Aplikacija za popunjavanje obrazaca

Aleksić, Davor

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

Stručni studij

APLIKACIJA ZA POPUNJAVANJE OBRAZACA

Završni rad

Davor Aleksić

Osijek, 2016

Sadržaj

1. UVOD

Cilj ovog završnog rada je realizacija aplikacije za popunjavanje obrazaca. Ovaj zadatak sam odabrao iz razloga što ću raditi s programskim okruženjima i programskim jezicima koji me zanimaju i s kojima ću raditi u budućnosti. Ideja završnog rada je da se preko aplikacije za popunjavanje obrazaca popuni životopis i klikom na gumb spremi, taj obrazac bi se spremio kao PDF dokument. Dakle, korisnik bi taj dokument koristio kao pomoć u pronalasku zapošljavanja na željeni posao. Ova aplikacija bi korisniku omogućila jednostavan, brz i efikasan dolazak do željenog životopisa. Prvi korak pri izradi je bio odabir programskog okruženja i programskog jezika s kojim će se izraditi aplikacija. Budući da se radi o desktop aplikaciji zbog mnogobrojnih primjera i jednostavnosti izrade odabran je programski jezik C# uz programsko okruženje Visual Studio 2015. Također, na zahtjev zadatka potrebno je raditi i u LaTeX programskom jeziku za pisanje teksta koji nam daje predložak obrasca ispisan u PDF dokumentu. LaTex alat za uređivanje teksta je pisan u MiKTeX programskom okruženju. U drugom poglavlju biti će opisana programska okruženja i programski jezici preko kojih se aplikacija izradila te će biti postavljeni osnovni primjeri koda svakog programskog jezika. S trećim dijelom završnog rada obuhvaćena je realizacija aplikacije i problematika pisanja izvornog koda za aplikaciju.

1.1 Zadatak završnog rada

Zadatak završnog rada je napraviti desktop aplikaciju pomoću Visual Studio 2015 programskog okruženja. Aplikacija treba ponuditi tekst i slike kao polja za unos te generirati pdf dokument s popunjenim obrascem. Predložak obrasca treba biti u LaTeX programu za pisanje teksta. Također nakon generiranja PDF dokumenta aplikacija mora imati bazu u koju će spremiti životopise kako bi se mogli ponovno učitati i prepraviti ako je potrebno.

2. PRIMJENJENE TEHNOLOGIJE I ALATI

U ovom poglavlju opisane su tehnologije koje su korištene za izradu aplikacije. Desktop aplikacija za popunjavanje obrazaca izrađena je pomoću C#, LaTeX i MySQL programskih jezika. Uz programska okruženja Visual Studio 2015 Community i MiKTeX 2.9. Programski jezik C# je korišten za kreiranje desktop aplikacije kojom bi klikom na gumb „Spremi kao PDF“ spremili PDF dokument na željenu lokaciju na računalu. Ispis PDF dokumenta omogućio nam je LaTeX alat za pisanje teksta. Spremljeni PDF dokumenti osim što su memorirani na neku lokaciju na lokalnom disku također su spremljeni u bazu podataka iz koje se mogu ponovno učitati i uređivati. U idućim pod poglavljima će pobliže biti predstavljeni navedeni programski jezici i razvojna okruženja.

Slika 2.1. Programski alati i jezici korišteni za generiranje PDF datoteke

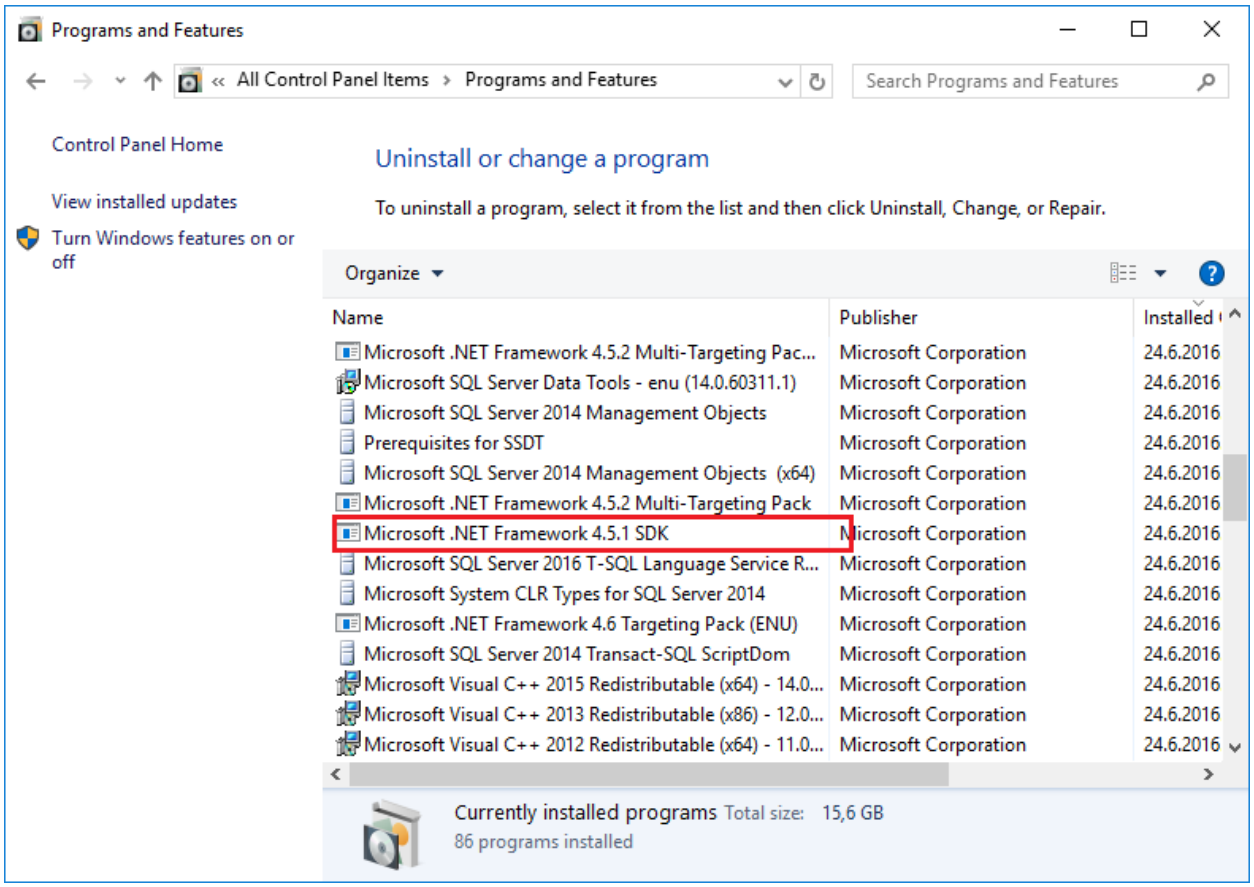
2.1. Programski jezik C#

Programski jezik C# je objektno-orijentirani programski jezik opće namjene. Razvio ga je Microsoft unutar .NET inicijative. Dobio je ime po glazbenoj noti koja ukazuje da napisana nota treba napraviti poluton višeg tona. C# teži biti moderan i jednostavan programski jezik. C# ima za prethodnike C i C++ od kojih je uzeo sve stvari koje su bile dobre i koje nisu zahtijevale poboljšanja. Izrazi, naredbe i skoro cijela sintaksa, što čini većinu tipičnih programa, je ostala nepromijenjena. Zbog toga je C# portabilan programski jezik te za programere koji rade u C ili C++ programskom jeziku nije teško savladati C#. [1]

2.1.1. Primjer jednog koda u C# programskom jeziku

U ovom poglavlju bit će predstavljen najjednostavniji C# kod i ono što nam je potrebno da krenemo s C# programiranjem. Po nepisanom pravilu svi primjeri počinju s tradicionalnim programom „Hello World“.

Prije nego bi mogli početi koristiti C# programski jezik na nekom računalu potrebno je imati instaliran Microsoft .NET Framework SDK, ali ako je već prethodno instaliran Visual Studio .NET računalo bi trebalo sadržavati .NET Framework. Ukoliko niste sigurni da je .NET Framework instaliran može se provjeriti na upravljačkoj ploči kao što je prikazano na slici.[2]



Slika 2.2. Microsoft .NET Framework SDK instaliran na računalu

Kako bi se napisao programski kod potrebno je programsko okruženje, u ovom slučaju za najjednostavniji program korišten je Visual Studio 2015.

```
using System;
public class HelloWorld
{
    public static void Main()
    {
        // komentar u jednoj liniji
        Console.WriteLine("Hello World!" );
        /*
        komentar
        vise
        linija
        */
   }
}
```

Nakon prevođenja (kompajliranja) aplikacija će nam na zaslonu ispisati tekst „Hello World“. Iz ovog primjera treba uočiti slijedeće karakteristike:

- mala/velika slova - C# pravi razliku između velikih i malih slova, tzv. case-sensitive. Npr. ako bi umjesto Console upisali console program se ne bi preveo,
- komentari - pišu se isto kao u C++ programskom jeziku, mogu biti jedno linijski i li više linijski,
- Console.WriteLine("Hello World!"); - naredba u kojoj je smješten cijeli kod i samo nam je ta naredba potrebna da bi se na zaslonu ispisao tekst „Hello World“,
- public class HelloWorld - deklaracija klase,
- using System - ova naredba omogućava da se nadalje u kodu koriste sve metode iz prostora System, a da se ne navodi riječ System. Tako se klasa Console koja pripada prostoru System može pozvati s

  `Console.WriteLine("Hello World!" );`

  umjesto

  `System.Console.WriteLine("Hello World!" );,`
- C# je objektno-orijentiran programski jezik i sve naredbe moraju stajati u klasama
- Main - u klasi HelloWorld nalazi se funkcija Main. Ova funkcija je deklarirana kao static i public. To je znak da program može imati samo jednu ulaznu točku i ona se ostvaruje preko ove funkcije,
- prevođenje (Kompajliranje) koda - izvršavanje ovog koda zahtjeva prevođenje koda na jezik razumljiv računalu.

2.2. Microsoft Visual Studio 2015

Programsko okruženje u kojem sam radio aplikaciju je Visual Studio 2015. Izrađeno je od strane Microsofta, a koristi se za razvoj aplikacija, programa za Windows i web stranica. Visual Studio podržava različite programske jezike, a neki od tih su: C, C++, C++/CLI (preko Visual C++), VB .NET (preko Visual Basic .NET-a), C# (preko Visual C#) i F#. Podrška za ostale programske jezike dostupna je instalacijom jezičnih servisa. Dodatni programski jezici koji mogu biti dostupni preko Visual Studia su: Python i Ruby, a također podržava XML, JavaScript i CSS. U ovom završnom radu je korišten Microsoft Visual Studio 2015 Community iz razloga što je ta verzija besplatna i dostupna privatnom korištenju.

Kako služi za razne programske jezike tako je Microsoft izdao više različitih verzija Visual Studia, a neka od njih su:

- Community - nastao u studenom 2014., besplatna verzija koja podržava više programskih jezika i omogućava podršku i nadogradnju. Namijenjena je individualnim programerima i manjim razvojnim timovima.
- Professional - osim svih programskih jezika koji su prisutni omogućava i podršku za Windows phone operacijske sustave
- Enterprise - pruža skup softvera i baza podataka te obogaćen dodatnim alatima za mjerenja, ispitivanja, arhitekturu uz sve značajke koje ima Visual Studio Professional
- Express - pruža samo mali skup alata.

