana razvojna okruženja jer su se programi pripremali koristeći dijagrame toka i unoseći programe s bušenim karticama. „Dartmouth BASIC“ je prvi programski jezik koji je stvoren s integriranim razvojnim okruženjem. Njegovo razvojno okruženje se temeljilo na naredbama i stoga nisu previše izgledali poput današnjih razvojnih okruženja koja su se pojavila nakon uvođenja grafičkog korisničkog sučelja. Međutim, integrirao je uređivanje, upravljanje datotekama, prevođenje kôda, ispravljanje grešaka i izvršavanje na način koji je u skladu s modernim razvojnim okruženjima.

„Maestro I“ je proizvod tvrtke „Softlab Munich“ i bio je prvo integrirano razvojno okruženje na svijetu. „Maestro I“ je instaliran za 22000 programera širom svijeta. Do 1989. godine u Saveznoj Republici Njemačkoj je postojalo 6000 instalacija. Danas se posljednja verzija može pronaći u muzeju informacijskih tehnologija u Arlingtonu.

2.1. Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okruženje tvrtke Microsoft. Prethodno navedeno razvojno okruženje se koristi za razvoj računalnih programa, web usluga, web stranica, web aplikacija i mobilnih aplikacija. Visual Studio koristi platforme koje je razvila tvrtka Microsoft, a to su Microsoft Silverlight, Windows Presentation Foundation, Windows Store, Windows API i Windows Form aplikacija.



Slika 1. Sučelje Microsoft Visual Studio razvojnog okruženja

Uređivač kôda, kojeg uključuje Visual Studio, podržava „IntelliSense“ komponentu za dovršetak i refaktoriranje kôda. Ostatak ugrađenih alata koje uključuje Visual Studio su dizajner za izgradnju grafičkog korisničkog sučelja, profiler kôda, dizajn web usluga, dizajner baze podataka i dizajner klasa. Visual Studio prihvaća dodatke na gotovo svakoj razini, odnosno mogu se uključiti dodaci za širok spektar platformi.

Visual Studio razvojno okruženje podržava čak 36 programskih jezika. Najosnovnija verzija Visual Studio razvojnog okruženja je „Community“ izdanje koje je besplatno. Slogan Visual Studio Community izdanje je: „Besplatan, potpuno opremljeno razvojno okruženje za studente i pojedinačne programere“. Izdanje koje je korišteno za razvoj programa za obračun plaće je Microsoft Visual Studio Enterprise kojeg studenti FERIT-a mogu dobiti besplatno.

2.2. .NET Framework

.NET Framework je softverska apstrakcija koju je razvio Microsoft i koji prvenstveno radi na Microsoft Windows sustavu. U računalnom programiranju, softverski okvir („Framework“) je apstrakcija u kojoj se softver koji pruža generičku funkcionalnost može selektivno mijenjati dodatnim kôdom.

Uključuje veliku biblioteku koja se naziva „Framework Class Library“ (FCL) i pruža mogućnost da se može koristiti na više programskih jezika, odnosno svaki programski jezik može koristiti kôd napisan na nekom drugom jeziku. „Common Language Runtime“ (CLR) je softversko okruženje u kojemu se izvršavaju programi napisani za .NET Framework.

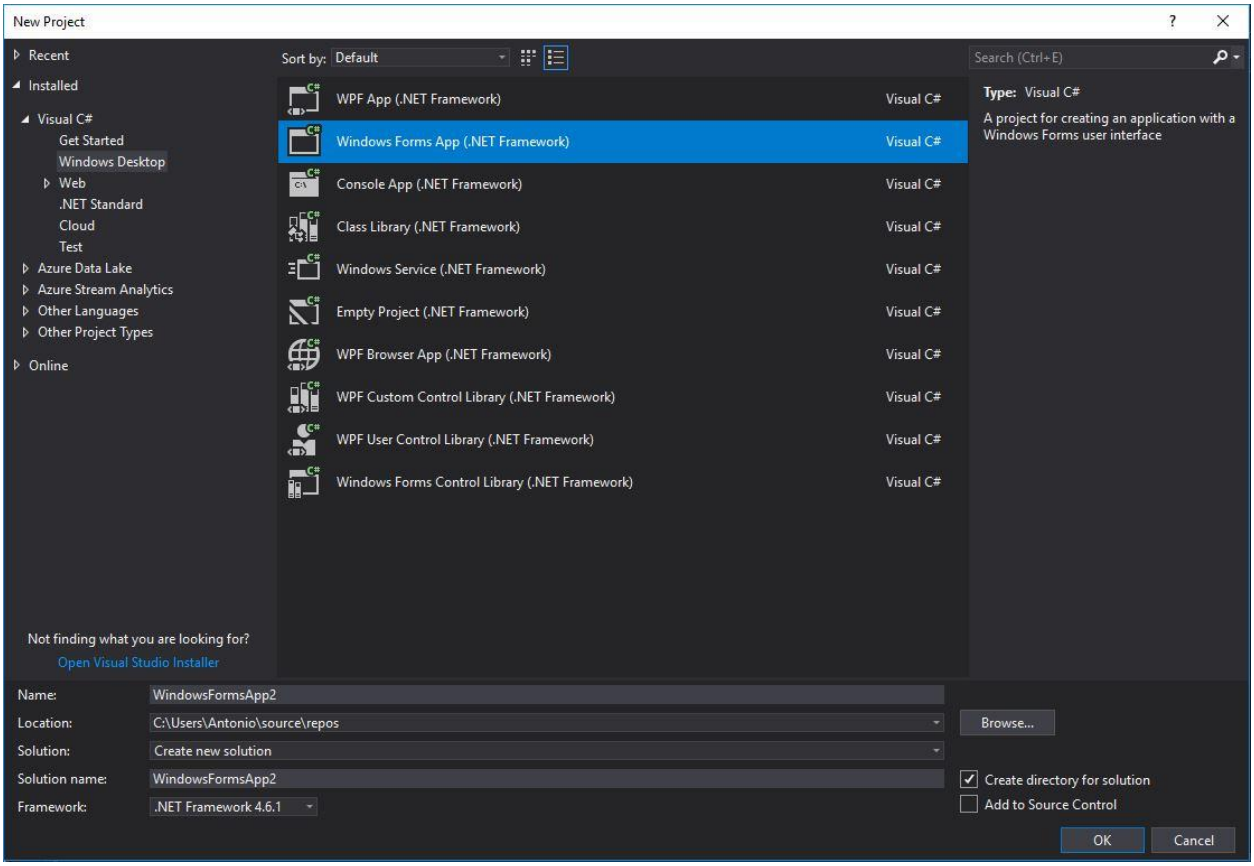
„Common Language Runtime“ (CLR) je zapravo virtualni stroj koji korisnicima pruža usluge rukovanja iznimkama, sigurnosti i upravljanja memorijom. Kada spojimo „Framework Class Library“ (FCL) i „Common Language Runtime“ (CLR) dobijemo .NET Framework.

„Framework Class Library“ (FCL) pruža mrežne komunikacije, pristup podacima, povezivanje s bazom podataka, korisničko sučelje, kriptografiju, razvoj web aplikacija i numeričke algoritme. Programeri razvijaju svoje aplikacije kombinirajući izvorni kôd s .NET Frameworkom i drugim bibliotekama.

Krajem 1990. je Microsoft počeo razvijati .NET Framework, izvorno pod nazivom „Next Generation Windows Services“ (NGWS) koji je bio dio .NET ideje. Oko 2000. godine su počele izlaziti prve beta verzije .NET 1.0. Microsoft i Intel su u kolovozu 2000. godine radili na standardizaciji „Common Language Infrastructure“ (CLI) i C#. „Common Language Infrastructure“ je otvorena specifikacija koja omogućuje da se koristi više jezika visoke razine na različitim računalnim platformama.

2.2.1. Windows Forme

Windows Forme su besplatna biblioteka s grafičkim korisničkim sučeljem kao dio Microsoft .NET Frameworka. Pruža mogućnost pisanja bogatih aplikacija za tablete, prijenosna i stolna računala. Iako se na Windows Forme gleda kao na zamjenu za raniju i nešto kompleksniju „Microsoft Foundation Class“ biblioteku koja je temeljena na C++ programskom jeziku, one ne nude usporedivu paradigmu i djeluju kao platforma za grafičko korisničko sučelje u višeslojnom rješenju.



Slika 2. Odabir novog Windows Forms App projekta

Windows Form aplikacija je aplikacija vođena događajima koju podržava Microsoft .NET Framework. Za razliku od nekih drugih programa, Windows Form aplikacija većinom čeka da korisnik nešto napravi, da popuni tekstualni okvir ili da pritiskom gumba aktivira određenu radnju nad podacima.

2.3. Programski jezik C#

Elegantan i „type-safe“ objektno orijentirani programski jezik koji omogućuje razvoj različitih robusnih i sigurnih aplikacija koje se pokreću u .NET ekosustavu. Razvijen od strane Microsofta, može se koristiti za izgradnju igara, mobilnih aplikacija, web aplikacija, desktop aplikacija i još puno toga.

Sintaksa C# programskog jezika je izražajna, ali je također laka za učenje i jednostavna. Korištenje vitičastih zagrada u C#-u će biti prepoznatljivo svima koji poznaju programske jezike C, C++ i Java. Ako programer zna jedan od tih programskih jezika, može u kratkom vremenu

početi produktivno raditi u C#-u. Sintaksa pojednostavljuje složenosti C++-a i omogućuje mnoštvo korisnih značajki.

