



Univerzitet u Novom Sadu
Tehnički fakultet »Mihajlo Pupin«
Zrenjanin



SEMINARSKI RAD

Predmet: Programski prevodioci

Tema: Kreiranje objektno-orjentisane C# desktop aplikacije uz samostalno kreiranje biblioteke klasa i analiza ispravnosti programskog koda primenom Visual Studio NET okruženja

***- Aplikacija za održavanje sati rada u IT firmi–
(TIP 2)***

Predmetni nastavnik: Prof. dr Ljubica Kazi

Student: Nikola Bobinac

Broj indeksa: SI 18/17

Zrenjanin, 2020.godina

Sadržaj

1. ZADATAK.....	4
2. OPIS POSLOVNOG KONTEKSTA REALIZOVANOG PRIMERA	5
3. KRATAK OPIS RAZVOJNOG OKRUŽENJA	6
3.1. Microsoft Visual Studio 2019	6
3.1.1. Namena alata.....	6
3.1.2. Izgled ekrana alata	6
3.1.3. Opis načina korišćenja	7
3.2. Microsoft SQL Server Management Studio 18.....	11
3.2.1. Namena alata.....	11
3.2.2. Izgled ekrana alata	11
3.2.3. Opis načina korišćenja	12
4. Korisničko upustvo razvijene aplikacije.....	14
4.1. Namena aplikacije i osnovne funkcije	14
4.2. Ekрани I način korišćenja.....	14
4.3. Podešavanja aplikacija	18
5. Ključni elementi implementacije	19
5.1. SQL Script.....	19
5.2. Ključni delovi programskog koda sa objašnjenjem	19
5.2.1. Uključivanje biblioteke klasa.....	19
5.2.2. Početni događaj pri otvaranju aplikacije.....	19
5.2.3. Klijent i Posao servisi	19
5.2.4. Popunjavanje tabele	22
5.2.5. Snimanje podataka	22
5.2.6. Poništavanje vrednosti iz forme.....	24
5.2.7. Menjanje watermark vrednosti za Date Picker	24
5.2.8. Ograničavanje textbox-a za sate na decimalne numeričke vrednosti	25
5.2.9. Filtriranje po imenu i prezimenu i reset filtera	26
6. Greške programskog koda	28
6.1. Segmenti programskog koda sa greškama i korekcijama	28
6.1.1. Sintaksne Greške.....	28

6.1.2.	Leksičke greške.....	28
6.1.3.	Semantičke greške	29
6.1.4.	Run-time greške	29
6.2.	Pravila I EBNF prikaz gramatičko ispravnog programskog koda	30
7.	Prikaz primene alata u detektovanju grešaka.....	31
7.1.	Ekranski prikaz izveštaja kompajlera nad neispravnim kodom i nakon popravke	31
7.2.	Ekranski prikaz reakcije na runtime greške	32
8.	Zaključak	34
9.	Literatura.....	35
10.	Listing	36
10.1.	Listing korisničkog interfejsa	36
10.1.1.	Dizajn MainWindow.xaml.....	36
10.1.2.	Funkcionalnost MainWindow.xaml.cs	37
10.1.3.	Konfiguracioni fajl App.config.....	42
10.1.4.	Funkcionalnost KlijentiServis.cs	42
10.1.5.	Funkcionalnost PosaoServis.cs	43
10.2.	Listing biblioteke KlasaPodataka	44
10.2.1.	Model clsRaspredRada.cs	44
10.2.2.	Funkcionalnost clsRasporedRadaDB.cs	45
10.3.	Listing biblioteke SQLDBUtils	46
10.3.1.	clsSqlKonekcija	46
10.3.2.	clsSqlTabela.....	48

1. ZADATAK

Kreirati desktop aplikaciju C# sa primenom gotove biblioteke SQL DB Utils. Kreirati biblioteku KlasePodataka koja radi sa podacima iz baze podataka i primenjuje SQLDBUtils (rad sa jednom tabelom iz baze podataka). Kreirati u kodu namerne greške 4 tipa – leksičku, sintaksnu, semantičku i run-time. Pustiti kompajler da detektuje greške i prikazati izveštaj koji daje u vezi grešaka i prikaz ispravnog rada.