2.3. LaTeX alat za pisanje teksta

LaTeX (čita se Latek) je alat za dokumente i sustav za stvaranje dokumenata korištenjem uređivačkog programa TeX. Koristi se u izdavačkim tvrtkama kao i u akademskoj zajednici, posebno zbog kvalitete krajnjeg produkta i bolje automatizacije tablice, figura i bibliografija. S ovim programskim jezikom za uređivanje teksta planiram napraviti PDF koji bi se ispisao klikom na gumb „Spremi kao PDF“. Postoji nekoliko različitih verzija TeX programa. Razlikuju se ovisno o operacijskom sustavu za kojeg su napisane. Zatim neke verzije imaju specijalizirana grafička sučelja, dok su druge zamišljene kao tekstualni uređivač u kojem pišemo izvorni kod te ga kasnije prevedemo u konačni dokument[4]. Neke od najpopularnijih verzija su:

- TeX Live - ova TeX verzija je standardna u Linux operacijskim sustavima. Multiplatformska je verzija koja osim Linux operacijskog sustava podržava i još neke operacijske sustave kao što su: AIX, FreeBSD, MAC OS X, Solaris, Windows.
- MiKTeX - verzija za Windows operacijske sustave s kojom će se raditi u ovom završnom radu.
- MaCTeX - verzija za MAC OS X operacijski sustav, a temelji se na TeX Live-u.

2.3.1. Način rada u LaTeX alatu za uređivanje teksta

Format svakog LaTeX izvornog koda se sastoji od zaglavlja i oblikovanja tijela dokumenta te mora sadržavati tri slijedeće naredbe:

```
\documentclass[12pt]{book}
\begin{document}
\end{document}
```

Kao što je u primjeru vidljivo zaglavlje se sastoji od sve što je iznad „/begin{document}“ naredbe. Tu se nalaze stvari kao što su definicija stilova dokumenata, postavke razmaka između odjeljaka, postavke margina, brojevi stranica.

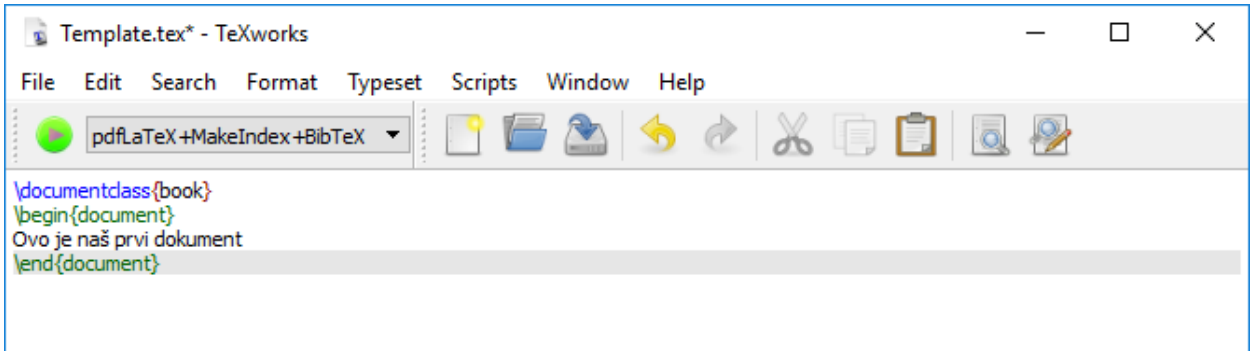
Prvi dio od ove tri naredbe nam određuje klasu i mogućnosti. Neke od standardnih klasa su: article, book, letter, report, slides. Mogućnosti se mogu izostaviti, pa bi se u tom slučaju koristila standardna postavka pojedine mogućnosti. Primjer toga je standardna postavka veličine slova 10 pt, ali ona se po volji može promijeniti na 12pt, 14pt itd. Moguće je staviti više mogućnosti koje se odvajaju zarezom bez razmaka. Neke od mogućnosti koje mogu biti napisane u našem LaTeX zaglavlju su:

- landscape - formatira izlaz za ispis u obliku pejzaža, ova opcija efektivno zamijeni širinu i duljinu papira
- letterpaper - formatira izlaz na odgovarajuću veličinu papira
- onecolumn|twocolumn - određuje hoće li se tekst prikazivati u jednom ili u dva stupca
- oneside/twoside - mogućnost kojom LaTeX formatira izlaz za ispis teksta na obje strane papira ili samo na jednu stranu. „twoside“ se najčeće koristi u kodovima gdje je klasa „book“.

```
\documentclass["mogućnosti"]{"klasa"}
```

Slika 2.3. Određivanje mogućnosti i klase

Drugi dio naredbi su tijela koja oblikuju izgled dokumenta. Naredba za početak dokumenta je `\begin{document}` dok je je naredba za kraj dokumenta `\end{document}`. Između ovih naredbi se piše proizvoljni tekst koji želimo da naš dokument sadrži. Na kraju klikom na prevođenje koda ispiše se PDF s našim tekstom.



Slika 2.4. Ovaj će program ispisati u PDF: „Ovo je naš prvi dokument“.

2.4. SQLite programski jezik

SQL (Structured Query Language) je programski jezik za izradu, ažuriranje, traženje i brisanje podataka iz baza podataka. U završnom radu je korišten iz razloga što je aplikacija trebala sadržati bazu za spremanje životopisa iz koje bi kasnije učitali te životopise koji bi se mogli ponovno uređivati. SQLite sustav je izabran iz razloga što je najpopularniji izbor za lokalnu uporabu u aplikacijama. Besplatan i open source sustav za upravljanje bazom podataka. Također, SQLite baza vrlo je optimizirana i stabilna te sadržana u C programskim bibliotekama.[5]

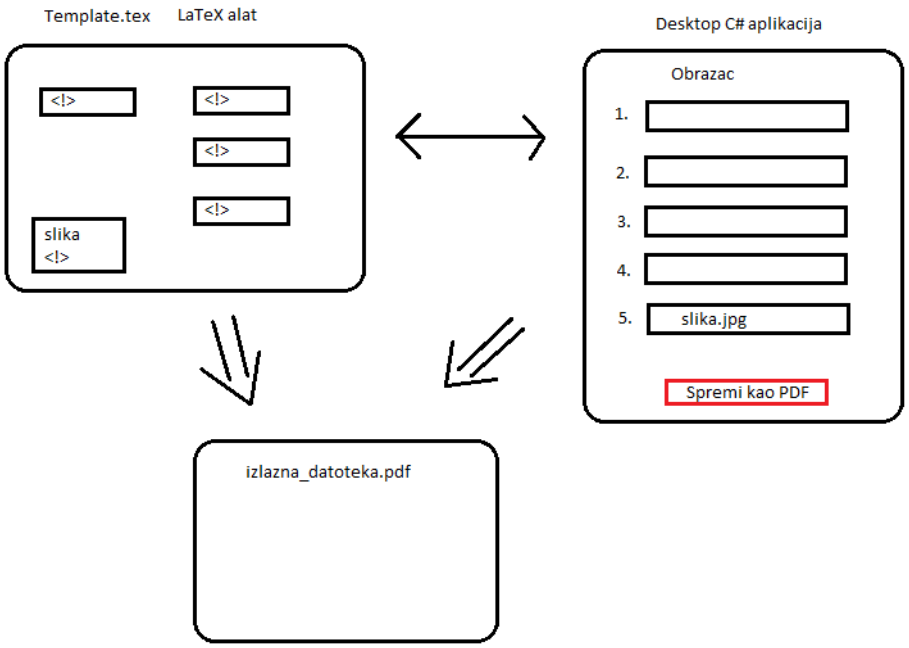
2.4.1 Glavne značajke SQLite relacijske baze

SQLite je relacijska baza, a najviše se primjenjuje u web stranicama, aplikacijama te ad hoc disk datotekama. Glavne značajke ove baze su:

- Nema potrebu za administracijom
- Podaci se ostaju u bazi pri nekom izvanrednom događaju (padu sustava ili nestanku električne energije)
- Cijela baza je smještena u jednoj datoteci na disku
- Jednostavno korisničko sučelje
- Open source kod

3. REALIZACIJA APLIKACIJE

Zadatak završnog rada je izrada aplikacije koja treba ponuditi tekst i sliku kao polja za unos te generirati pdf dokument s popunjenim obrascem. Za realizaciju su bili potrebni alati i programski jezici koji su predstavljeni u drugom poglavlju. Pomoću C# programskog jezika izradila se aplikacija dok se pomoću LaTeX programskog jezika omogućio ispis PDF dokumenta u željenom formatu i željenom poretku slike i teksta. Osim što aplikacija sprema obrazac kao PDF dokument, također sprema taj popunjeni obrazac u bazu podataka koja ga čuva kako bi se kasnije mogao ponovno učitati i izmijeniti. Početak stvaranja aplikacije je bio riješiti problematiku povezivanja C# programskog kod s LaTeX izvornim kodom. Rješenje se pronašlo u posebnim placeholderima koji su prevodili LaTeX-u gdje treba upisati ispunjena polja iz aplikacije u PDF dokument. Placeholderi koji povezuju C# i LaTeX će biti objašnjeni nešto kasnije u radu. Nakon definiranja placeholdera bilo je potrebno odrediti od koliko će se polja za unos teksta aplikacija sastojati i što će se sve unositi u obrazac.



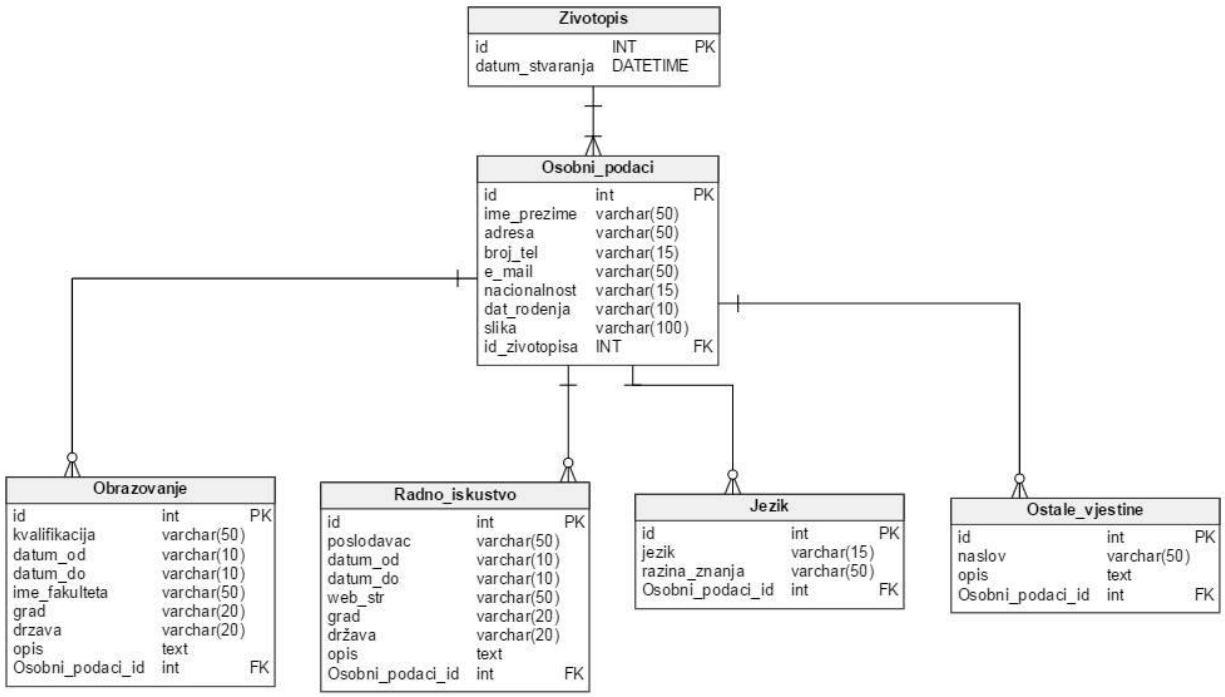
3.1. Početna ideja aplikacije

Aplikacija je dizajnirana tako da se sastoji od šest kategorija, a one su: osobni podaci, obrazovanje, jezik, radno iskustvo, ostale vještine i postojeći zapisi. Svaka od kategorija osim „postojećih zapisa“ sadrži polja za upis teksta ili u slučaju cjeline „osobni podaci“, unos slike i teksta. Prvi korak pri realizaciji aplikacije je bio definirati sam njen izgled i odrediti formu kako će se funkcije u aplikaciji izvršavati. Ove dvije radnje su se odrađivale paralelno. Za dizajn aplikacije korištena je drag&drop metoda koja je u Windows forms-u. Windows forms je grafička biblioteka koja je dio Microsoft .NET framework-a.