„C# je objektno orijentirani programski jezik koji je usmjeren na komponente“[1]

Kada kažemo da je programski jezik u potpunosti objektno orijentiran, misli se na to da je svaki podatak predstavljen kao objekt određene klase. Postoje četiri glavna načela koja neki jezik čine objektno orijentiranim. To su enkapsulacija, apstrakcija, polimorfizam i nasljeđivanje.

- Enkapsulacija je mehanizam skrivanja implementacije podataka ograničavanjem pristupa javnim metodama.
- Apstrakcija znači koncept ili ideju koja nije povezana s nekom određenom instancom. Korištenjem apstraktne klase ili sučelja izražavamo namjeru klase, a ne stvarnu implementaciju.
- Polimorfizam u suštini znači višeobličje. Osobina kod koje jedna metoda može imati različiti broj i vrstu parametara.
- Nasljeđivanje prikazuje odnos „je“ i „ima“ između klasa. Može se opisati kao kada jedna klasa nasljeđuje određene osobine nadklase.

2.4. ADO.NET

ADO.NET je tehnologija koja omogućuje korisnicima pristup podacima iz Microsoft .NET Framework-a i omogućuje komunikaciju između relacijskih i nerelacijskih sustava. To je određena grupa komponenti koje programeri mogu iskoristiti za pristup podacima i podatkovnim uslugama iz baze podataka. Dio je osnovne biblioteke koja je uobičajeno uključena u Microsoft .NET Framework.

„ADO.NET pruža najizravniji način pristupa podacima unutar .NET Framework-a“[2]

Najčešće se koristi za izmjenu, dodavanje i brisanje podataka u relacijskim sustavima baza podataka. ADO.NET se smatra nasljednikom „ActiveX Dana Objects“ (ADO) tehnologije, ali je kompletno izmijenjen pa ga se može promatrati kao potpuno novim proizvodom.

[1] [1] A tour od the C# language
[2] [2] ADO.NET Overview

3. Baza podataka

Baza podataka je organizirana kolekcija podataka koji se pohranjuju i se može pristupiti preko računalnog sustava. Kada su baze podataka kompleksnije, često se izrađuju uz pomoć tehnika modeliranja i formalnog dizajna.

„Računalna baza podataka jest spremnik objekata. Jedna baza podataka može sadržavati više od jedne tablice.“[3]

Postoje tzv. SQL (relacijske) i NoSQL (nerelacijske) baze podataka. Relacijske baze podataka upravljaju i sadrže relacijski strukturirane podatke i imaju mogućnost manipuliranja tim podacima. Podaci se spremaju u tablice koje se sastoje od redova i stupaca. Nerelacijske baze nemaju točnu definiciju. To je baza koja je dizajnirana za distribuciju, pristup i pohranu podataka što čini pomoću metoda, za razliku od relacijskih baza.

3.1 SQL

„Structured Query Language“ je programski jezik visoke razine. Trenutno najpopularniji strukturni upitni jezik za ažuriranje, izradu, brisanje i traženje podataka iz relacijskih baza podataka. Jezik je standardiziran preko ISO i ANSI standarda.

SQL je prvotno bio razvijen u IBM-u, a sudjelovali su Donald D. Chamberlin i Raymond F. Boyce nakon što su početkom 1970. saznali za relacijski model Edgara F. Codda. Prva verzija, koja se u početku zvala „SEQUEL“ što znači „Structured English Query Language“, je dizajnirana za manipulaciju i pronalaženje podataka koji su bili pohranjeni u IBM-ovom relacijskom sustavu.

Vidjevši potencijal u opisu koncepta Chamberlina i Boycea, Relational Software Inc. (današnji Oracle Corporation) je razvio vlastitu inačicu RDBMS za CIA, mornaricu i ostale. Relational Software Inc. je u ljeto 1979. godine predstavio Oracle V2 za VAX računala kao prva komercijalno dostupna implementacija SQL-a.

[3] [3] Osnove baza podataka

3.1.1. Naredbe

1.) <u>Pretraga i grupiranje podataka</u>

- SELECT
  - FROM, JOIN
  - WHERE
  - GROUP BY
  - ORDER BY
  - HAVING

PRIMJER:

```
SELECT * FROM Videoteka
WHERE Cijena < 120
ORDER BY NazivFilma;
```

Slika 3. Primjer pretrage i grupe podataka

2.) <u>Manipulacija podacima</u>

- MERGE
- DELETE
- UPDATE
- INSERT

PRIMJER:

```
INSERT INTO MojaTablica (Stupac1, Stupac2, Stupac3)
VALUES(
    'provjera',
    'Ž',
    '12'
);
```

Slika 4. Primjer manipulacije podataka

3.) Kontrola transakcije

- ROLLBACK
- COMMIT
- BEGIN WORK

PRIMJER:

```
BEGIN WORK;
UPDATE Inventura
SET Kolicina = kolicina - 3
WHERE Proizvod = Kapa;
COMMIT;
```

Slika 5. Primjer kontrole transakcije

4.) Definicija podataka

- DROP
- TRUNCATE
- CREATE
- ALTER

PRIMJER:

```
CREATE TABLE MojaTablica
(
    Stupac1 INT NOT NULL,
    Stupac2 VARCHAR(30),
    Stupac3 DATE,
    CONSTRAINT pk_Stupac1 PRIMARY KEY(Stupac1)
);
```

Slika 6. Primjer definicije podataka

5.) <u>Kontrola prava</u>

- REVOKE
- GRANT

PRIMJER:

```
GRANT SELECT, UPDATE
ON MojaTablica TO Korisnik1, Korisnik2;
```

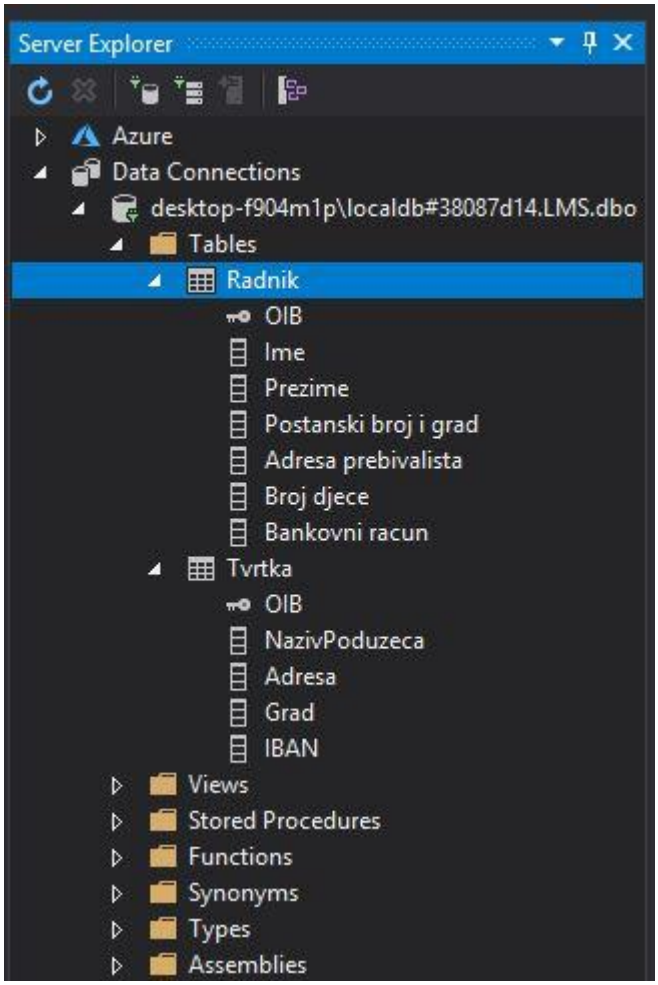
Slika 7. Primjer kontrole prava

4. Razvoj i opis korištenja aplikacije

U nastavku je objašnjeno kako izgleda završna verzija aplikacije, koji gumb što radi i izgled tablica koje su pohranjene u lokalnoj bazi podataka.