2. OPIS POSLOVNOG KONTEKSTA REALIZOVANOG PRIMERA

U ovom seminarskom radu opisana je desktop aplikacija za evidenciju odrađenih sati u IT firmi koja se bavi outsourcing-om. Aplikacija omogućava unos novih poslova u raspored rada, kao i prikaz istorije odrađenih sati. Za svaku stavku rada su dati sledeći podaci: *Ime*, *Prezime*, *Klijent*, *Posao*, *Datum*, *Sati*. Podatak klijent je moguće ručno ukucati ili odabrati klijenta sa predefinisane liste koja se čuva u tekstualnom fajlu, dok za Posao je moguće samo birati poslove sa predefinisane liste koja se čuva u tekstualnom fajlu.

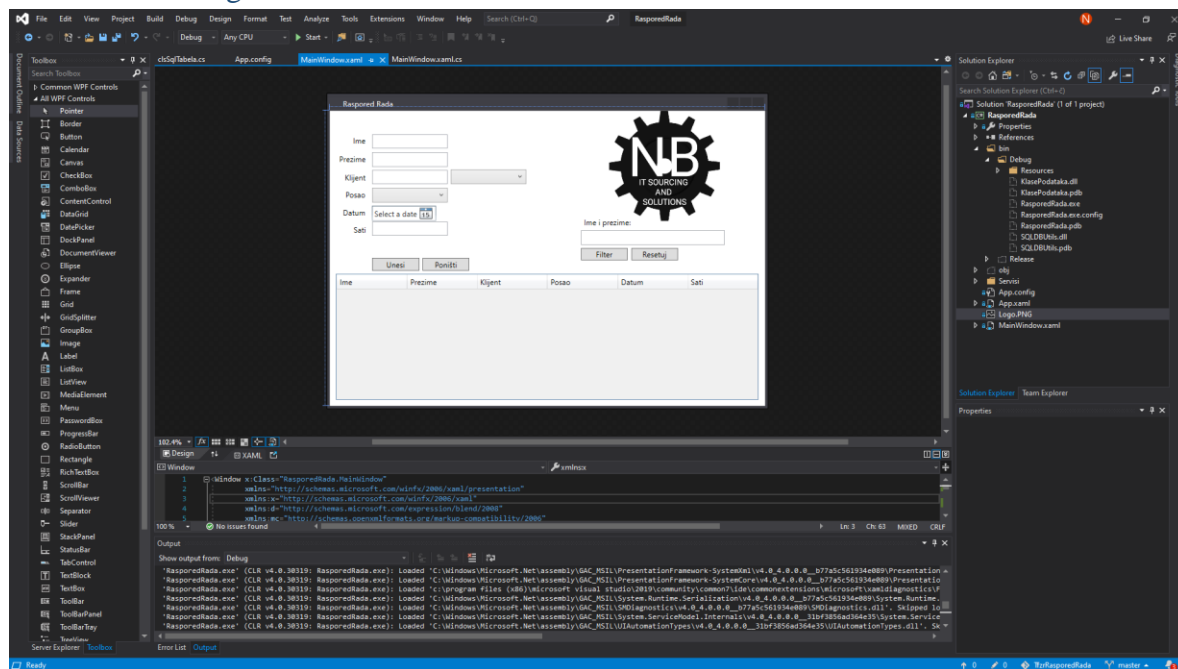
3. KRATAK OPIS RAZVOJNOG OKRUŽENJA

3.1. Microsoft Visual Studio 2019

3.1.1. Namena alata

Microsoft Visual Studio 2019 je Majkrosoftovo integrisano razvojno okruženje (IDE-Integrated development enviroment). Koristi se za razvoj kompjuterskih programa, sajtova, veb sajtova, veb aplikacija kao i mobilnih aplikacija. Visual Studio koristi različite platforme za kreiranje projekata, poput WPF (Windows Presentation Foundation) i Class Library. Podržava 36 raličitih programskih jezika. Visual Studio uključuje debugger koji radi i kao ispravljač grešaka na nivou izvora i kao ispravljač na nivou mašine.