Sama struktura koda je zasnovana na događajima koji odgovaraju na pitanje „što bi moglo biti ako se klikne određeni gumb“. Počinje s otvaranjem glavne forme koja je nazvana „form1“. Definiraju se globalne varijable i klase koje će se kasnije koristiti za dodavanje objekata u ListBox. Poslije inicijalizacije osnovne forme postavlja se naslov koji će se nalaziti u zaglavlju glavnog prozora aplikacije. Zatim se definira se baza podataka i kreiraju se tablice potrebne za spremanje podataka. Također, stvara se metoda za dodavanje podataka u bazu podataka koja će biti objašnjena u poglavlju gdje se govori o funkcioniranju baze podataka. Kada se definirao rad baza podataka potrebno je definirati i rad same aplikacije, pa se tako u strukturi koda mogu naći metode za odabir slike i metode kada se dodaju ili brišu podaci iz „listbox-a“ u određenim kategorijama kao i metoda upozoravajućih poruka ako se ne ispuni određeno polje. Glavni dio koda je samo izvršavanje programa i zamjenjivanje placeholdera u LaTeX-u s tekstom u unesenim poljima. Svi nabrojani dijelovi će biti objašnjeni u idućim poglavljima.

3.1. Funkcioniranje baze podataka

Baza podataka je rađena pomoću SQLite sustava za baze podataka. Uz izvršavanje programa čini glavninu koda aplikacije. Baza podataka u ovoj aplikaciji nam služi za spremanje, učitavanje postojećih zapisa te brisanje istih. Vidljiva je u lijevom dijelu aplikacije pod kategorijom „Postojeći zapisi“ gdje je još vidljiv ID, Ime i prezime i Datum stvaranja. Tri nabrojana atributa ujedno čine i strukturu tablice u bazi podataka pod nazivom Životopis.sqlite. Osim tablice Životopis postoji još pet tablica. Sve tablice imaju međusobni odnos 1:N što znači da u svakoj tablici postoji po jedan primarni ključ i jedan strani ključ. Glavna tablica na koju se sve povezuje je tablica Osobni_podaci koja osim svojih atributa ima strani ključ id_zivotopisa kako bi se prepoznalo koji podaci idu pod određeni spremljeni zapis. Ostale tablice imaju strani ključ iz Osobni_podaci_id jer je pri spremanju baze potrebno znati pod koje osobne podatke idu kategorije Obrazovanje, Radno_iskustvo, Jezik i Ostale_vještine.



Slika 3.2. E-R dijagram baze podataka

3.1.1. Kreiranje baze podataka

Prilikom pokretanja aplikacije prvo što program radi je provjerava postoji li baza podataka sa životopisima i ako ne postoji kreira novu bazu podataka te stvara tablice. Ako postoji baza podataka onda prikuplja podatke iz kreirane baze pomoću naredbe getDataFromDatabase(). Dakle, postupak ako ne postoji baza podataka ide slijedećim redom:

1. Provjeravanje postoji li baza podataka
2. Ako ne postoji, stvara se datoteka koja predstavlja bazu podataka u bin direktoriju
3. Povezivanje s bazom
4. Otvaranje veze
5. Kreiranje tablice

Ovih pet procesa je vidljivo na slici ispod.

```
private void Form1_Load(object sender, EventArgs e)
{

    if (!File.Exists("Zivotopisi.sqlite"))
    {

        SQLiteConnection.CreateFile("Zivotopisi.sqlite");

        my_dbConnection = new SQLiteConnection("Data source=Zivotopisi.sqlite; Version=3;");

        my_dbConnection.Open();

        createTables();
    }
```

Slika 3.3. Provjera postojanja baze podataka

Ako baza ne postoji i ako je potrebno kreirati onda se poseže za dijelom koda koji se poziva s metodom createTables(). To je metoda za stvaranje tablica (entiteta) baze podataka i izvodi se pomoću SQL upita koji se nalaze između navodnika. Sintaksa za izradu tablica je:
CREATE TABLE ime_tablice (atribut1 TIP_PODATKA (*NAPOMENA*), atribut 2 TIP_PODATKA,….)
NAPOMENA svaka tablica mora imati primarni ključ i on mora biti različit za svaki unos. Nakon kreiranja tablica potrebno ih je spojiti. Za spajanje se koristi strani ključ čija sintaksa glasi:
FOREIGN KEY(ime_stranog_ključa) REFERENCES
ime_tablice_s_kojom_se_povezuje(ime_primarnog_ključa). Dio koda s kreiranjem baze podataka će biti vidljiv u prilogu.

Ukoliko pri početnoj provjeri već postoji baza podataka onda se poseže za naredbom getDataFromDatabase() koja preuzima podatke iz baze i postavlja ih u DataGridView, a sintaksa odabira podataka iz baze glasi ovako:

SELECT ime_tablice.atribut_n, ime_tablice.atribut_m FROM ime_tablice WHERE ime_tablice.strani_ključ=ime_tablice_s_kojom_se_povezuje.primarni_ključ ORDER BY ime_tablice.

Ako se zapisi žele poredati po određenom atributu onda se koristi „Order by“ naredba. Nakon odabira podataka postavlja se DataGridView koji ima tri stupca. Prvi je postavljen „ID“ zatim „Ime i prezime“ i na kraju „Datum stvaranja“. Postavljanjem stupaca pokreće se while naredba koja dodaje u svaki stupac pročitani podatak što je vidljivo u slikama 3.4. i 3.5.

```
dgvDatabase.Columns.Clear();
dgvDatabase.ColumnCount = 3;
dgvDatabase.Columns[0].Name = "ID";
dgvDatabase.Columns[1].Name = "Ime i prezime";
dgvDatabase.Columns[2].Name = "Datum stvaranja";

while (reader.Read())
{

    DataGridViewRow row = new DataGridViewRow();
    row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[0].ToString() });
    row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[1].ToString() });
    row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[2].ToString() });

    dgvDatabase.Rows.Add(row);
}
```

Slika 3.4. Postavljanje DataGridView

Postojeći zapisi:

ID	Ime i prezime	Datum stvaranja
1	Davor Aleksic	24.9.2016. 4:12:12
2	Ivan Horvat	24.9.2016. 4:12:53
3	Pero Peric	24.9.2016. 4:13:11

Slika 3.5. DataGridView s učitanim podacima

3.1.2. Čitanje iz baze podataka

Glavni razlog zbog čega je kreirana baza podataka je čitanje iz nje. Ova funkcija nam je korisna u slučaju da želimo promijeniti već spremljeni životopis. Tada klikom na gumb „Otvori“ ispod DataGridView-a možemo otvoriti određeni zapis pod određenim ID-om, imenom i prezimenom te datumom stvaranja. Kada se pokrene događaj klikom gumba „Otvori“ prvo se brišu sva polja koja su bila prije ispunjena u obrascu kako bi se napravilo mjesta za učitani životopis. Zatim se definira varijabla koja predstavlja odabrani životopis te se čitaju podaci iz tablice Osobni_podaci gdje je id jednak odabranom id-u iz DataGridView-a. Nakon postavljanja podataka u polja iz jedne tablice postavljaju se podaci iz druge. U ostalim tablicama je drugačija situacija jer je moguće da one imaju više objekata koji se kreiraju na gumb „Dodaj“. Takav primjer tablice se može vidjeti u tablici „Obrazovanje“ gdje se nalaziti više objekata. Upotrebljava se „while“ petlja koja unosi objekte o obrazovanju u ListBox dok god ih ima. Isti način unosa objekata je korišten za ostale tri klase. Kod za čitanje podataka iz baze će biti vidljiv u prilogu. Ako hoćemo izbrisati zapis iz baze podataka za to nam služi gumb btnDel_Click koji briše podatke iz DataGridView-a¸. Sintaksa za brisanje unosa iz baze podataka glasi:
DELETE FROM ime_tablice WHERE uvjet. Ukoliko je uvjet istinit unos se briše.

3.1.3. Dodavanje podataka u bazu

Kada se klikne gumb „Spremi kao PDF“ osim što se dokument spremi na lokalni disk također se spremaju podaci u kreirane tablice iz baze podataka. Ovaj događaj se izvršava metodom addDataToDatabase() koja funkcionira pomoću try/catch iznimke. Prvi korak je provjera primarnog ključa iz tablice Životopis. Ukoliko već postoje unosi potrebno je pročitati koji je najveći id u tablici jer ne smiju biti dva ista id-a. Zatim se ide na dodavanje podataka u tablice. Sintaksa za dodavanje podataka glasi:
INSERT INTO ime_tablice (atribut_1,atribut_2) VALUES (vrijednost_atributa_1, vrijednost_atributa_2).
Kao i u ostalim slučajevima za tablice Osobni_podaci i Zivotopis unos podataka je klasičan. Za druge tablice taj unos je drugačiji jer postoji mogućnost dodavanja više objekata u te tablice. Zbog toga je potrebno za svaki objekt provjeriti koji je najveći id da se ne bi ponavljali i isto tako svaki put ih uvećati nakon dodavanja novog objekta.

3.2. Globalne varijable i definiranje klasa

Globalne varijable su definirane na samom početku glavne forme, to su varijable koje moraju biti definirane u glavnom tijelu izvornog koda izvan svih funkcija. Globalnim varijablama se može pristupiti bilo gdje u kodu čak i u funkcijama.

```
//globalne varijable
    private string _generatedPDF = null;
    private List<Education> EducationList = new List<Education>();
    private List<Work> WorkList = new List<Work>();
    private List<Language> LangList = new List<Language>();
    private List<Qualification> QualifyList = new List<Qualification>();
    SQLiteConnection my_dbConnection = new SQLiteConnection();
```

Slika 3.6. Globalne varijable

U programu su definirane klase koje određuju podatke i ponašanje programa. Nakon određivanja klasa moguće je stvoriti objekte koji nam omogućuju više zapisa po jednoj kategoriji ispunjavanja životopisa. Definirane klase su: Education, Program, Language, Work, Qualification.

```
class Work
{
    private string employer;
    private string website;
    private string from;
    private string until;
    private string city;
    private string country;
    private string description;
    public Work(string e, string f, string u, string w, string cty, string ctry, string desc)
    {
        employer = e; from = f; until = u; website = w; city = cty; country = ctry; description = desc;
    }
    public string Employer
    {
        get { return this.employer; }
        set { this.employer = value; }
    }
    public string From
    {
        get { return this.from; }
        set { this.from = value; }
    }
    public string Until
    {
        get { return this.until; }
        set { this.until = value; }
    }
    public string Website
    {
        get { return this.website; }
        set { this.website = value; }
    }
```

Slika 3.7. Primjer definiranja klase

3.3. Funkcionalnost aplikacije (način rada)

U ovom poglavlju biti će objašnjen način rada aplikacije te sve funkcije, petlje i metode koje su korištene da bi aplikacija u potpunosti radila. Od načina rada pojedinih gumba i što se događa u programu nakon klika na njih do generiranja PDF dokumenta i povezivanja samog programa s LaTeX-om. U prošlom poglavlju rečeno je da se aplikacija zasniva na događajima koji se pokreću klikom na pojedini gumb u prozoru aplikacije

Prilikom prvog pokretanja aplikacije kreira se baza podataka Zivotopisi.sqlite. Ukoliko je ona već kreirana onda se uzimaju podaci ako ih ima te se postavljaju u polje „Postojeći zapisi“. Sve metode kreiranja, učitavanja, otvaranja podataka opisano je u poglavlju 3.NEŠTO. Nakon kreiranja tablice, uzimanja podataka iz nje i postavljanja tih podataka u polje „Postojeći zapisi“ može se krenuti s ispunjavanjem novog obrasca.

3.3.1. Odabir slike

Prvi segment u popunjavanju životopisa koji se mora ispuniti je odabir slike. Prilikom klika na taj gumb otvara nam se prozor u windowsima za odabir slike. Tijekom postavljanja gumba za odabir slike stvara se OnChooseImageClick metoda.