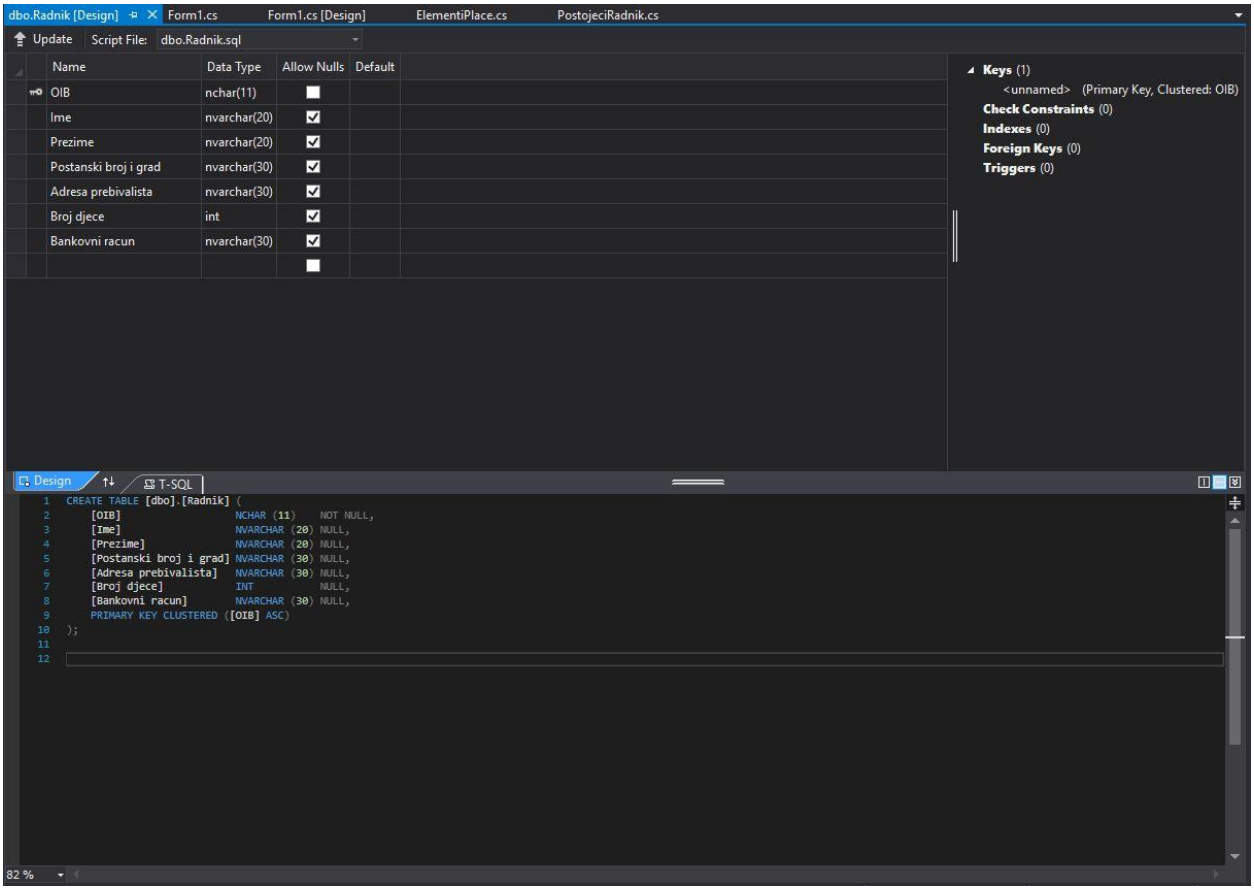
4.1. Izgled i logičko oblikovanje baze podataka

Korištena je lokalna baza podataka u kojoj imamo dvije tablice, „Radnik“ i „Tvrtka“. U nastavku možemo vidjeti kako izgleda alatna traka za pretraživanje mogućih podatkovnih veza.



Slika 8. Izgled trake za pregled podatkovnih veza

Tablica „Radnik“ sadrži atribute OIB, Ime, Prezime, Poštanski broj i grad, Adresa prebivališta, Broj djece i Bankovni račun. U nastavku možemo vidjeti samu tablicu i pripadajući kôd.



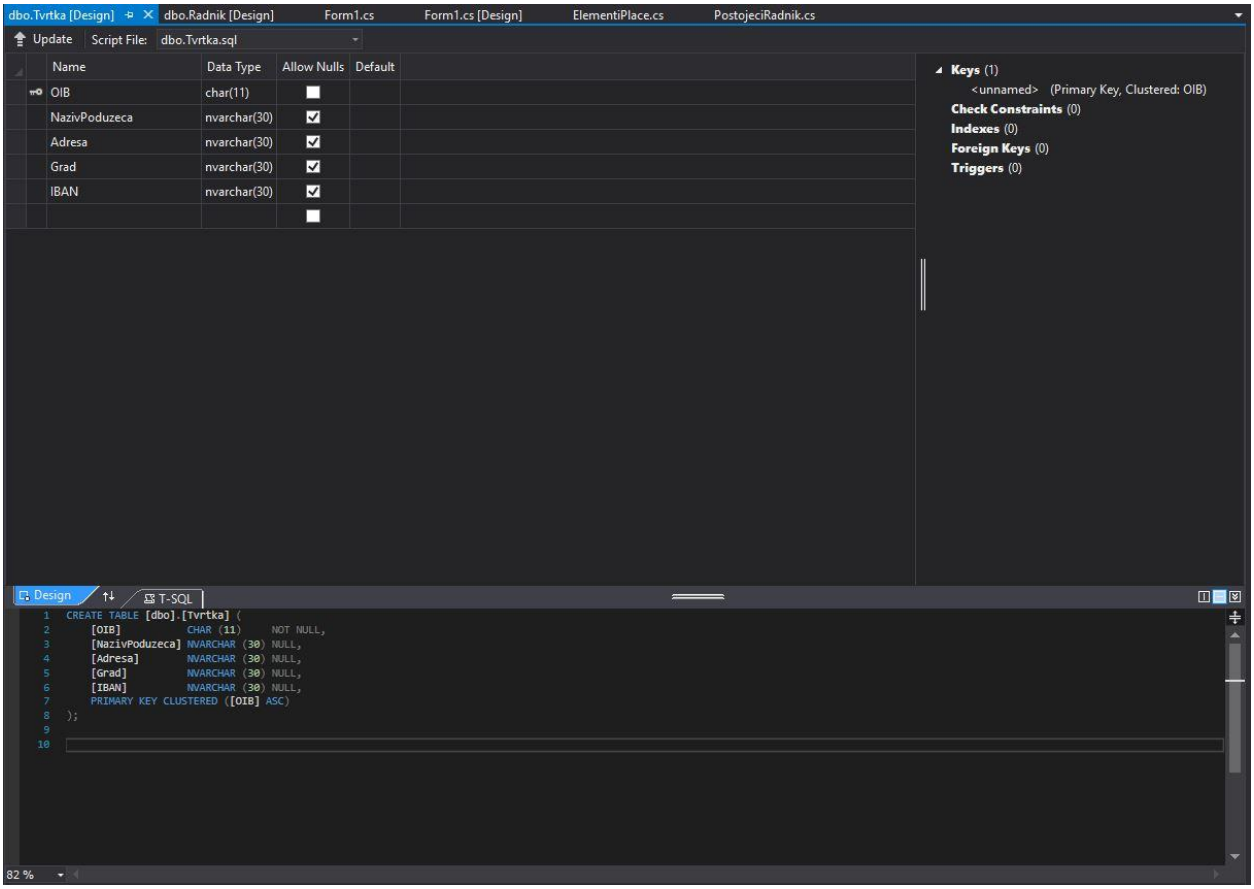
Slika 9. Logičko oblikovanje tablice „Radnik“

OIB	Ime	Prezime	Postanski broj i grad	Adresa prebivalista	Broj djece	Bankovni racun
88719287261	Želimir	Pervan	31000 Osijek	Županijska ulica 20	4	HR9987389287816374867 2
90876765142	Ivan	Požega	32000 Slavonski Brod	Osječka ulica 20	3	HR8987617263516278726 1
98076787654	Ivo	Patjera	31550 Valpovo	Ivanova Ulica 2	1	HR1234567890890756431
98726718912	Ivan	Ivić	31550 Valpovo	Antuna Branka Šimića 1	1	HR7876271891092647651 7
98761526761	Marko	Marić	10000 Zagreb	Ivana Gorana Kovačića 21	1	HR7836728167287614091 8
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Slika 10. Podaci unutar tablice „Radnik“

Podaci unutar tablice su eksperimentalni i upisani su samo zbog testiranja napisanog kôda.

Druga tablica, „Tvrtka“ ima atribute OIB, NazivPoduzeca, Adresa, Grad i IBAN. U nastavku možemo vidjeti kôd i samu tablicu.



Slika 11. Logičko oblikovanje tablice „Tvrtka“

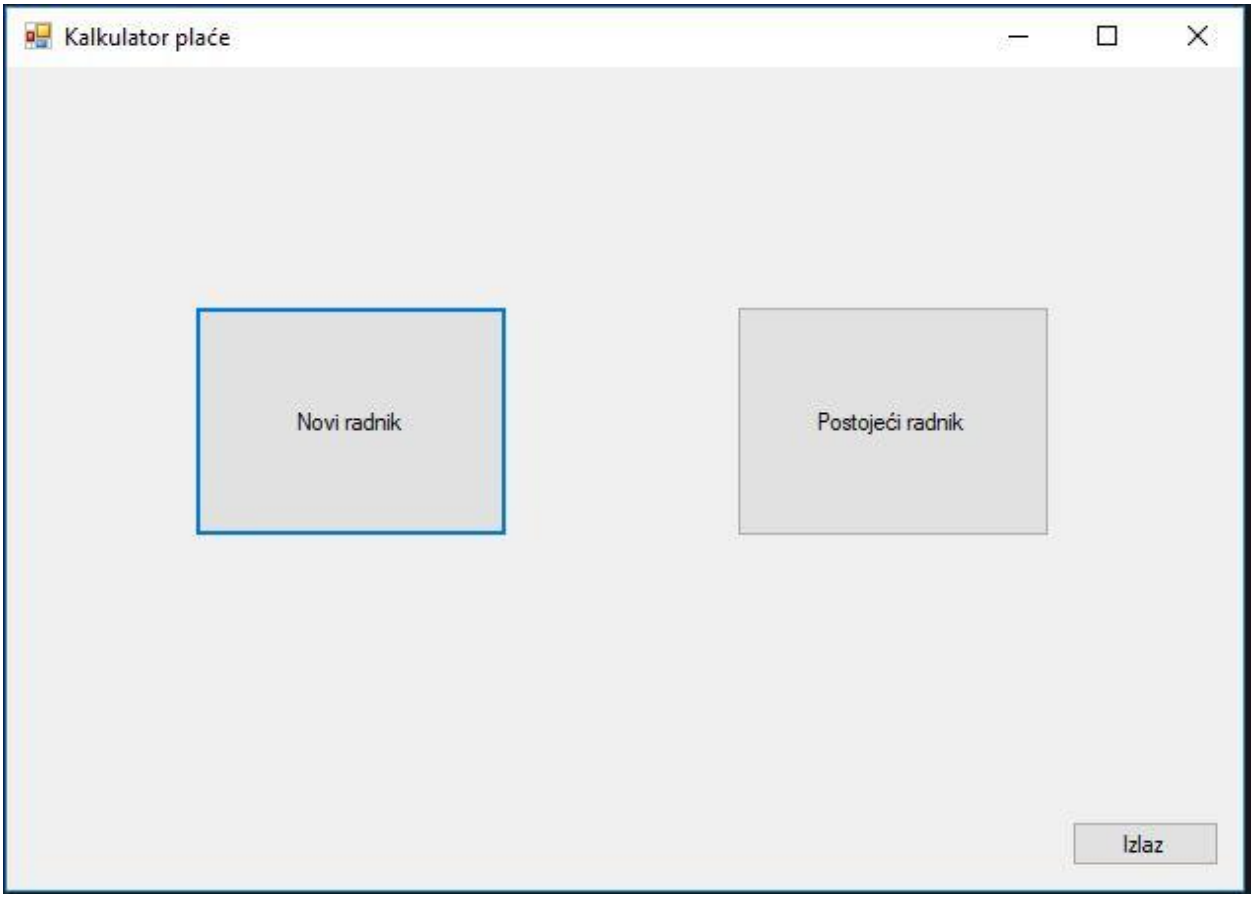
OIB	NazivPoduzeca	Adresa	Grad	IBAN
78910298171	Flestican d.o.o.	Zagrebačka 11	31000 Osijek	HR8971829873829182716 7
NULL	*NULL*	*NULL*	*NULL*	*NULL*