3.1.2. Izgled ekrana alata



Slika 1. Izgled alata Microsoft Visual Studio 2019 nakon učitavanja projekta desktop aplikacije

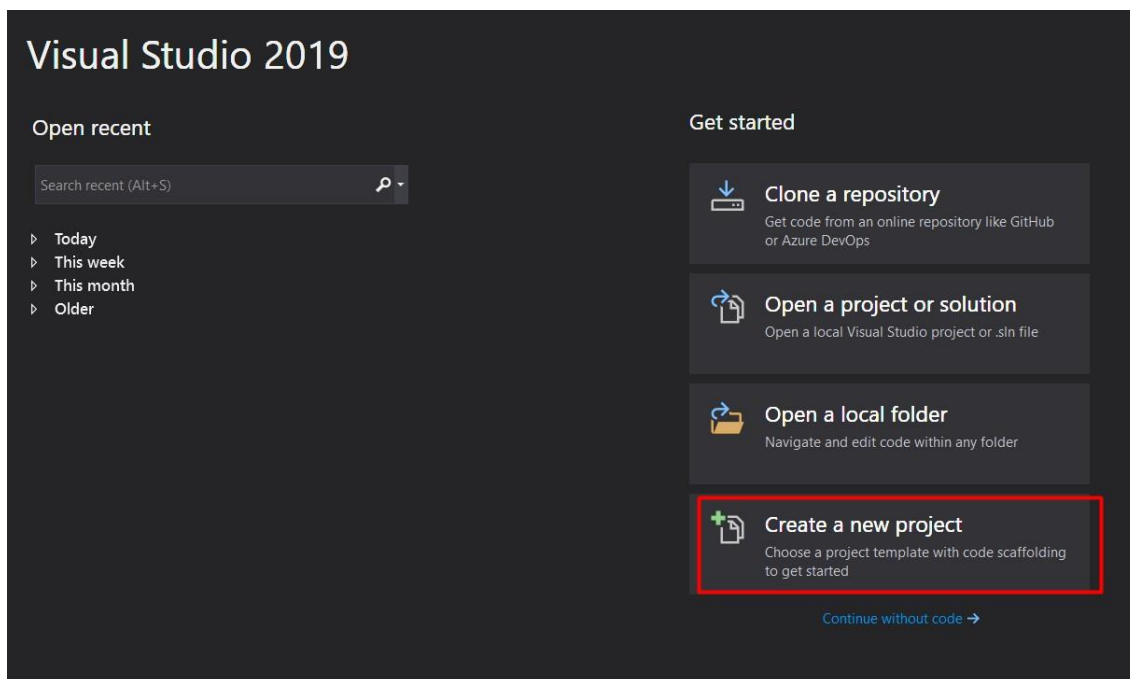
Na slici 1.prikazan je završni izgled aplikacije u razvojnom okruženju Microsoft Visual Studio 2019. Sa leve strane nalazi se *ToolBox*, koji omogućava lakše postavljanje grafičkih kontrola (dugmad, labele, textbox, itd). Sa desne strane se nalazi *Solution Explorer*, koji daje pregled fajlova uključenih u ovaj projekat.

Preko *Solution Explorer* ->*References* uključujemo naše biblioteke klasa, *KlasePodataka* i *SQLDBUtils*.

Zatim ispod *Solution Explorer*-a nalazi se prozor *Properties*, gde se podešavaju karakteristike (pozicija, naziv, tekst koji se prikazuje, itd) vezane za grafičke kontrole.