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

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        txtImagePath.Text = openFileDialog1.FileName;
        try
        {
            File.Copy(txtImagePath.Text, @"..\..\images\image" + new FileInfo(txtImagePath.Text).Extension, true);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);
        }
    }
}
```

Slika 3.8. Dio koda za odabir slike

U kodu iznad je korištena try/catch iznimka zbog toga što je još na početku definirano da program očekuje unos .jpg ili .png formata slika. Unutar try dijela piše se kod koji bi se trebao izvršiti. Ako se taj kod ne izvrši onda se poziva catch dio.

3.3.2. Dodavanje i brisanje objekata u ListBox

Nakon odabiranja slike i ispunjavanja osobnih podataka, slijedeća kategorija za ispunjavanje obrasca je „Obrazovanje“. Slične kategorije su „Jezik“, „Radno iskustvo“ i „Ostale vještine“. Te kategorije povezuje jedan isti segment, a to je ListBox gdje možemo dodati više objekata za pojedinu kategoriju, a poslije dodavanja možemo ih izbrisati ako je potrebno. Također, klikom na već dodani objekt u ListBox ponovno nam se ispisuje onaj tekst koji smo upisali u objektu.

```
private void btnAddEdu_Click(object sender, EventArgs e)
{
    Education edu = new Education(tbDegree.Text, tbFromY.Text, tbTilY.Text,
        tbUniversityName.Text, tbUnivCity.Text, tbUnivCountry.Text, rtbDescription.Text);
    EducationList.Add(edu);
    lbEducations.Items.Clear();
    foreach(Education ed in EducationList)
    {
        lbEducations.Items.Add(ed.Degree);
    }
    foreach (Control tb in gbEducation.Controls)
    {
        if (tb is TextBox)
            tb.Text = "";
    }
    rtbDescription.Text = "";
}
```

3.9. Dodavanje objekta obrazovanje u ListBox

Nakon ispunjavanja kategorije „Obrazovanje“ i klika na gumb „Dodaj“ ispunjeni objekt se spremio u Listbox „lbEducations“ s nazivom kvalifikacije koji smo naveli u polju tb.Degree.Text. Dodavanjem još jednog ili više objekata u ListBox stvara se mogućnost biranja i prepravljanja objekata klikom na pojedini. To nam omogućuje metoda:

`lbEducations_SelectedIndexChanged.`

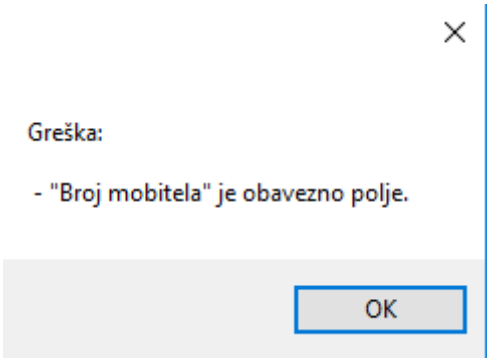
Ako se korisnik odluči izbrisati objekt iz ListBox-a tada se poziva događaj `btnDelEdu_Click` gdje se izabrani objekt briše s ListBox-a. Ovdje se koristi try/catch iznimka s foreach naredbom. Druga solucija za brisanje objekta iz ListBox-a bi mogla biti s Delete tipkom na tipkovnici pomoću događaja Delete_Click. Na isti način su napisane i ostale tri kategorije.

```
private void btnDelEdu_Click(object sender, EventArgs e)
{
    try
    {
        EducationList.RemoveAt(lbEducations.SelectedIndex);
        lbEducations.Items.Clear();
        foreach (Education edu in EducationList)
        {
            lbEducations.Items.Add(edu.Degree);
        }
        foreach (Control c in gbEducation.Controls)
        {
            if (c is TextBox || c is RichTextBox) c.Text = "";
        }
    }
    catch (Exception ex) { MessageBox.Show("Odaberite unos s popisa koji želite obrisati!"); }
}
```

Slika 3.10. Brisanje objekta obrazovanje iz ListBox-a

3.3.3. Poruke upozorenja

Ako je korisnik završio s popunjavanjem obrasca, a pritom zaboravio unijeti ime i prezime, adresu ili broj mobitela/telefona. Klikom na gumb „Spremi kao PDF“ aplikacija neće generirati PDF nego će izbaciti poruku upozorenja koja da jedno od polja nije ispunjeno. Prilikom izvršavanja programa ovaj dio koda se prvi provjerava. Poruke upozorenja nam omogućuju sigurnost da će korisnik u životopisu upisati svoje ime i prezime, adresu i broj mobitela/telefona.



Slika 3.11. Poruka upozorenja

Dio koda koji nam omogućuje izbacivanje poruka upozorenja se sastoji od if/else naredbe gdje koja provjerava je li polje prazno i ako je izbacuje poruku upozorenja, a ako je polje popunjeno program se dalje izvršava.

```
StringBuilder validationMessages = new StringBuilder();
if (String.IsNullOrWhiteSpace(tbName.Text))
{
    validationMessages.Append("\r\n - \"Ime i prezime\" je obavezno polje.");
    valid = false;
}
if (String.IsNullOrWhiteSpace(tbAdress.Text))
{
    validationMessages.Append("\r\n - \"Adresa\" je obavezno polje.");
    valid = false;
}
if (String.IsNullOrWhiteSpace(tbPhoneNumber.Text))
{
    validationMessages.Append("\r\n - \"Broj mobitela\" je obavezno polje.");
    valid = false;
}

if (!valid)
{
    validationMessages.Insert(0, "Greška: \r\n");
    MessageBox.Show(validationMessages.ToString());
}
else
{
```

Slika 3.12. Naredbe za izvođenje poruka upozorenja

3.4. Generiranje PDF dokumenta

Glavni cilj aplikacije je generirati PDF dokument koji će se preko LaTeX programskog jezika obraditi i ponuditi korisniku spremanje na određeno mjesto na lokalnom disku koje odabere. Popunjeni obrazac će biti ispisan u PDF formatu koji je uređen LaTeX alatom za uređivanje. Nakon provjere jesu li sva polja popunjena i ako jesu aplikacija će početi zamjenjivati placeholdere u LaTeX-u i unositi polja upisana u obrascu. Kada odradi taj korak pokreće se run.bat datoteka koja će generirati PDF i ponuditi ga korisniku na spremanje u lokalni disk.

Prvi korak u generiranju PDF dokumenta je pozivanje pdflatex alata preko run.bat datoteke kojom se prosljeđuju parametri upisani u obrascu. Alat pdflatex se poziva iz instalacijske datoteke. Dakle, da bi generiranje uspjelo potrebno je na računalo instalirati MikTeX paket koji sadrži pdflatex alat. Nakon pozivanja pdflatex alata kreće putanja do template.tex datoteke iz koje se generira PDF dokument preko batch datoteke. Sadržaj run.bat datoteke je:
"C:\Program Files (x86)\MiKTeX 2.9\miktex\bin\pdflatex" "..\..\temp\%1". Izlazni direktorij u kojem će se PDF spremiti je "..\..\temp\%1" koji ima suprotnu putanju u odnosu na radni direktorij tj. bin datoteku. Pomoću „..\..\“ se izlazi iz bin datoteke i traži se temp datoteka.

Naredbom:

```
string populatedName = Guid.NewGuid().ToString()
PopulateTemplate(@"..\..\template.tex", populatedName);,
```

uzima se template.tex i popunjava se sadržaj iz unesenih polja. Tako dobivena datoteka se prosljeđuje pdflatex alatu na obradu.

```
Guid.NewGuid().ToString();
```

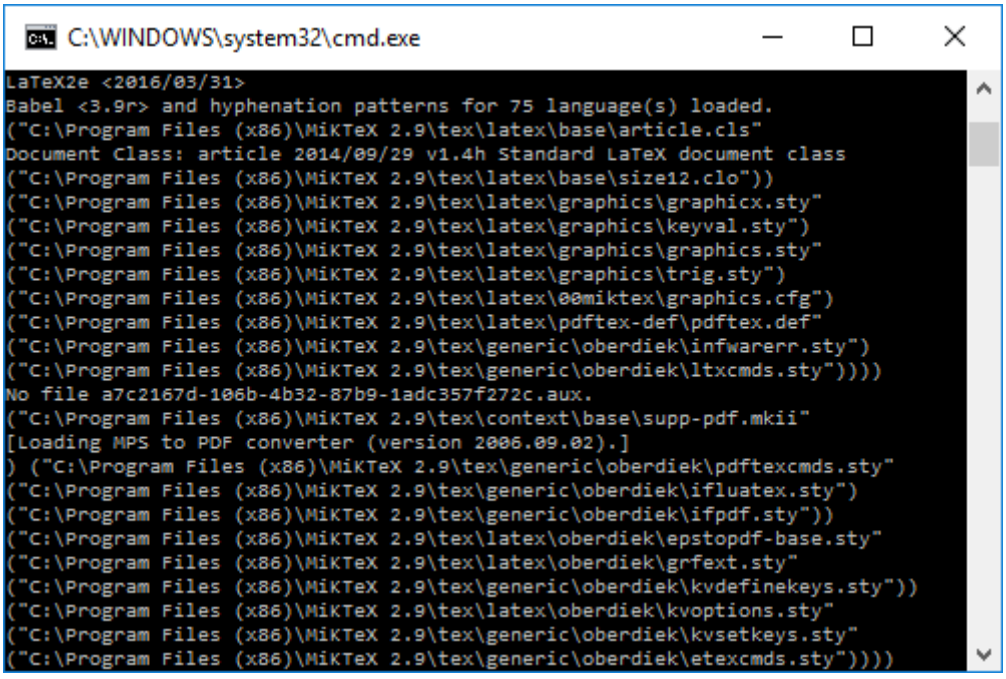
„Globally unique identifier“ naredba kojom se datoteka sigurno generira. Šanse da se dva puta generira ista datoteka na bilo kojem računalu na su nemoguće zato se koristi ta naredba kako bi se izbjegli sukobi u nazivima. Nakon prosljeđivanja template.tex datoteke pdflatex alatu poziva se naredba

```
process.WaitForExit();
```

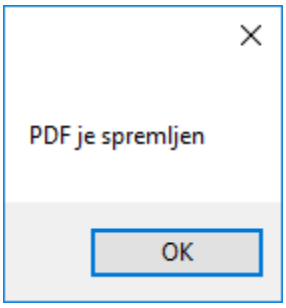
kojom čekamo da pdflatex dovrši svoj posao. Kada se izvrši ta naredba otvara se prozor za spremanje PDF datoteke na lokalni disk.

```
saveFileDialog1.ShowDialog()
```

Nakon spremanja PDF dokumenta brišu se svi privremeni objekti i polja kako bi aplikacija ostala dostupna za slijedeće popunjavanje obrasca. Također, spremljeni podaci u PDF-u se dodaju u bazu podataka gdje će nam biti vidljivi u listi postojećih zapisa.



3.13. Generiranje PDF dokumenta



Slika 3.14. Obavijest na kraju uspješno spremljenog obrasca

3.5. LaTeX uređivanje PDF dokumenta

U ranijim poglavljima se spominjalo da je LaTeX alat za uređivanje teksta. U ovom slučaju LaTeX se koristio za uređivanje PDF dokumenta. Kod uređivanja PDF dokumenta se koristio već napravljeni predložak s interneta. Korišteni predložak se zove Kajbrig CV, a može se naći na stranici:
https://www.overleaf.com/articles/kajbring-cv/bsvjnmqwxrqd#.V-WoW_CLS01[6]
Prepoznavanje template.tex predložka gdje treba upisati određena polja s obrasca se odradilo pomoću placeholdera koji su u upisani u oba programska jezika.