Slika 12. Podaci unutar tablice „Tvrtka“

Podaci unutar tablice „Tvrtka“ su također eksperimentalni i koriste se samo zbog testiranja aplikacije.

4.2. Funkcionalnost aplikacije i pripadajući kôd

Izgled aplikacije je zamišljen tako da se na početku može odabrati hoćemo li unositi novog radnika ili ćemo računati obračun plaće postojećem radniku. U nastavku je slika prozora kada se otvori aplikacija.



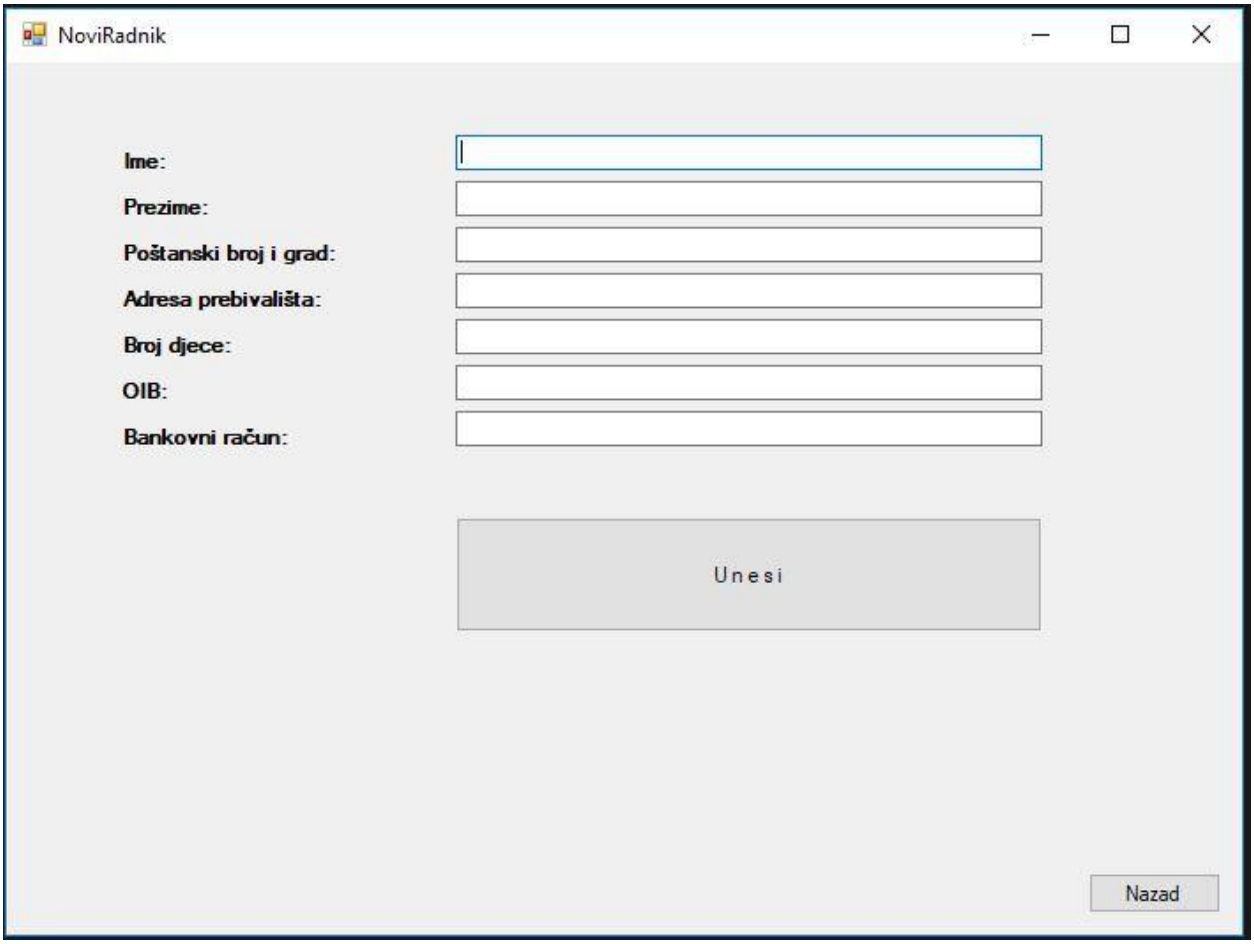
Slika 13. Izgled početnog prozora

Možemo vidjeti kako je dizajn početnog prozora prilično jednostavan i praktičan, uz to dodan je i gumb za izlaz iz aplikacije. U nastavku slijedi slika prozora kada se pritisne gumb „Novi radnik“ i pripadajući kôd koji izvršava radnju otvaranja novog prozora pritiskom na gumb.

```
1 reference
private void btn_NoviRadnik_Click(object sender, EventArgs e)
{
    Hide();

    NoviRadnik noviRadnik = new NoviRadnik();
    noviRadnik.Show();
}
```

Slika 14. Kôd za otvaranje prozora „NoviRadnik“

Nakon pritiska na gumb prvo se sakriva početni prozor, a to se izvršava uz pomoć „Hide()“ metode. Instanciranje klase NoviRadnik je izvršeno odmah ispod i prikazivanje prozora „NoviRadnik“ se omogućuje uz pomoć „Show()“ metode.



Slika 15. Izgled prozora „NoviRadnik“

Pritiskom na gumb „Novi radnik“ otvara nam se novi prozor u kojemu se vide tekstualni okviri u koje trebamo unijeti sve potrebne podatke o novom radniku. Podaci koji su u ovoj situaciji bili bitni su ime, prezime, poštanski broj i grad, adresa prebivališta, broj djece, OIB i bankovni račun, odnosno IBAN.

Nakon što se popune svi tekstualni okviri, pritisne se gumb „Unesi“ i podaci se spremaju u bazu podataka, točnije u tablicu „Radnik“. Za povratak na početni prozor se koristi gumb „Nazad“. Povratkom na početni zaslon i odabirom gumba „Postojeći radnik“ se otvara novi prozor koji je prikazan u nastavku.

```
1 reference
private void btn_Nazad_Click(object sender, EventArgs e)
{
    Close();

    Form1 form1 = new Form1();
    form1.Show();
}
```