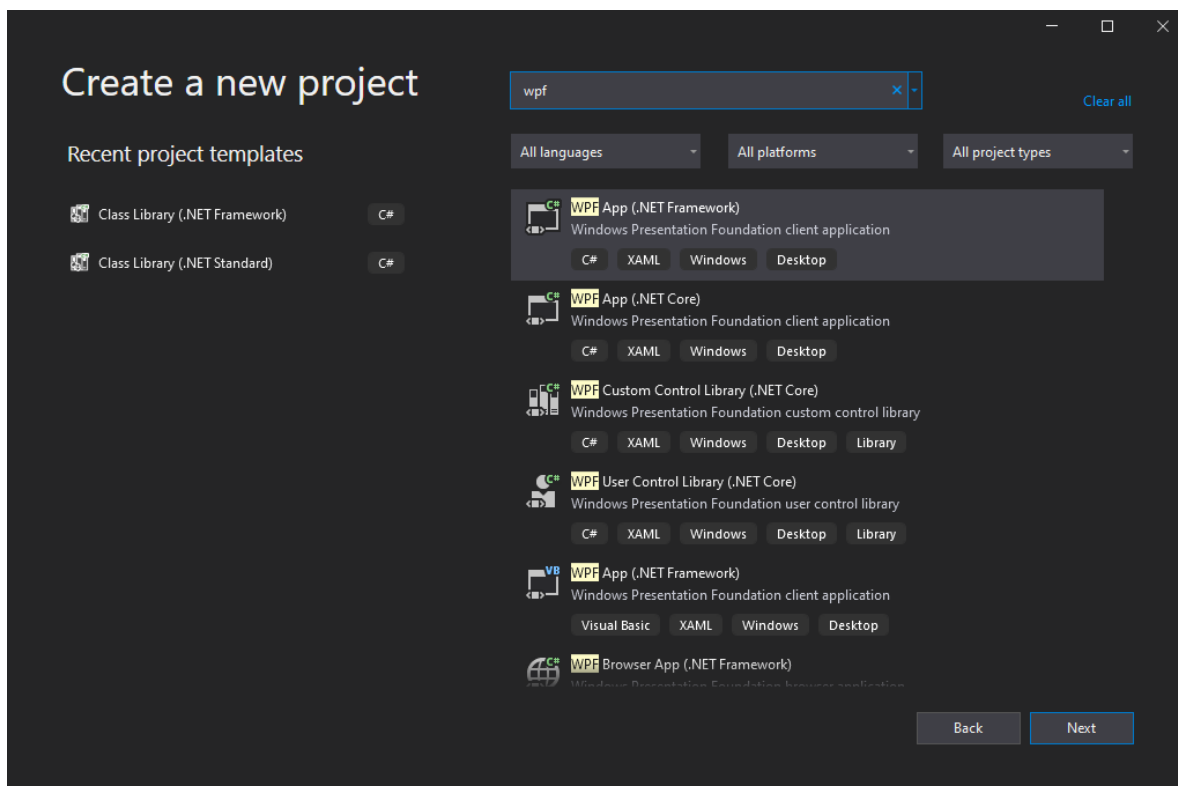
3.1.3. Opis načina korišćenja

Kada otvorimo alat u kojem pravimo našu aplikaciju, izgled ekrana je sledeći (Slika 2.):