```
private void PopulateTemplate(string templateName, string populatedName)
{
    string education = null, work = null, qualification = null, language = null;
    string content = File.ReadAllText(templateName);

    content = content.Replace("<!Name>", tbName.Text);
    content = content.Replace("<!Adress>", tbAdress.Text);
    content = content.Replace("<!PhoneNumber>", tbPhoneNumber.Text);
    content = content.Replace("<!Email>", tbEmail.Text);
    content = content.Replace("<!Nationality>", tbNationality.Text);
    content = content.Replace("<!DateOfBirth>", dpDateOfBirth.Text);
```

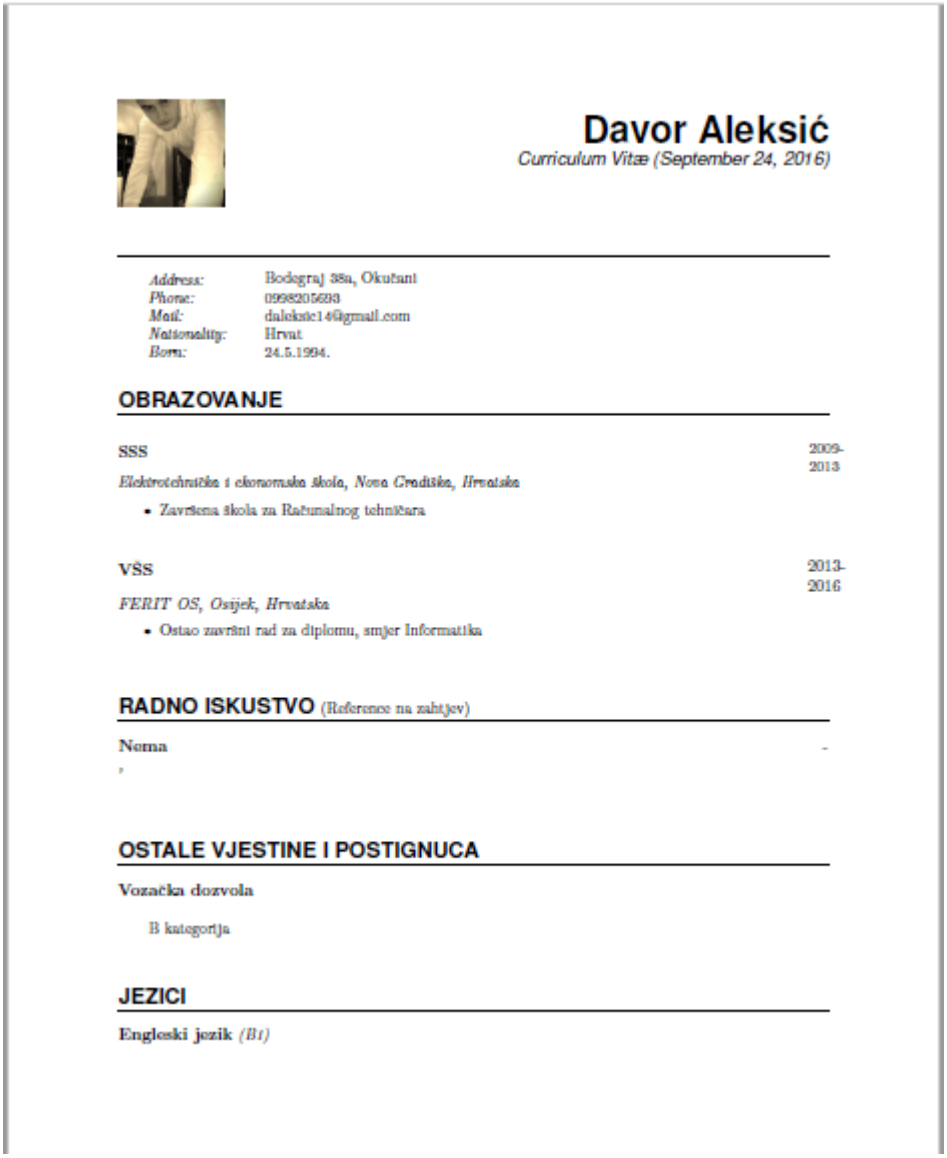
Slika 3.15. Primjer placeholdera u C#

```
%%% Personal details
%%% ----------------------------------------------------------------
\NewPart{}{}

\PersonalEntry{Address:}{<!Adress>}
\PersonalEntry{Phone:}{<!PhoneNumber>}
\PersonalEntry{Mail:}{<!Email>}
\PersonalEntry{Nationality:}{<!Nationality>}
\PersonalEntry{Born:}{<!DateOfBirth>}
```

Slika 3.16. Primjer placeholdera u LaTeX-u

U izgledu predloška na vrhu je postavljena slika te ime i prezime prikazano većim fontom, a ispod datum. Ispod slike, imena i prezimena te datuma postavljeni su ostali podaci. Svaka kategorija je odvojen jednom horizontalnom linijom.



Davor Aleksić

Curriculum Vitæ (September 24, 2016)

Address: Bodegraj 38a, Okučani
Phone: 0998205693
Mail: dalekstic14@gmail.com
Nationality: Hrvat
Born: 24.5.1994.

OBRAZOVANJE

SSS 2009-2013

Elektrotehnička i ekonomska škola, Nova Gradiška, Hrvatska

- Završena škola za Računalnog tehničara

VŠS 2013-2016

FERIT OS, Osijek, Hrvatska

- Ostao završni rad za diplomu, smjer Informatika

RADNO ISKUSTVO (Reference na zahtjev)

Nema -

OSTALE VJESTINE I POSTIGNUCA

Vozačka dozvola

B kategorija

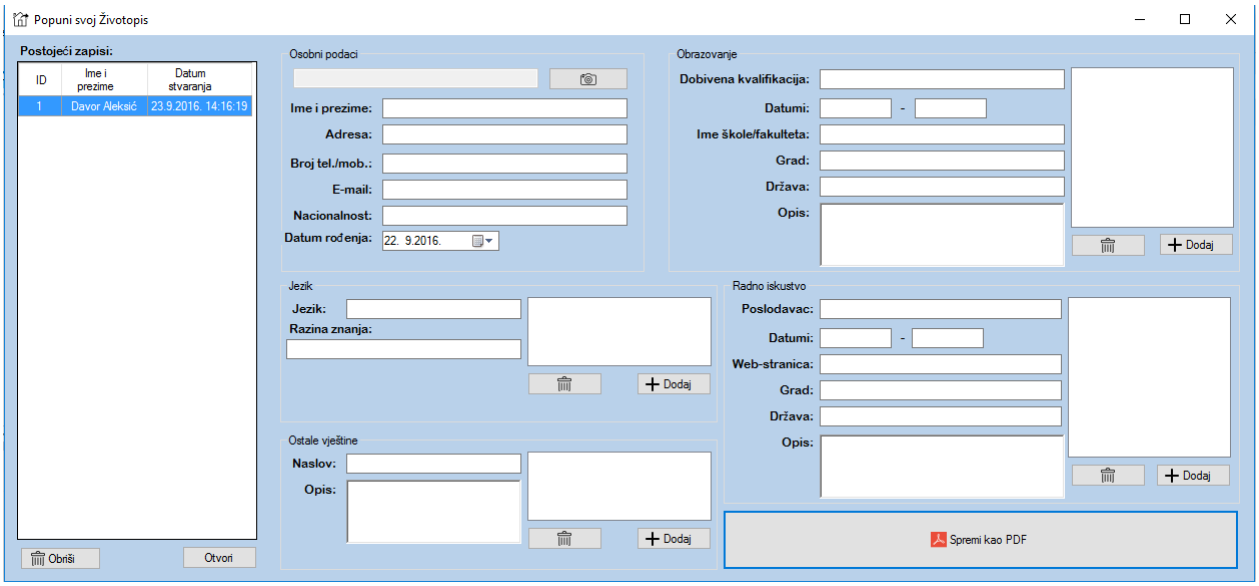
JEZICI

Engleski jezik *(B1)*

Slika 3.17. Izgled predloška

3.6. Dizajn aplikacije

Za dizajn aplikacije se koristila drag&drop metoda u Windows formsu. Pomoću ove metode možemo lako doći do dizajna jer nam Visual Studio sam generira izvorni kod. Pošto je ovo aplikacija za popunjavanje obrazaca ona u svom dizajnu najviše sadržava polja za unos teksta. Dizajn aplikacije se sastoji od šest kategorija koje su redom postavljene kako bi korisniku bilo lakše ispunjavati obrazac svog životopisa, pa su tako za prvu kategoriju na početku aplikacije postavljeni „Osobni podaci“ koji su najvažniji u svakom životopisu. Nakon „Osobnih podataka“ korisnik može preći na popunjavanje „Obrazovanja“ gdje može dodati više obrazovanja preko gumba „Dodaj“. Ostale kategorije su slične kao „Obrazovanje“, a to su: Jezik, Radno iskustvo i Ostale vještine. Još jedan dio izgleda aplikacije je kategorija „Postojeći zapisi“ u kojoj se vide životopisi koji su već ispunjeni.



Slika 3.18. Izgled aplikacije

Na slici je vidljivo da aplikacija ima:

- 21 polje za unos teksta
- jedno polje za odabir datuma rođenja
- jedan gumb za odabir slike
- četiri „Dodaj“ gumba koja služe za dodavanje ispunjenog obrasca u određenoj kategoriji
- četiri gumba za brisanje objekata s liste
- dva gumba za brisanje i otvaranje postojećih životopisa

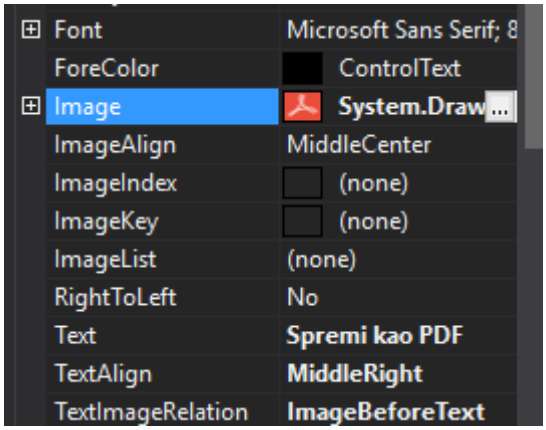
- pet prozora gdje možemo vidjeti koje smo stavke dodali u životopis i polje gdje možemo vidjeti postojeće životopise
- Gumb koji koristimo kada životopis ispunimo do kraja „Spremi kao PDF“

3.6.1. Ikone

U dizajnu aplikacije se pojavljuju ikone koje nam služe umjesto teksta ili zbog uljepšavanja izgleda samog prozora aplikacije. Ikone su pronađene na stranici www.iconfinder.com s koje su preuzete u odgovarajućoj veličini kako bi se uklopile u dizajn aplikacije. Tako su npr. ikone koje se nalaze unutar prozora aplikacije preuzimane u veličini 16x16 piksela, a glavna ikona aplikacije je preuzeta u veličini 48x48 piksela.

Slika 3.19. Ikone koje su prikazane na aplikaciji

Način dodavanja ikona u programskom alatu Visual Studio 2015 je dosta lagan. Samo je potrebno odabrati pojedini element aplikacije kojem želimo pridružiti ikonu. Nakon odabira elementa na desnoj strani prozora Visual Studia postoje postavke elementa kojeg smo odabrali. U tim postavkama između ostalog se nalazi i opcija „image“ za odabiranje slike elementa. Klikom na „image“ odaberemo ikonu u odgovarajućoj veličini (16x16 piksela) te nam se ista pojavi na prozoru aplikacije. Također, možemo birati poziciju ikone u elementu pomoću opcije „TextImageRelation“.



Slika 3.20. Opcije odabiranja ikona

5. ZAKLJUČAK

U ovom završnom radu napravljena je aplikacija za popunjavanje obrazaca pomoću određenih programskih jezika i programskih okruženja. Pri izradi aplikacije koristio se programski jezik C# uz programsko okruženje Visual Studio dok se za obradu PDF dokumenta koristio LaTeX alat za pisanje teksta uz programsko okruženje MiKTeX. Ova aplikacija je jednostavna i brza za korištenje te korisna. U završnom su opisane tehnologije korištene pri izradi stranice i realizacija aplikacije koja je opisana u više poglavlja. Za poboljšanje ove aplikacije bilo bi dobro omogućiti korisniku izbor željenog predloška koji nam definira izgled PDF dokumenta. Dakle, ova aplikacija je uvelike korisna jer prvi korak pri stupanju u radni odnos je ispunjavanje životopisa.

LITERATURA

[1] Knjiga o C# - Tehnike vizuelnog programiranja C#, Zoran Ćirović, Ivan Dunđerski, 2005

[2] Tutorali o C# - http://www.tutorialspoint.com/csharp/

[3] Materijali za C# - Microsoft® Visual C# Step by Step, John Sharp

[4] LaTeX tutoriali - https://www.tug.org/twg/mactex/tutorials/ltxprimer-1.0.pdf

[5] SQLite tutoriali - http://www.tutorialspoint.com/sqlite/

[6] Predložak - https://www.overleaf.com/articles/kajbring-cv/bsvjnmqwxrqd#.V-WoW_CLS01

SAŽETAK

Naslov: Aplikacija za popunjavanje obrazaca

U ovom završnom radu je napravljena aplikacija za popunjavanje obrazaca pomoću C# programskog jezika i Latex alata za uređivanje teksta. Popunjavanjem obrasca aplikacija generira PDF dokument i sprema ga na lokalni disk. Obrazac se sastoji od polja za unos teksta te gumba koji imaju određene funkcije. Svrha aplikacije je pomoći korisniku pri unosu životopisa.

Ključne riječi: Desktop aplikacija, PDF dokument, C# programiranje, obrazac, slika

ABSTRACT

Title: Application for filling forms

Desktop application that fill in forms was successfuly created using C# programming language and LaTeX tool for text editing. Filling out the forms application generates PDF document and saves it on local disc. The form consists of boxes for text input and buttons which have certain funcktions. The purpose of application is to assist the to fill his CV.

Key words: Desktop application, PDF document, C# programming, form, image

ŽIVOTOPIS

Davor Aleksić rođen je 24.05.1994. g. u Wertheimu, Njemačka. Nakon završene osnovne škole Okučani u Okučanima, upisuje Elektrotehničku i Ekonomsku školu, Nova Gradiška te maturira 2013. g. Iste te godine upisuje stručni studij na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija. Tijekom srednjoškolskog obrazovanja osvojio je drugo mjesto u natjecanju iz Osnova elektrotehnike. U trajanju studiranja pohađao je stručnu praksu u sklopu Fakulteta. Pred obranom je završnog rada.

PRILOG A. Izvorni kod aplikacije