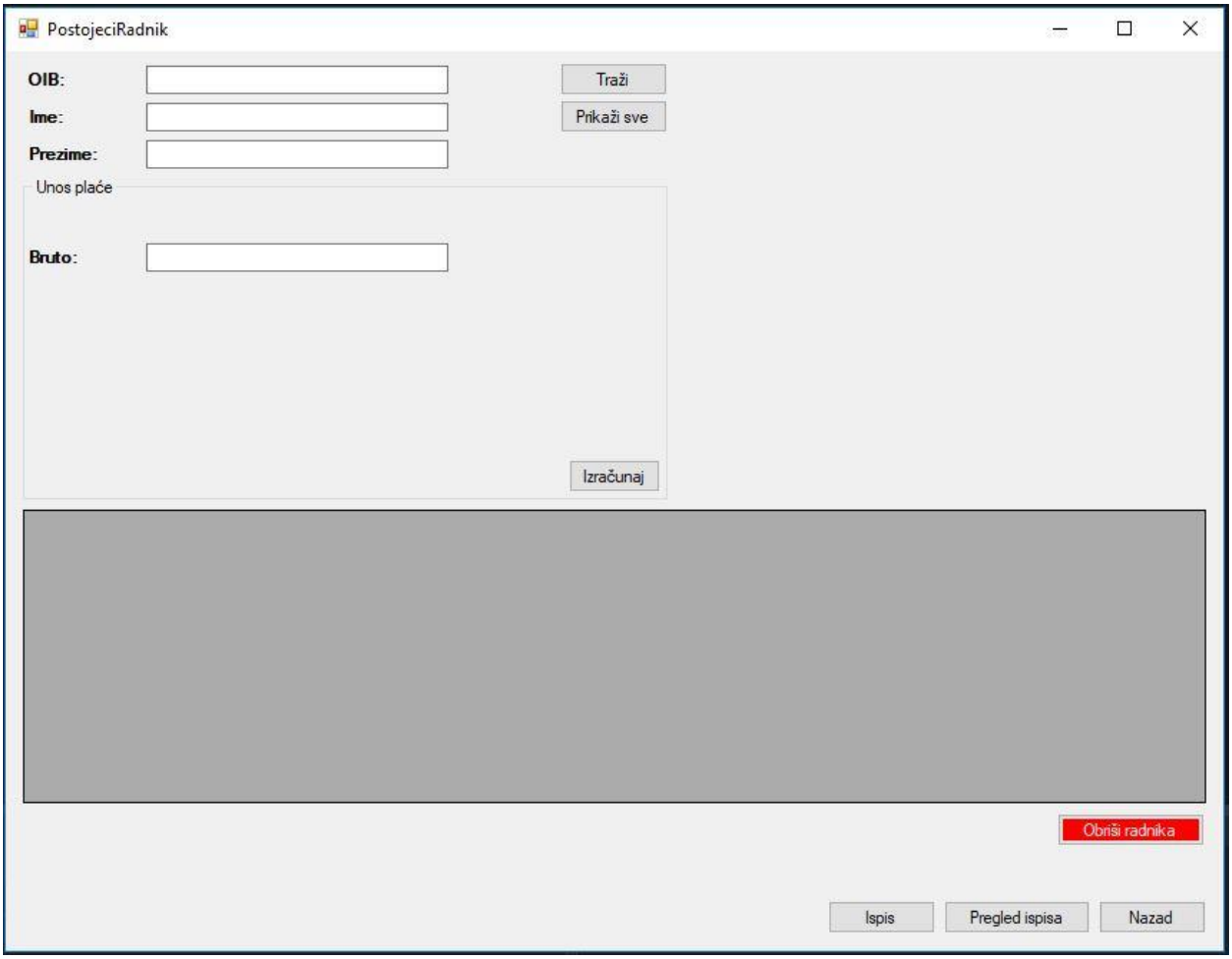
Slika 16. Kôd za povratak na početni prozor

Pritiskom na gumb „Nazad“ zatvara se prozor „NoviRadnik“, to se ostvaruje uz pomoć metode „Close()“ koja u potpunosti zatvara taj prozor nakon što su upisani svi podaci o novom radniku i otvara početni prozor „Form1“.

```
1 reference
private void btn_Unesi_Click(object sender, EventArgs e)
{
    LMSEntities context = new LMSEntities();
    Radnik radnik = new Radnik();
    try
    {
        radnik.Ime = Ime_textBox.Text;
        radnik.Prezime = Prezime_textBox.Text;
        radnik.Postanski_broj_i_grad = PostanskiBrojiGrad_textBox.Text;
        radnik.Adresa_prebivalista = AdresaPre_textBox.Text;
        radnik.Broj_djece = Convert.ToInt32(BrojDjece_textBox.Text);
        radnik.OIB = OIB_textBox.Text;
        radnik.Bankovni_racun = BankovniRacun_textBox.Text;

        context.Radniks.Add(radnik);
        context.SaveChanges();
    }catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Slika 17. Kôd za unos novog radnika

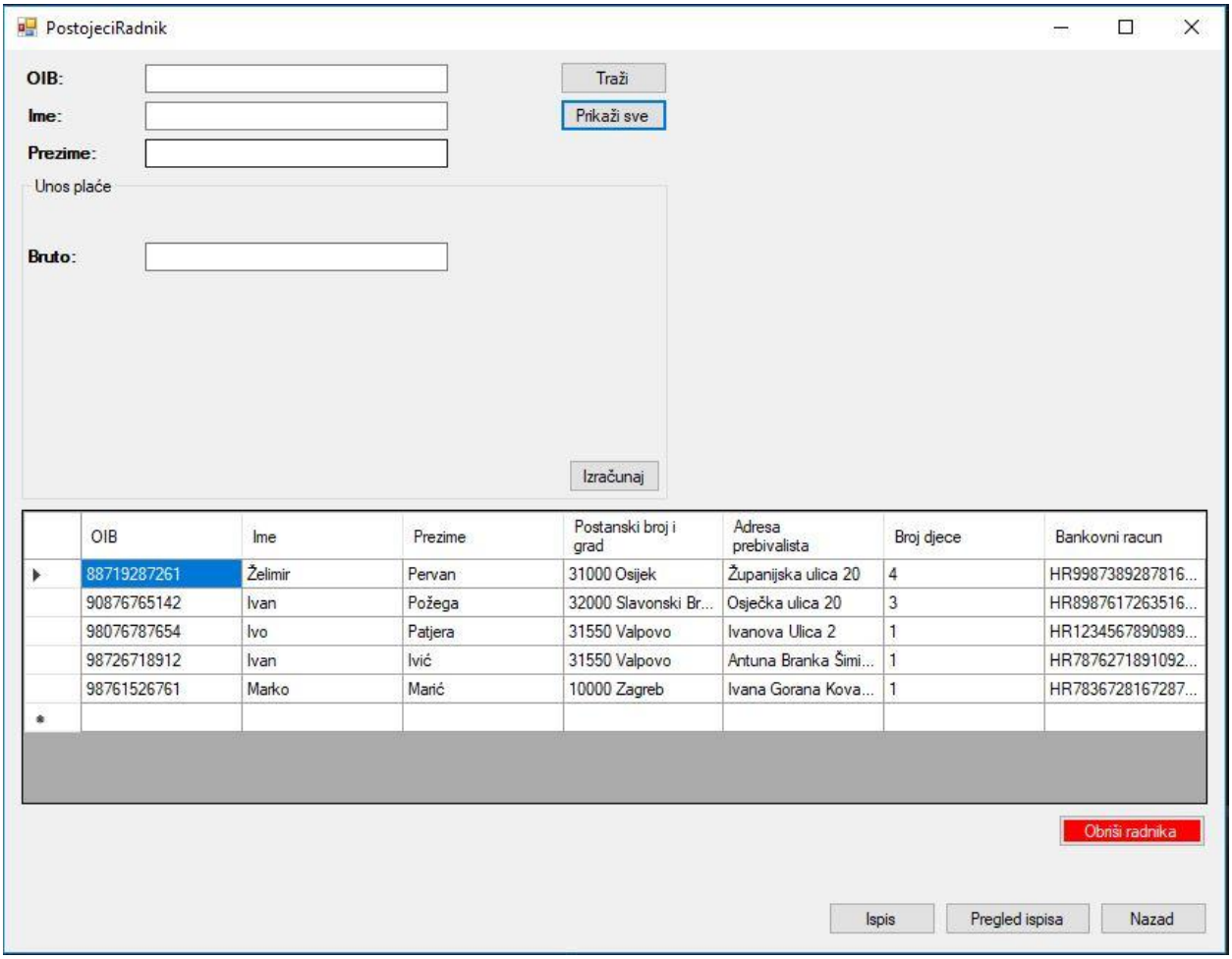
Novi radnik je definiran kao klasa koja je spojena na istoimenu tablicu uz pomoć „Library Management System“ alata, tako da nakon što su upisani potrebni podaci o radniku i pritisnut gumb „Unesi“, podaci se spreme u tablicu.



Slika 18. Izgled prozora „PostojeciRadnik“

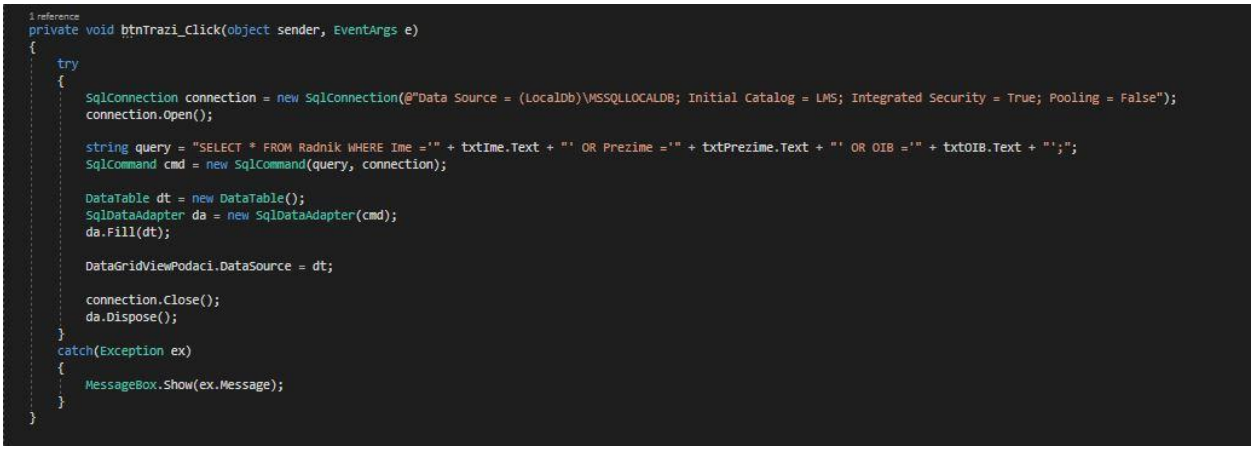
Nakon što se prozor „NoviRadnik“ otvori, vidimo u gornjem lijevom uglu da imamo tekstualne okvire uz pomoć kojih možemo pretraživati radnike po imenu, prezimenu i OIB-u. Ne mora se upisati sva 3 podatka nego možemo pretraživati samo po imenu, po prezimenu i po OIB-u.

Ispod tekstualnih okvira za pretraživanje imamo dio za unos bruto plaće i gumb za izračunavanje iste. Ako se pritisne gumb „Prikaži sve“, u sivom prostoru se prikazuju postojeći radnici. U nastavku je prikazan prozor nakon pritiska gumba „Prikaži sve“.



Slika 19. Izgled prozora „PostojeciRadnik“ nakon pritiska gumba „Prikaži sve“

Postojeći radnici se prikazuju u obliku tablice. Ispod sivog područja za prikaz postojećih radnika je gumb „Obriši radnika“ koji služi za direktno brisanje odabranog radnika iz tablice „Radnik“. Postupak je vrlo jednostavan, označavanjem reda kojeg zauzima određeni radnik i pritiskom gumba „Obriši radnika“, isti se briše iz baze podataka.

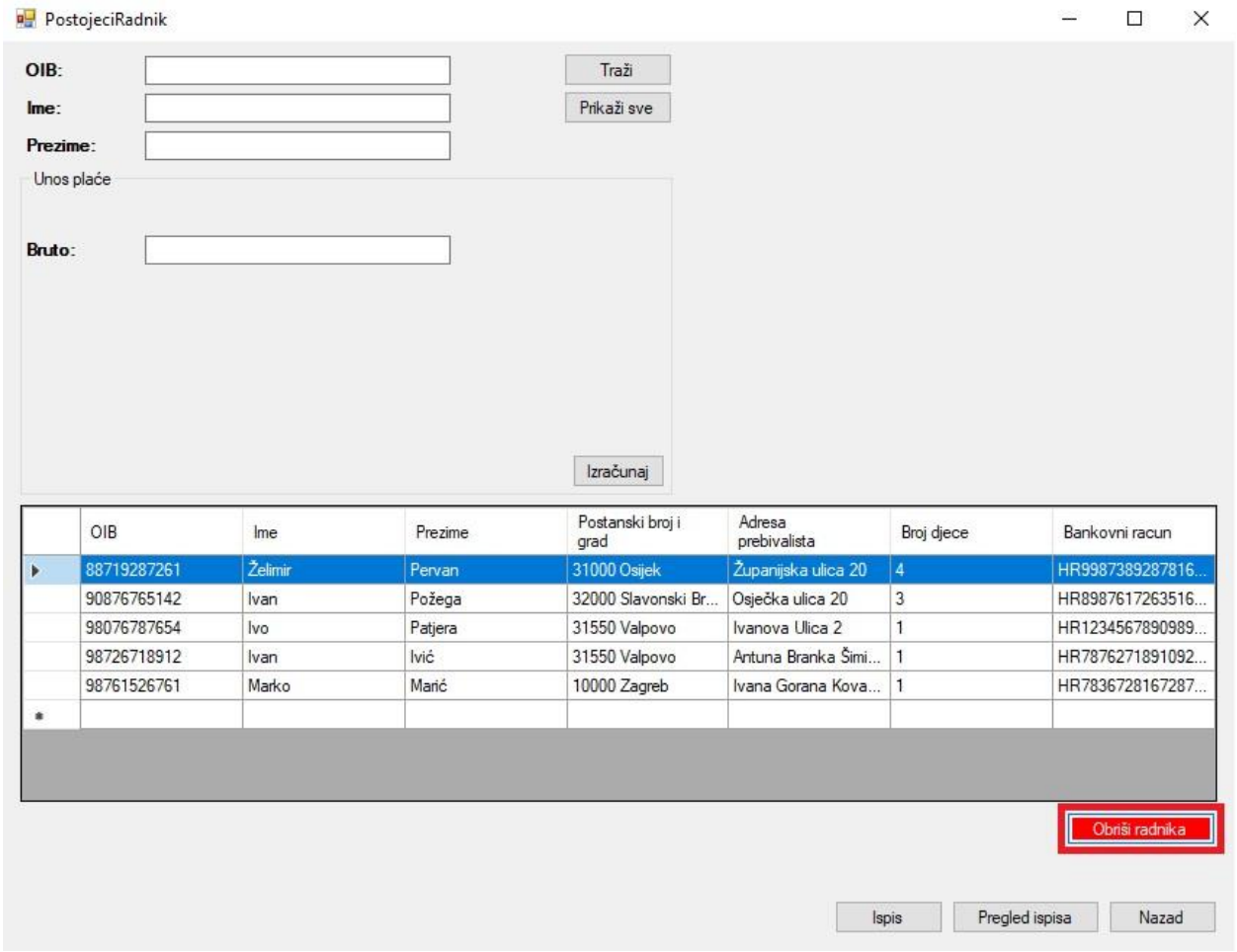


Slika 20. Kôd za pretraživanje radnika po imenu, prezimenu ili OIB-u

Pritiskom na gumb „Traži“ aplikacija se spaja na lokalnu bazu i izvršava se upit ovisno o tome koji podaci o radniku su upisani. Nakon pretrage se rezultati ispisuju u „DataGridView“ i zatvara se baza podataka.

Slika 21. Kôd za ispis svih radnika u DataGridView

Pritiskom na gumb „Prikaži sve“ se izvršava gore prikazani kôd i ispisuju se svi radnici koji se trenutno nalaze u tablici.



Slika 22. Brisanje odabranog radnika iz baze podataka

Prebacivanje podataka iz „DataGridView“ u sve ostale dijelove gdje treba koristiti odabranog radnika se izvršava pomoću metode „DataGridViewPodaci_CellClick“ koja je prikazana u nastavku.