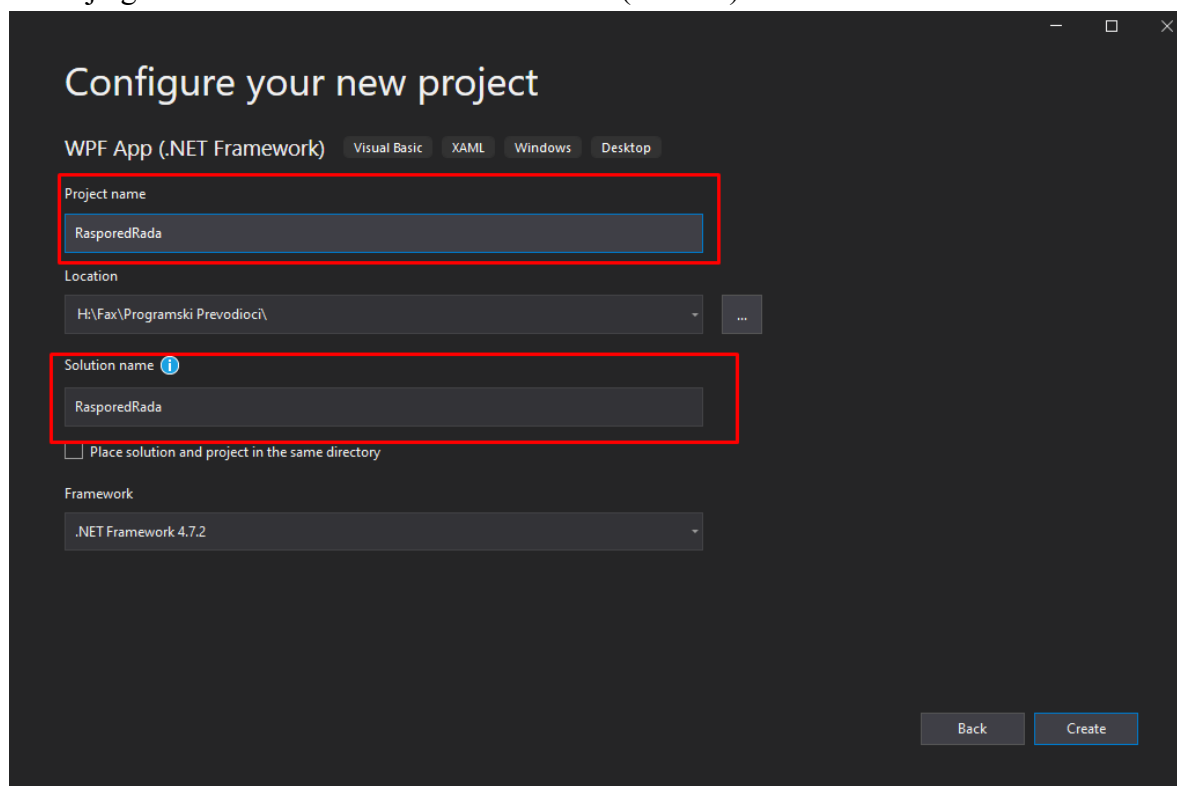
Slika 2. Kreiranje novog projekta u alatu Microsoft Visual Studio 2019

Kada se pritisne na kreiranje novog projekta, dobija se sledeći prozor u kojem se pomoću filtera bira programski jezik, platforma, kao i koju vrstu aplikacije želimo da pravimo. U našem slučaju biramo programski jezik C#, platforma je Windows, aplikacija je Desktop. Zatim nam program da koje sve vrste aplikacija možemo praviti, biramo WPF App(.NET Framework) (Windows Presentation Foundation), prikazano na slici 3.



Slika 3. Kreiranje WPF App (.NET Framework)

Kao poslednji korak pri kreiranju projekta, jeste davanje naziva projektu i njegovu lokaciju gde će biti sačuvan na našem računaru (Slika 4.).