```
public partial class Form1 : Form
      {

    //globalne varijable
              private string _generatedPDF = null;
        private List<Education> EducationList = new List<Education>();
        private List<Work> WorkList = new List<Work>();
        private List<Language> LangList = new List<Language>();
        private List<Qualification> QualifyList = new List<Qualification>();
        SQLiteConnection my_dbConnection = new SQLiteConnection();

              public Form1()
              {

                    Application.CurrentCulture =
System.Globalization.CultureInfo.CreateSpecificCulture("hr-HR");
                    System.Threading.Thread.CurrentThread.CurrentCulture =
System.Globalization.CultureInfo.CreateSpecificCulture("hr-HR");
                    System.Threading.Thread.CurrentThread.CurrentUICulture =
System.Globalization.CultureInfo.CreateSpecificCulture("hr-HR");




                    InitializeComponent();

                    // Naslov.
                    Text = "Popuni svoj Životopis";

                    dpDateOfBirth.Format = DateTimePickerFormat.Custom;
                    dpDateOfBirth.CustomFormat =
System.Threading.Thread.CurrentThread.CurrentUICulture.DateTimeFormat.ShortDatePattern;


                    openFileDialog1.Filter = "JPG (.jpg)|*.jpg|PNG (.png)|*.png";
                    openFileDialog1.FilterIndex = 1;
                    openFileDialog1.CheckFileExists = true;
                    openFileDialog1.Title = "Odaberite sliku";

                    saveFileDialog1.Filter = "PDF (.pdf)|*.pdf";
                    saveFileDialog1.FilterIndex = 1;
                    saveFileDialog1.Title = "Odaberite lokaciju za spremanje obrasca";
              }


        private void Form1_Load(object sender, EventArgs e)
        {

            if (!File.Exists("Zivotopisi.sqlite"))
            {

                SQLiteConnection.CreateFile("Zivotopisi.sqlite");

                my_dbConnection = new SQLiteConnection("Data source=Zivotopisi.sqlite;
Version=3;");

                my_dbConnection.Open();

                createTables();
            }
```

```
            else {
                my_dbConnection = new SQLiteConnection("Data source=Zivotopisi.sqlite;
Version=3;");
                my_dbConnection.Open();

                getDataFromDatabase(); }
        }



        private void createTables()
        {
            try
            {

                SQLiteCommand command = new SQLiteCommand("CREATE TABLE Zivotopis (id INT
PRIMARY KEY, datum_stvaranja DATETIME);", my_dbConnection);

                command.ExecuteNonQuery();
                command = new SQLiteCommand("CREATE TABLE Osobni_podaci(id INT PRIMARY
KEY, ime_prezime VARCHAR(50), adresa VARCHAR(50), broj_tel VARCHAR(15), e_mail
VARCHAR(50), nacionalnost VARCHAR(15), dat_rodenja VARCHAR(10), slika VARCHAR(100),
id_zivotopisa INT, FOREIGN KEY (id_zivotopisa) REFERENCES Zivotopis(id));",
my_dbConnection);
                command.ExecuteNonQuery();
                command = new SQLiteCommand("CREATE TABLE Obrazovanje(id INT PRIMARY KEY,
kvalifikacija VARCHAR(50), datum_od VARCHAR(10), datum_do VARCHAR(10), ime_fak
VARCHAR(50), grad VARCHAR(20), drzava VARCHAR(20), opis TEXT, id_op INT, FOREIGN KEY
(id_op) REFERENCES Osobni_podaci(id));", my_dbConnection);
                command.ExecuteNonQuery();
                command = new SQLiteCommand("CREATE TABLE Radno_iskustvo(id INT PRIMARY
KEY, poslodavac VARCHAR(50), datum_od VARCHAR(10), datum_do VARCHAR(10), web_str
VARCHAR(50), grad VARCHAR(20), drzava VARCHAR(20), opis TEXT, id_op INT, FOREIGN KEY
(id_op) REFERENCES Osobni_podaci(id));", my_dbConnection);
                command.ExecuteNonQuery();
                command = new SQLiteCommand("CREATE TABLE Jezik(id INT PRIMARY KEY, jezik
VARCHAR(15), razina_znanja VARCHAR(50), id_op INT, FOREIGN KEY (id_op) REFERENCES
Osobni_podaci(id));", my_dbConnection);
                command.ExecuteNonQuery();
                command = new SQLiteCommand("CREATE TABLE Ostale_vjestine(id INT PRIMARY
KEY, naslov VARCHAR(50), opis TEXT, id_op INT, FOREIGN KEY (id_op) REFERENCES
Osobni_podaci(id));", my_dbConnection);
                command.ExecuteNonQuery();

            }
            catch (Exception ex) {  }
        }



        private void getDataFromDatabase()
        {
            string sql = "SELECT Osobni_podaci.id, Osobni_podaci.ime_prezime,
Zivotopis.datum_stvaranja FROM Zivotopis,Osobni_podaci WHERE
Osobni_podaci.id_zivotopisa=Zivotopis.id ORDER BY datum_stvaranja";
            SQLiteCommand command = new SQLiteCommand(sql, my_dbConnection);
            SQLiteDataReader reader = command.ExecuteReader();

            dgvDatabase.Columns.Clear();
            dgvDatabase.ColumnCount = 3;
            dgvDatabase.Columns[0].Name = "ID";
```

```
            dgvDatabase.Columns[1].Name = "Ime i prezime";
            dgvDatabase.Columns[2].Name = "Datum stvaranja";

            while (reader.Read())
            {

                DataGridViewRow row = new DataGridViewRow();
                row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[0].ToString()
});
                row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[1].ToString()
});
                row.Cells.Add(new DataGridViewTextBoxCell { Value = reader[2].ToString()
});


                dgvDatabase.Rows.Add(row);
            }
        }

        private void btnOpen_Click(object sender, EventArgs e)
        {

            clearEverything();


            string id = dgvDatabase.SelectedRows[0].Cells[0].Value.ToString();

            string sql = "SELECT * FROM Osobni_podaci WHERE id=" + id + ";";
            SQLiteCommand command = new SQLiteCommand(sql, my_dbConnection);
            SQLiteDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {

                tbName.Text = reader["ime_prezime"].ToString();
                tbAdress.Text = reader["adresa"].ToString();
                tbPhoneNumber.Text = reader["broj_tel"].ToString();
                tbEmail.Text = reader["e_mail"].ToString();
                tbNationality.Text = reader["nacionalnost"].ToString();
                dpDateOfBirth.Text = reader["dat_rodenja"].ToString();
                txtImagePath.Text = reader["slika"].ToString();
            }

            sql="SELECT * FROM Obrazovanje WHERE id_op="+id+";";
            command = new SQLiteCommand(sql, my_dbConnection);
            Education edu;
            reader = command.ExecuteReader();
            while (reader.Read())
            {

                edu = new Education(reader["kvalifikacija"].ToString(),
reader["datum_od"].ToString(), reader["datum_do"].ToString(),
reader["ime_fak"].ToString(), reader["grad"].ToString(), reader["drzava"].ToString(),
reader["opis"].ToString());

                EducationList.Add(edu);
            }

            foreach (Education ed in EducationList)
            {
                lbEducations.Items.Add(ed.Name);
            }
            sql = "SELECT * FROM Radno_iskustvo WHERE id_op = " + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
```

```
            Work w;
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                w = new Work(reader["poslodavac"].ToString(),
reader["datum_od"].ToString(), reader["datum_do"].ToString(),
reader["web_str"].ToString(), reader["grad"].ToString(), reader["drzava"].ToString(),
reader["opis"].ToString());
                WorkList.Add(w);
            }
            foreach (Work work in WorkList)
            {
                lbWorkExpes.Items.Add(work.Employer);
            }
            sql = "SELECT * FROM Jezik WHERE id_op=" + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
            Language lang;
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                lang = new Language(reader["jezik"].ToString(),
reader["razina_znanja"].ToString());
                LangList.Add(lang);
            }
            foreach (Language l in LangList)
            {
                lbLanguages.Items.Add(l.Lang);
            }
            sql = "SELECT * FROM Ostale_vjestine WHERE id_op =" + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
            Qualification q;
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                q = new Qualification(reader["naslov"].ToString(),
reader["opis"].ToString());
                QualifyList.Add(q);
            }
            foreach(Qualification qual in QualifyList)
            {
                lbQualifications.Items.Add(qual.Qualify);
            }
        }

        /// <summary>
        /// Odabir slike
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void OnChooseImageClick(object sender, EventArgs e)
             {

                    if (openFileDialog1.ShowDialog() == DialogResult.OK)
                    {
                           txtImagePath.Text = openFileDialog1.FileName;
                           try
                           {
                                  File.Copy(txtImagePath.Text, @"..\..\images\image" +
new FileInfo(txtImagePath.Text).Extension, true);
                           }
                           catch (Exception ex)
                           {
```

```
                                MessageBox.Show("Error: Could not read file from disk.
Original error: " + ex.Message);
                            }
                        }
                    }

        private void OnGeneratePDFClick(object sender, EventArgs e)
        {
            _generatedPDF = null;

            /// Izvršavanje programa
            // 1.
            bool valid = true;
            StringBuilder validationMessages = new StringBuilder();
            if (String.IsNullOrWhiteSpace(tbName.Text))
            {
                validationMessages.Append("\r\n - \"Ime i prezime\" je obavezno polje.");
                valid = false;
            }
            if (String.IsNullOrWhiteSpace(tbAdress.Text))
            {
                validationMessages.Append("\r\n - \"Adresa\" je obavezno polje.");
                valid = false;
            }
            if (String.IsNullOrWhiteSpace(tbPhoneNumber.Text))
            {
                validationMessages.Append("\r\n - \"Broj mobitela\" je obavezno polje.");
                valid = false;
            }

            if (!valid)
            {
                validationMessages.Insert(0, "Greška: \r\n");
                MessageBox.Show(validationMessages.ToString());
            }
            else
            {
                // 2.
                string populatedName = Guid.NewGuid().ToString();
            PopulateTemplate(@"..\..\template.tex", populatedName);