```
1 reference
private void DataGridViewPodaci_CellClick(object sender, DataGridViewCellEventArgs e)
{
    Int32 selectedCellCount = DataGridViewPodaci.GetCellCount(DataGridViewElementStates.Selected);
    if(selectedCellCount > 0)
    {
        if (DataGridViewPodaci.AreAllCellsSelected(true))
        {
            MessageBox.Show("Sve ćelije su odabrane", "Odabrane ćelije");
        }
        else
        {
            odabraniRadnik.OIB = DataGridViewPodaci.SelectedCells[0].Value.ToString();
            odabraniRadnik.Ime = DataGridViewPodaci.SelectedCells[1].Value.ToString();
            odabraniRadnik.Prezime = DataGridViewPodaci.SelectedCells[2].Value.ToString();
            odabraniRadnik.PostanskiBrojiGrad = DataGridViewPodaci.SelectedCells[3].Value.ToString();
            odabraniRadnik.AdresaPrebivalista = DataGridViewPodaci.SelectedCells[4].Value.ToString();
            odabraniRadnik.BrojDjece = Convert.ToInt32(DataGridViewPodaci.SelectedCells[5].Value);
            odabraniRadnik.BankovniRacun = DataGridViewPodaci.SelectedCells[6].Value.ToString();
        }
    }
}
```

Slika 23. Kôd za prebacivanje podataka iz reda u klasu

Kada je odabran radnik iz „DataGridView“, isti se prebacuje u klasu „OdabraniRadnik“ kako bi bilo moguće dalje raditi s odabranim podacima.

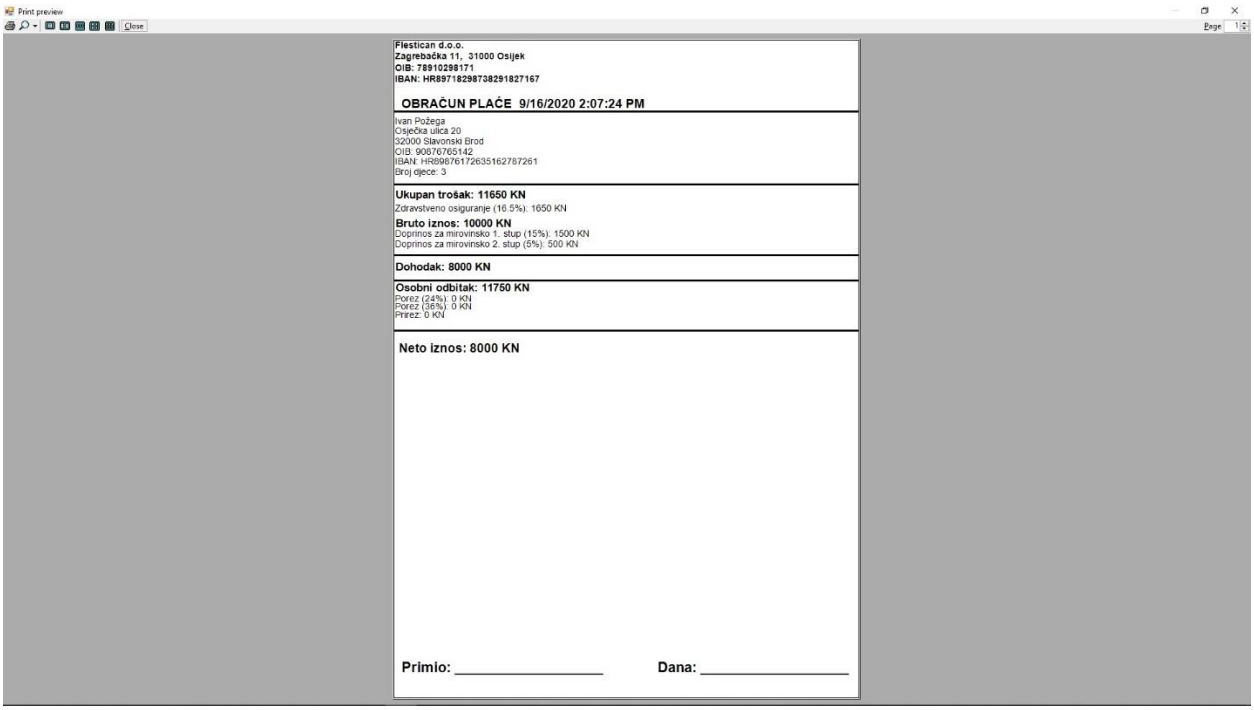
```
namespace KalkulatorPlace_zavrsni
{
    2 references
    public class OdabraniRadnik
    {
        3 references
        public string Ime { get; set; }
        2 references
        public string Prezime { get; set; }
        11 references
        public string PostanskiBrojiGrad { get; set; }
        2 references
        public string OIB { get; set; }
        2 references
        public string AdresaPrebivalista { get; set; }
        9 references
        public int BrojDjece { get; set; }
        2 references
        public string BankovniRacun { get; set; }
    }
}
```

Slika 24. Klasa „OdabraniRadnik“

```
1 reference
private void btn_Obrisi_Click(object sender, EventArgs e)
{
    try
    {
        SqlConnection connection = new SqlConnection(@"Data Source = (LocalDb)\MSSQLLOCALDB; Initial Catalog = LMS; Integrated Security = True; Pooling = False");
        connection.Open();

        string query = "DELETE FROM Radnik WHERE Ime ='" + odabraniRadnik.Ime + "';";

        SqlCommand cmd = new SqlCommand(query, connection);
        cmd.ExecuteNonQuery();

        string statement = "SELECT * FROM Radnik;";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(statement, connection);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        DataGridViewPodaci.DataSource = dataTable;

        connection.Close();
        dataAdapter.Dispose();
    }catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Slika 25. Kôd za brisanje odabranog radnika iz tablice

Za računanje obračuna je potrebno unijeti bruto iznos u tekstualni okvir iznad tablice s prikazom svih radnika. Nakon što se bruto iznos upisan i odabran radnik za kojega se računa obračun, pritiskom na gumb „Izračunaj“ se izračuna obračun plaće. Nakon što je gumb „Izračunaj“ pritisnut, ako kliknemo na „Pregled ispisa“, možemo vidjeti kako će izgledati sav proračun koji je vezan uz obračun plaće. U nastavku je slika s pregledom ispisa.



Slika 26. Pregled ispisa

```
1 reference
private void ButtonPrintPreview_Click(object sender, EventArgs e)
{
    PrintPreviewDialog.Document = PrintDocument;
    PrintPreviewDialog.ShowDialog();
}

1 reference
private void PrintDocument_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    UcitajTvrtku();

    try
    {
        e.Graphics.DrawString(tvrtka.NazivPoduzeca, new Font("Arial", 12, FontStyle.Bold), Brushes.Black, new Point(0, 0));
        e.Graphics.DrawString(tvrtka.Adresa + ",", new Font("Arial", 12, FontStyle.Bold), Brushes.Black, new Point(0, 20));
        e.Graphics.DrawString(tvrtka.Grad, new Font("Arial", 12, FontStyle.Bold), Brushes.Black, new Point(130, 20));
        e.Graphics.DrawString("OIB: " + tvrtka.OIB, new Font("Arial", 12, FontStyle.Bold), Brushes.Black, new Point(0, 40));
        e.Graphics.DrawString("IBAN: " + tvrtka.IBAN, new Font("Arial", 12, FontStyle.Bold), Brushes.Black, new Point(0, 60));

        e.Graphics.DrawString("OBRAČUN PLAĆE", new Font("Arial", 16, FontStyle.Bold), Brushes.Black, new Point(10, 100));
        e.Graphics.DrawString(DateTime.Now.ToString(), new Font("Arial", 16, FontStyle.Bold), Brushes.Black, new Point(220, 100));
        e.Graphics.DrawString("_____", new Font("Arial", 25, FontStyle.Bold), Brushes.Black, new Point(-5, 90));

        e.Graphics.DrawString(odabraniRadnik.Ime + " " + odabraniRadnik.Prezime, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 135));
        e.Graphics.DrawString(odabraniRadnik.AdresaPrebivalista, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 153));
        e.Graphics.DrawString(odabraniRadnik.PostanskiBrojiGrad, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 171));
        e.Graphics.DrawString("OIB: " + odabraniRadnik.OIB, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 189));
        e.Graphics.DrawString("IBAN: " + odabraniRadnik.BankovniRacun, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 207));
        e.Graphics.DrawString("Broj djece: " + odabraniRadnik.BrojDjece, new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 225));
        e.Graphics.DrawString("_____", new Font("Arial", 25, FontStyle.Bold), Brushes.Black, new Point(-5, 220));

        e.Graphics.DrawString("Ukupan trošak: " + (Convert.ToDouble(BrutoIznos_textBox.Text) + (Convert.ToDouble(BrutoIznos_textBox.Text) * elementiPlace.Zdravstveno)) + " KN", new Font("Arial", 14, FontStyle.Bold), Brushes.Black, new Point(0, 265));
        e.Graphics.DrawString("Zdravstveno osiguranje (16.5%): " + (Convert.ToDouble(BrutoIznos_textBox.Text) * elementiPlace.Zdravstveno) + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 290));
        e.Graphics.DrawString("Bruto iznos: " + BrutoIznos_textBox.Text + " KN", new Font("Arial", 14, FontStyle.Bold), Brushes.Black, new Point(0, 315));
        e.Graphics.DrawString("Doprinos za mirovinsko 1. stup (15%): " + elementiPlace.MIOstup1 + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 335));
        e.Graphics.DrawString("Doprinos za mirovinsko 2. stup (5%): " + elementiPlace.MIOstup2 + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 353));
        e.Graphics.DrawString("_____", new Font("Arial", 25, FontStyle.Bold), Brushes.Black, new Point(-5, 345));

        e.Graphics.DrawString("Dohodak: " + elementiPlace.Dohodak + " KN", new Font("Arial", 14, FontStyle.Bold), Brushes.Black, new Point(0, 393));
        e.Graphics.DrawString("_____", new Font("Arial", 25, FontStyle.Bold), Brushes.Black, new Point(-5, 390));

        e.Graphics.DrawString("Osobni odbitak: " + elementiPlace.odbitakzaOdjecu + " KN", new Font("Arial", 14, FontStyle.Bold), Brushes.Black, new Point(0, 430));
        e.Graphics.DrawString("Porez (24%): " + elementiPlace.Porez1 + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 450));
        e.Graphics.DrawString("Porez (36%): " + elementiPlace.Porez2 + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 465));
        e.Graphics.DrawString("Prirez: " + elementiPlace.Prirez + " KN", new Font("Arial", 12, FontStyle.Regular), Brushes.Black, new Point(0, 480));
        e.Graphics.DrawString("_____", new Font("Arial", 25, FontStyle.Bold), Brushes.Black, new Point(-5, 480));

        e.Graphics.DrawString("Neto iznos: " + elementiPlace.netoPlaca + " KN", new Font("Arial", 16, FontStyle.Bold), Brushes.Black, new Point(5, 535));
        e.Graphics.DrawString("Primio: _____                Dana: _____", new Font("Arial", 18, FontStyle.Bold), Brushes.Black, new Point(10, 1100));
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

}
```

Slika 27. Kôd za ispis i pregled ispisa

Slika prikazana iznad nam pokazuje kôd za ispis dokumenta na printer i pregled ispisa. Raspored elemenata ispisa je napravljen uz pomoć klase „Point“, gdje se unose X i Y koordinate za položaj na papiru. Prije svega se učitavaju podaci za tvrtku koji su spremljeni iz baze u istoimenu klasu „Tvrtka“.

```
namespace KalkulatorPlace_zavrsni
{
    2 references
    public class Tvrtka
    {
        2 references
        public string OIB { get; set; }
        2 references
        public string Adresa { get; set; }
        2 references
        public string NazivPoduzeca { get; set; }
        2 references
        public string Grad { get; set; }
        2 references
        public string IBAN { get; set; }
    }
}
```

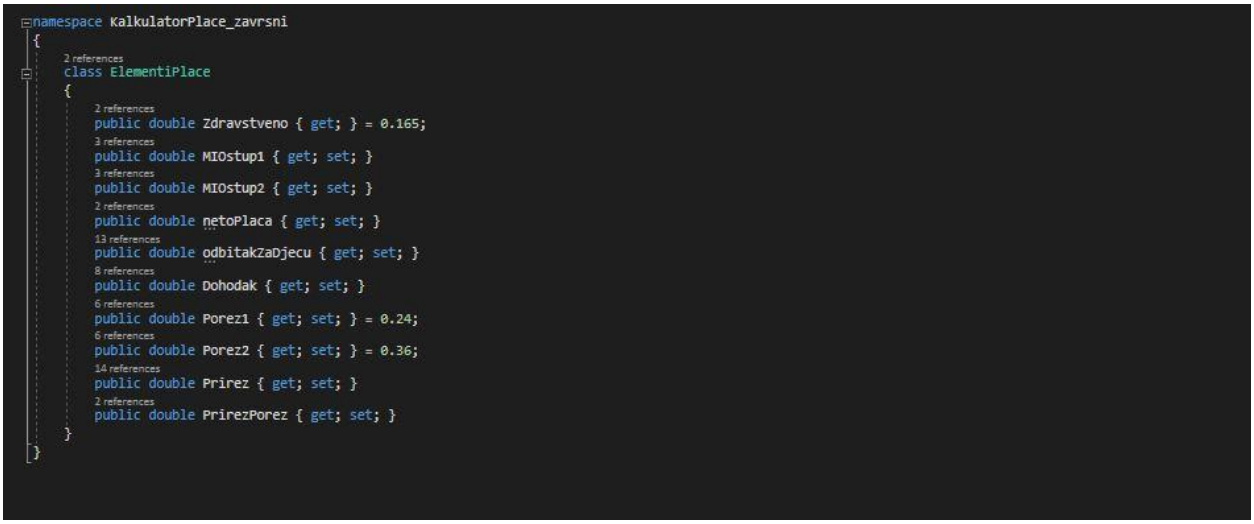
Slika 28. Klasa „Tvrtka“

U klasu „Tvrtka“ se spremaju podaci iz tablice „Tvrtka“, ona se ne mijenja jer je pretpostavka da je aplikacija rađena za jednu tvrtku.