            // 3.
            Process process = new Process { StartInfo = { Arguments = populatedName +
".tex" } };

            //Pokretannje batch-a.
            process.StartInfo.FileName = @"..\..\run.bat";

            if (process.Start())
            {
                process.WaitForExit();

                // 4. Download PDF-a
                _generatedPDF = @"..\..\temp\" + populatedName + ".pdf";
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    string name = saveFileDialog1.FileName;
                    try
                    {
                        File.Copy(_generatedPDF, name, true);
                        MessageBox.Show("PDF je spremljen");
                    }
```

```
                    catch (Exception ex)
                    {
                        MessageBox.Show("Dogodila se greška prilikom spremanja PDF
dokumenta.");
                    }
                }

                // 5. Brišemo sve privremene fajlove.
                ClearDirectory(new DirectoryInfo(@"..\..\temp"));
                addDataToDatabase();
                clearEverything();
            }
            else
            {
                MessageBox.Show("Generiranje nije uspjelo.");
            }
                //Nakon kreiranja PDF-a podaci se dodaju u bazu podataka
            }
        }
        //metoda za dodavanje podataka u bazu podataka
        private void addDataToDatabase()
        {
            try {

                int id = 0;

                string sql = "SELECT id FROM Osobni_podaci ORDER BY id DESC;";
                SQLiteCommand command = new SQLiteCommand(sql, my_dbConnection);
                SQLiteDataReader reader = command.ExecuteReader();

                if (reader.Read()) id = Convert.ToInt32(reader["id"]);


                sql = "INSERT INTO Zivotopis (id,datum_stvaranja) VALUES(" + (id +
1).ToString() + ", DATETIME('NOW'));";
                command = new SQLiteCommand(sql, my_dbConnection);
                command.ExecuteNonQuery();
                sql = "INSERT INTO Osobni_podaci
(id,ime_prezime,adresa,broj_tel,e_mail,nacionalnost,dat_rodenja,slika,id_zivotopisa)
VALUES (" + (id + 1).ToString() + ",'" + tbName.Text + "','" + tbAdress.Text + "','" +
tbPhoneNumber.Text + "','" + tbEmail.Text + "','" + tbNationality.Text + "','" +
dpDateOfBirth.Text + "','" + txtImagePath.Text + "'," + (id + 1).ToString() + ");";
                command = new SQLiteCommand(sql, my_dbConnection);
                command.ExecuteNonQuery();


                int idEdu = 0;
                sql = "SELECT id FROM Obrazovanje ORDER BY id DESC;";
                command = new SQLiteCommand(sql, my_dbConnection);
                reader = command.ExecuteReader();
                if (reader.Read()) idEdu = Convert.ToInt32(reader["id"]);
                foreach (Education e in EducationList)
                {
                    sql = "INSERT INTO Obrazovanje
(id,kvalifikacija,datum_od,datum_do,ime_fak,grad,drzava,opis,id_op) VALUES (" +
(++idEdu).ToString() + ",'" + e.Degree + "','" + e.From + "','" + e.Until + "','" +
e.Name + "','" + e.City + "','" + e.Country + "','" + e.Description + "'," + (id +
1).ToString() + ");";
                    command = new SQLiteCommand(sql, my_dbConnection);
                    command.ExecuteNonQuery();
                }
                int idWork = 0;
```

```
                sql = "SELECT id FROM Radno_iskustvo ORDER BY id DESC;";
                command = new SQLiteCommand(sql, my_dbConnection);
                reader = command.ExecuteReader();
                if (reader.Read()) idWork = Convert.ToInt32(reader["id"]);
                foreach (Work w in WorkList)
                {
                    sql = "INSERT INTO Radno_iskustvo
(id,poslodavac,datum_od,datum_do,web_str,grad,drzava,opis,id_op) VALUES (" +
(++idWork).ToString() + ",'" + w.Employer + "','" + w.From + "','" + w.Until + "','" +
w.Website + "','" + w.City + "','" + w.Country + "','" + w.Description + "'," + (id +
1).ToString() + ");";
                    command = new SQLiteCommand(sql, my_dbConnection);
                    command.ExecuteNonQuery();
                }
                int idLang = 0;
                sql = "SELECT id FROM Jezik ORDER BY id DESC;";
                command = new SQLiteCommand(sql, my_dbConnection);
                reader = command.ExecuteReader();
                if (reader.Read()) idLang = Convert.ToInt32(reader["id"]);
                foreach (Language l in LangList)
                {
                    sql = "INSERT INTO Jezik (id,jezik,razina_znanja,id_op) VALUES (" +
(++idLang).ToString() + ",'" + l.Lang + "','" + l.Level + "'," + (id + 1).ToString() +
");";
                    command = new SQLiteCommand(sql, my_dbConnection);
                    command.ExecuteNonQuery();
                }
                int idQualify = 0;
                sql = "SELECT id FROM Ostale_vjestine ORDER BY id DESC;";
                command = new SQLiteCommand(sql, my_dbConnection);
                reader = command.ExecuteReader();
                if (reader.Read()) idQualify = Convert.ToInt32(reader["id"]);
                foreach (Qualification q in QualifyList)
                {
                    sql = "INSERT INTO Ostale_vjestine (id,naslov,opis,id_op) VALUES (" +
(++idQualify).ToString() + ",'" + q.Qualify + "','" + q.Description + "'," + (id +
1).ToString() + ");";
                    command = new SQLiteCommand(sql, my_dbConnection);
                    command.ExecuteNonQuery();
                }
            }
            catch (Exception ex) { }

            getDataFromDatabase();
        }

        /// <summary>
        /// Zamjenjuje placeholdere u template-u unosom sa forme.
        /// </summary>
        /// <param name="templateName"></param>
        /// <param name="populatedName"></param>

        private void PopulateTemplate(string templateName, string populatedName)
        {
            string education = null, work = null, qualification = null, language = null;
            string content = File.ReadAllText(templateName);

            content = content.Replace("<!Name>", tbName.Text);
            content = content.Replace("<!Adress>", tbAdress.Text);
            content = content.Replace("<!PhoneNumber>", tbPhoneNumber.Text);
            content = content.Replace("<!Email>", tbEmail.Text);
            content = content.Replace("<!Nationality>", tbNationality.Text);
```

```
            content = content.Replace("<!DateOfBirth>", dpDateOfBirth.Text);



            foreach (Education e in EducationList)
            {

                education = education+ "\n\\EducationEntry{"+e.Degree+"}{"+e.From+"-
"+e.Until+"}{"+e.Name+ ", "+e.City+", "+e.Country+"} {\\begin{itemize}
\\item{"+e.Description+"} \\end{itemize}} \\sepspace";
            }


            content = content.Replace("<!Education>", education);

            foreach (Work w in WorkList)
            {
                work = work + "\n\\WorkEntry{"+w.Employer+"}{\\href{"+w.Website+"}
}{"+w.From+" -"+w.Until+"}{"+w.City+", "+w.Country+"}{"+w.Description+"}\\sepspace";
            }
            content = content.Replace("<!Work>", work);
            foreach (Qualification q in QualifyList)
            {
                qualification = qualification +
"\n\\WorkEntry{"+q.Qualify+"}{{}}{}{}{"+q.Description+"}\\sepspace";
            }
            content = content.Replace("<!Qualification>", qualification);
            foreach (Language l in LangList)
            {
                language = language + "\n \\Language{"+l.Lang+"}{("+l.Level+")}";
            }
            content = content.Replace("<!Language>", language);

            //Upisivanje podataka u template
            using (StreamWriter sw = File.CreateText(@"..\..\temp\" + populatedName +
".tex"))
                    {
                            foreach (char ch in content)
                            {
                                    sw.Write(ch);
                            }
                    }
             }


              /// <param name="directory"></param>
              private void ClearDirectory(DirectoryInfo directory)
              {
                    try
                    {
                            foreach (FileInfo file in directory.GetFiles())
                            {
                                    file.Delete();
                            }
                            foreach (DirectoryInfo dir in directory.GetDirectories())
                            {
                                    dir.Delete(true);
                            }
                    }
                    catch (Exception ex)
                    {
```

```
                }
            }
        //Dodavanje objekta obrazovanje u listBox nakon pritiska na gumb dodaj
        //Dodavanje u listbox
        private void btnAddEdu_Click(object sender, EventArgs e)
        {
            Education edu = new Education(tbDegree.Text, tbFromY.Text, tbTilY.Text,
tbUniversityName.Text, tbUnivCity.Text, tbUnivCountry.Text, rtbDescription.Text);
            EducationList.Add(edu);
            lbEducations.Items.Clear();
            foreach(Education ed in EducationList)
            {
                lbEducations.Items.Add(ed.Degree);
            }
            foreach (Control tb in gbEducation.Controls)
            {
                if (tb is TextBox)
                    tb.Text = "";
            }
            rtbDescription.Text = "";
        }

        private void btnWAdd_Click(object sender, EventArgs e)
        {
            Work w = new Work(tbEmployer.Text, tbWFromY.Text, tbWTilY.Text,
tbEmplyerWebSite.Text, tbEmployerCity.Text, tbEmployerCountry.Text,
rtbWDescription.Text);
            WorkList.Add(w);
            lbWorkExpes.Items.Clear();
            foreach(Work work in WorkList)
            {
                lbWorkExpes.Items.Add(work.Employer);
            }
            foreach (Control tb in gbWorkExp.Controls)
            {
                if (tb is TextBox)
                    tb.Text = "";
            }
            rtbWDescription.Text = "";
        }

        private void btnAddLang_Click(object sender, EventArgs e)
        {
            Language l = new Language(tbLanguage.Text, tbLevelOfKnowledge.Text);
            LangList.Add(l);
            lbLanguages.Items.Clear();
            foreach(Language lang in LangList)
            {
                lbLanguages.Items.Add(lang.Lang);
            }
            foreach (Control tb in gbLanguage.Controls)
            {
                if (tb is TextBox)
                    tb.Text = "";
            }
        }

        private void btnAddQ_Click(object sender, EventArgs e)
        {
                Qualification q = new Qualification(tbOtherQualification.Text,
rtbQDescription.Text);
            QualifyList.Add(q);
```

```
            lbQualifications.Items.Clear();
            foreach(Qualification qualify in QualifyList)
            {
                lbQualifications.Items.Add(qualify.Qualify);
            }
            foreach (Control tb in gbQualification.Controls)
            {
                if (tb is TextBox)
                    tb.Text = "";
            }
            rtbQDescription.Text = "";
        }
        private void btnDelEdu_Click(object sender, EventArgs e)
        {
            try
            {
                EducationList.RemoveAt(lbEducations.SelectedIndex);
                lbEducations.Items.Clear();
                foreach (Education edu in EducationList)
                {
                    lbEducations.Items.Add(edu.Degree);
                }
                foreach (Control c in gbEducation.Controls)
                {
                    if (c is TextBox || c is RichTextBox) c.Text = "";
                }
            }
            catch (Exception ex) { MessageBox.Show("Odaberite unos s popisa koji želite
obrisati!"); }
        }

        private void btnDelWork_Click(object sender, EventArgs e)
        {
            try
            {
                WorkList.RemoveAt(lbWorkExpes.SelectedIndex);
                lbWorkExpes.Items.Clear();
                foreach (Work w in WorkList)
                {
                    lbWorkExpes.Items.Add(w.Employer);
                }
                foreach (Control c in gbWorkExp.Controls)
                {
                    if (c is TextBox || c is RichTextBox) c.Text = "";
                }
            }
            catch (Exception ex) { MessageBox.Show("Odaberite unos s popisa koji želite
obrisati!"); }
        }

        private void btnDelLang_Click(object sender, EventArgs e)
        {
            try
            {
                LangList.RemoveAt(lbLanguages.SelectedIndex);
                lbLanguages.Items.Clear();
                foreach (Language l in LangList)
                {
                    lbLanguages.Items.Add(l.Lang);
                }
                foreach (Control c in gbLanguage.Controls)
                {
```

```
                    if (c is TextBox || c is RichTextBox) c.Text = "";
                }
            }
            catch (Exception ex) { MessageBox.Show("Odaberite unos s popisa koji želite
obrisati!"); }
        }

        private void btnDelQ_Click(object sender, EventArgs e)
        {
            try
            {
                QualifyList.RemoveAt(lbQualifications.SelectedIndex);
                lbQualifications.Items.Clear();
                foreach (Qualification q in QualifyList)
                {
                    lbQualifications.Items.Add(q.Qualify);
                }
                foreach (Control c in gbQualification.Controls)
                {
                    if (c is TextBox || c is RichTextBox) c.Text = "";
                }
            }
            catch (Exception ex) { MessageBox.Show("Odaberite unos s popisa koji želite
obrisati!"); }
        }

        //ispisivanje u textbox kada odaberemo neki objekt iz listbox
        private void lbEducations_SelectedIndexChanged(object sender, EventArgs e)
        {
            tbDegree.Text = EducationList[lbEducations.SelectedIndex].Degree;
            tbFromY.Text = EducationList[lbEducations.SelectedIndex].From;
            tbTilY.Text = EducationList[lbEducations.SelectedIndex].Until;
            tbUnivCity.Text = EducationList[lbEducations.SelectedIndex].City;
            tbUnivCountry.Text = EducationList[lbEducations.SelectedIndex].Country;
            tbUniversityName.Text = EducationList[lbEducations.SelectedIndex].Name;
            rtbDescription.Text = EducationList[lbEducations.SelectedIndex].Description;
        }

        private void lbLanguages_SelectedIndexChanged(object sender, EventArgs e)
        {
            tbLanguage.Text = LangList[lbLanguages.SelectedIndex].Lang;
            tbLevelOfKnowledge.Text = LangList[lbLanguages.SelectedIndex].Level;
        }

        private void lbWorkExpes_SelectedIndexChanged(object sender, EventArgs e)
        {
            tbEmployer.Text = WorkList[lbWorkExpes.SelectedIndex].Employer;
            tbWFromY.Text = WorkList[lbWorkExpes.SelectedIndex].From;
            tbWTilY.Text = WorkList[lbWorkExpes.SelectedIndex].Until;
            tbEmplyerWebSite.Text = WorkList[lbWorkExpes.SelectedIndex].Website;
            tbEmployerCity.Text = WorkList[lbWorkExpes.SelectedIndex].City;
            tbEmployerCountry.Text = WorkList[lbWorkExpes.SelectedIndex].Country;
            rtbWDescription.Text = WorkList[lbWorkExpes.SelectedIndex].Description;
        }

        private void lbQualifications_SelectedIndexChanged(object sender, EventArgs e)
        {
            tbOtherQualification.Text =
QualifyList[lbQualifications.SelectedIndex].Qualify;
            rtbQDescription.Text =
QualifyList[lbQualifications.SelectedIndex].Description;
        }
```

```
//nakon Spremi kao PDF briše sve textboxove i liste osim baze
private void clearEverything()
{
    foreach (Control c in gbPersonal.Controls)
    {
        if (c is TextBox) c.Text = "";
        if (c is RichTextBox) c.Text = "";
        if (c is DateTimePicker) c.Text = "";
    }
    foreach (Control c in gbQualification.Controls)
    {
        if (c is TextBox) c.Text = "";
        if (c is RichTextBox) c.Text = "";
        if (c is DateTimePicker) c.Text = "";
    }
    foreach (Control c in gbWorkExp.Controls)
    {
        if (c is TextBox) c.Text = "";
        if (c is RichTextBox) c.Text = "";
        if (c is DateTimePicker) c.Text = "";
    }
    foreach (Control c in gbLanguage.Controls)
    {
        if (c is TextBox) c.Text = "";
        if (c is RichTextBox) c.Text = "";
        if (c is DateTimePicker) c.Text = "";
    }
    foreach (Control c in gbEducation.Controls)
    {
        if (c is TextBox) c.Text = "";
        if (c is RichTextBox) c.Text = "";
        if (c is DateTimePicker) c.Text = "";
    }
    lbEducations.Items.Clear();
    lbLanguages.Items.Clear();
    lbQualifications.Items.Clear();
    lbWorkExpes.Items.Clear();
    QualifyList.Clear();
    EducationList.Clear();
    LangList.Clear();
    WorkList.Clear();
}


private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    my_dbConnection.Close();
}


private void btnDel_Click(object sender, EventArgs e)
{
    string id = dgvDatabase.SelectedRows[0].Cells[0].Value.ToString();
    string sql = "DELETE FROM Osobni_podaci WHERE id=" + id + ";";
    SQLiteCommand command = new SQLiteCommand(sql, my_dbConnection);
    command.ExecuteNonQuery();
    sql = "DELETE FROM Zivotopis WHERE id=" + id + ";";
    command = new SQLiteCommand(sql, my_dbConnection);
    command.ExecuteNonQuery();
    sql = "DELETE FROM Radno_iskustvo WHERE id_op=" + id + ";";
    command = new SQLiteCommand(sql, my_dbConnection);
```

```
            command.ExecuteNonQuery();
            sql = "DELETE FROM Jezik WHERE id_op=" + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
            command.ExecuteNonQuery();
            sql = "DELETE FROM Obrazovanje WHERE id_op=" + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
            command.ExecuteNonQuery();
            sql = "DELETE FROM Ostale_vjestine WHERE id_op=" + id + ";";
            command = new SQLiteCommand(sql, my_dbConnection);
            command.ExecuteNonQuery();
            getDataFromDatabase();
            clearEverything();
        }

        private void rtbWDescription_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

PRILOG B. Izvorni kod LaTeX predloška

```
\documentclass[paper=letter,fontsize=11pt]{scrartcl} % KOMA-article class
\usepackage[english]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[protrusion=true,expansion=true]{microtype}
\usepackage{amsmath,amsfonts,amsthm} % Math packages
\usepackage{graphicx} % Enable pdflatex
\graphicspath{ {../images/} }
\DeclareGraphicsExtensions{.pdf,.jpeg,.png,.jpg}
\usepackage[svgnames]{xcolor} % Colors by their 'svgnames'
\usepackage{geometry}
%\textheight=700px % Saving trees ;-)
%\usepackage{url}
\usepackage[colorlinks=true,
linkcolor=blue,
urlcolor=blue]{hyperref}
\usepackage{float}
\usepackage{etaremune}
\usepackage{wrapfig}
\usepackage{attachfile}


\frenchspacing % Better looking spacings after periods
\pagestyle{empty} % No pagenumbers/headers/footers

%\addtolength{\voffset}{-40pt}
%\addtolength{\textheight}{20pt}

\setlength\topmargin{0pt}
\addtolength\topmargin{-\headheight}
\addtolength\topmargin{-\headsep}
\setlength\oddsidemargin{0pt}
\setlength\textwidth{\paperwidth}
\addtolength\textwidth{-2in}
\setlength\textheight{\paperheight}
%\addtolength\textheight{-3in}
\addtolength\textheight{-2in}
\usepackage{layout}

%%% Custom sectioning}{sectsty package)
%%% ----------------------------------------------------------
\usepackage{sectsty}

\sectionfont{% % Change font of \section command
\usefont{OT1}{phv}{b}{n}% % bch-b-n: CharterBT-Bold font
\sectionrule{0pt}{0pt}{-5pt}{1pt}}

%%% Macros
%%% ----------------------------------------------------------
```

```
\newlength{\spacebox}
\settowidth{\spacebox}{8888888888} % Box to align text
\newcommand{\sepspace}{\vspace*{1em}} % Vertical space macro

\newcommand{\MyName}[1]{ % Name
\Huge \usefont{OT1}{phv}{b}{n} \hfill #1
\par \normalsize \normalfont}
\newcommand{\MySlogan}[1]{ % Slogan}{optional)
\large \usefont{OT1}{phv}{m}{n}\hfill \textit{#1}
\par \normalsize \normalfont}

\newcommand{\NewPart}[2]{\section*{\uppercase{#1} \small \normalfont #2}}

\newcommand{\NewParttwo}[1]{
\noindent \huge \textbf{#1}
\normalsize \par}



\newcommand{\PersonalEntry}[2]{\small
\noindent\hangindent=2em\hangafter=0 % Indentation
\parbox{\spacebox}{ % Box to align text
\textit{#1}} % Entry name}{birth, address, etc.)
\small\hspace{1.5em} #2 \par} % Entry value

\newcommand{\SkillsEntry}[2]{ % Same as \PersonalEntry
\noindent\hangindent=2em\hangafter=0 % Indentation
\parbox{\spacebox}{ % Box to align text
\textit{#1}} % Entry name}{birth, address, etc.)
\hspace{1.5em} #2 \par} % Entry value
\newcommand{\EducationEntry}[4]{
\noindent \textbf{#1} \hfill % Study
\colorbox{White}{%
\parbox{1em}{%
\hfill\color{Black}#2}} \par % Duration
\noindent \textit{#3} \par % School
\noindent\hangindent=2em\hangafter=0 \small #4 % Description
\normalsize \par}

\newcommand{\WorkEntry}[5]{
\noindent \textbf{#1}
\noindent \small \textit{#2}
\hfill % Study
\colorbox{White}{%
\parbox{6em}{%
\hfill\color{Black}#3}} \par % Duration
\noindent \textit{#4} \par % School
\noindent\hangindent=2em\hangafter=0 \small #5 % Description
\normalsize \par}
```

```
\newcommand{\Language}[2]{
\noindent \textbf{#1}
\noindent \small \textit{#2}}
\newcommand{\Text}[1]{\par
\noindent \small #1
\normalsize \par}
\newcommand{\Textlong}[4]{
\noindent \textbf{#1} \par
\sepspace
\noindent \small #2
\par\sepspace
\noindent \small #3
\par\sepspace
\noindent \small #4
\normalsize \par}

\newcommand{\PaperEntry}[7]{
\noindent #1, ``\href{#7}{#2}", \textit{#3} \textbf{#4}, #5 (#6).}


\newcommand{\ArxivEntry}[3]{
\noindent #1, ``\href{http://arxiv.org/abs/#3}{#2}", \textit{{cond-mat/}#3}.}
\newcommand{\BookEntry}[4]{
\noindent #1, ``\href{#3}{#4}", \textit{#3}.}
\newcommand{\FundingEntry}[5]{
\noindent #1, ``#2", \$#3 (#4, #5).}

\newcommand{\TalkEntry}[4]{
\noindent #1, #2, #3 #4}

\newcommand{\ThesisEntry}[5]{
\noindent #1 -- #2 #3 ``#4" \textit{#5}}

\newcommand{\CourseEntry}[3]{
\noindent \item{#1: \textbf{#2} \\ #3}}

%%% Begin Document
%%% ------------------------------------------------------------
\begin{document}

%\layout

% you can upload a photo and include it here...
\begin{wrapfigure}{l}{0.5\textwidth}
\vspace*{-2em}
\includegraphics[width=0.15\textwidth]{image}
\end{wrapfigure}

\MyName{<!Name>}
\MySlogan{Curriculum Vit\ae\ (\today)}
```

```
\sepspace
\sepspace

%%% Personal details
%%% ----------------------------------------------------------
\NewPart{}{}

\PersonalEntry{Address:}{<!Adress>}
\PersonalEntry{Phone:}{<!PhoneNumber>}
\PersonalEntry{Mail:}{<!Email>}
\PersonalEntry{Nationality:}{<!Nationality>}
\PersonalEntry{Born:}{<!DateOfBirth>}



%%% Education
%%% ----------------------------------------------------------
\NewPart{Obrazovanje}{}

<!Education>

%%% Work experience
%%% ----------------------------------------------------------
\NewPart{Radno iskustvo}{(Reference na zahtjev)}

<!Work>



%%% OTHER QUALIFICATIONS
%%% ----------------------------------------------------------
\NewPart{Ostale vjestine i postignuca}{}

<!Qualification>



%%% LANGUAGES
%%% ----------------------------------------------------------
\NewPart{Jezici}{}

<!Language>


\end{document}
```