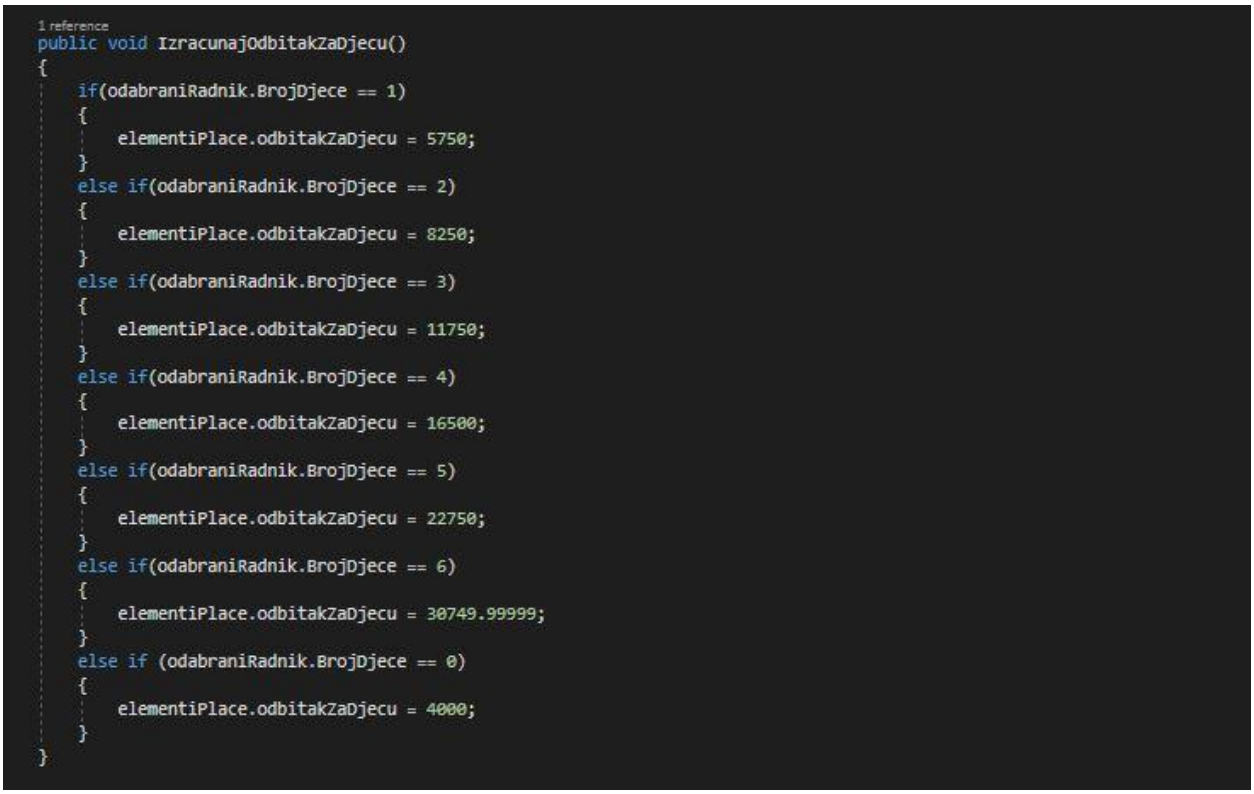
```
1 reference
private void Btn_IzracunajPlacu_Click(object sender, EventArgs e)
{
    try
    {
        elementiPlace.MIOstup1 = Convert.ToDouble(BrutoIznos_textBox.Text) * 0.15;
        elementiPlace.MIOstup2 = Convert.ToDouble(BrutoIznos_textBox.Text) * 0.05;
        elementiPlace.Dohodak = Convert.ToDouble(BrutoIznos_textBox.Text) - (elementiPlace.MIOstup1 + elementiPlace.MIOstup2);
        IzracunajOdbitakZaDjecu();
        DohvatiPrirez();

        if (elementiPlace.odbitakZaDjecu > elementiPlace.Dohodak)
        {
            elementiPlace.Porez1 = 0;
            elementiPlace.Porez2 = 0;
            elementiPlace.Prirez = 0;
        }
        else if ((elementiPlace.Dohodak - elementiPlace.odbitakZaDjecu) <= 30000)
        {
            elementiPlace.Porez1 = (elementiPlace.Dohodak - elementiPlace.odbitakZaDjecu) * 0.24;
            elementiPlace.Porez2 = 0;
        }
        else if ((elementiPlace.Dohodak - elementiPlace.odbitakZaDjecu) > 30000)
        {
            RacunajPorez2();
        }

        elementiPlace.Prirez = (elementiPlace.Porez1 + elementiPlace.Porez2) * elementiPlace.Prirez;

        elementiPlace.PrirezPorez = elementiPlace.Prirez + elementiPlace.Porez1 + elementiPlace.Porez2;

        elementiPlace.netoPlaca = elementiPlace.Dohodak - elementiPlace.PrirezPorez;
    }catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Slika 29. Kôd za izračun

Gore prikazani kôd je povezan s gumbom „Izračunaj“. Pritiskom na navedeni gumb se provodi obračun plaće, računa se dohodak za djecu ovisno o broju djece, mirovinsko osiguranje 1. i 2. stup, dohodak, porez 1 i 2 i prirez ovisno o gradu. Izračunati elementi se spremaju u klasu „ElementiPlace“.



Slika 30. Klasa „ElementiPlace“



Slika 31. Metoda za izračunavanje odbitka za djecu

Dohvaćanje iznosa prireza se dobiva na način da se provjeri koji grad određeni radnik ima u stupcu „PostanskiBrojiGrad“, prikazano je na slici 32.

```
1 reference
public void DohvatiPrirez()
{
    if (odabraniRadnik.PostanskiBrojiGrad.Contains("Zagreb"))
    {
        elementiPlace.Prirez = 0.18;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Osijek"))
    {
        elementiPlace.Prirez = 0.13;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Slavonski Brod"))
    {
        elementiPlace.Prirez = 0.12;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Valpovo"))
    {
        elementiPlace.Prirez = 0.08;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Našice"))
    {
        elementiPlace.Prirez = 0.08;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Donji Miholjac"))
    {
        elementiPlace.Prirez = 0.05;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Beli Manastir"))
    {
        elementiPlace.Prirez = 0.05;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Belišće"))
    {
        elementiPlace.Prirez = 0.07;
    }
    else if (odabraniRadnik.PostanskiBrojiGrad.Contains("Požega"))
    {
        elementiPlace.Prirez = 0.07;
    }
}
```

Slika 32. Metoda za dohvaćanje prireza

5. Zaključak

U samom opisu zadatka je spomenuto kako su prije obračuni trajali puno duže jer se sav izračun radio na papiru. Sada kada smo došli u moderno i razvijeno doba imamo si mogućnost olakšati i ubrzati taj proces. Unosom bruto plaće, odabirom radnika i klikom na gumb se sve izračuna i imamo mogućnost ispisati obračun.

Aplikacija je napisana programskim jezikom C# u Microsoft Visual Studio razvojnom okruženju, taj programski jezik i razvojno okruženje su odabrani u kombinaciji jer uvelike olakšavaju izradu i odgovaraju postavljenom zadatku uz trenutačno stečeno znanje i iskustvo o programiranju.

Literatura

[1] A tour od the C# language, https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ rujan 2020.

[2] ADO.NET Overview, https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview rujan 2020.

[3] Osnove baza podataka, https://support.microsoft.com/hr-hr/office/osnove-baza-podataka-a849ac16-07c7-4a31-9948-3c8c94a7c204 rujan 2020.

[4] What is .NET Framework?, https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework rujan 2020.

Sažetak

Naslov: Windows Form aplikacija za obračun plaće

U ovom završnom radu su opisane .NET tehnologije, Microsoft Visual Studio razvojno okruženje i baze podataka. Prikazan je razvoj aplikacije za obračun plaće uz pripadajuće slike kôda i opis funkcionalnosti. Aplikacija je prilično jednostavna za korištenje i ima mjesta za proširenja i nadogradnje. Krajnji korisnik može upisivati i brisati podatke iz tablice „Radnik“, ali tablicu „Tvrtka“ ne može mijenjati. Za razvoj aplikacije je korišteno razvojno okruženje Microsoft Visual Studio Enterprise 2017., kôd je pisan u C# programskom jeziku i aplikacija je spojena na lokalnu bazu podataka.

Ključne riječi: baza podataka, C#, Microsoft Visual Studio, SQL, Windows Form App

Abstract

Title: Windows Form Application for calculating salary

This final paper describes .NET technologies, the Microsoft Visual Studio IDE and databases. The development of the application for calculating the salary is presented with the corresponding code images and description of the functionality. The application is pretty easy to use and has room for extensions and upgrades. The end user can enter and delete dana from the „Employee“ table, but the „Company“ table cannot be changed. The Microsoft Visual Studio Enterprise 2017 development environment was used to develop the application, the code was written in C# programming language and the application was connected to a local database.

Keywords: C#, database, Microsoft Visual Studio, SQL, Windows Form App