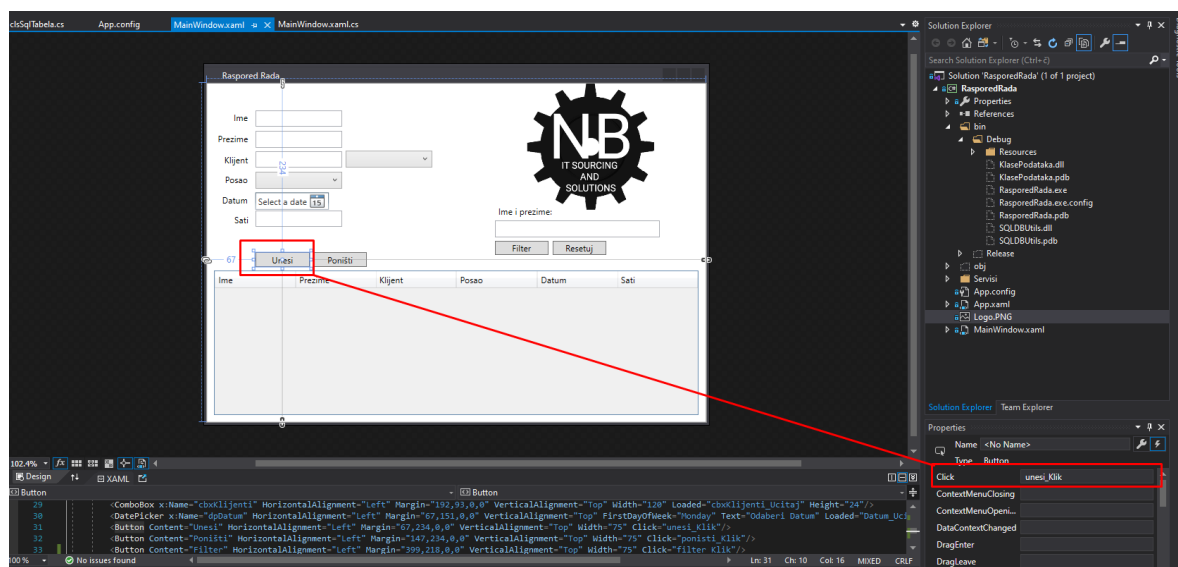


Slika 4. Dodavanje naziva i lokaciju projekta

Nakon pritiska na dugme *Create*, dobijamo grafički dizajn korisničkog interfejsa, a zatim u programskom kodu dodajemo određene akcije.

Kada se kreira grafički dizajn korisničkog interfejsa, treba kodirati funkcionalnosti vezane za našu aplikaciju. Te funkcionalnosti možemo dobiti na tri načina:

1. Način – Preko prozora *Properties*, pritiskom na „munju“, mogu se dodeliti događaji koje želimo za određenu alatku. (npr.za dugme želim događaju „klik“).Slika 5.



Slika 5. Opcija za dodelu događaja grafičkom objektu

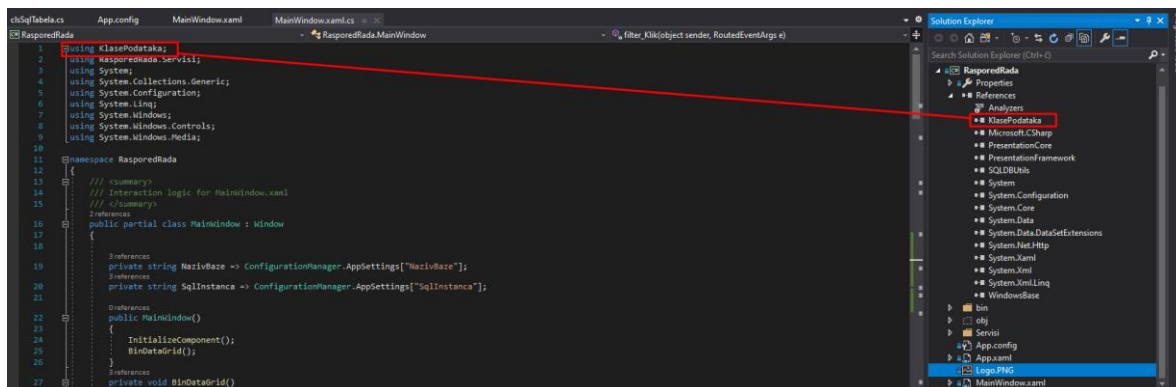
2. Način – da se pozicioniramo na određeni alat kojem želimo da dodelimo neki događaj i da kliknemo dva puta na njega. Ovo je kraći put ako želimo da nam događaj bude „klik“.
3. Način – u XAML ručno napišemo event handler za Click akciju (npr Click="unesi_Klik").

Pošto naš projekat zahteva biblioteke klasa, da bi se one kreirale biramo poseban projekat *Class Library*. Pri otvaranju tog projekta dobijamo klasu, koju sami modifikujemo u odnosu na naše potrebe. Ako želimo još klasa u datom projektu, biramo opciju *Project -> Add class...* (ili prečicom preko tastature, *Shift+Alt+C*).

Da bi koristili funkcionalnosti tih klasa, moramo da ih priključimo našoj aplikaciji. Pored standardnih biblioteka koje naša aplikacija automatski koristi, mi biramo i naše biblioteke koje smo napravili. Te biblioteke su *KlasaPodataka* i *SQLDBUtils*.

Dodaju se preko *References*, koji se nalazi u *Solution Explorer*-u. Desni klik na *References* -> *Add references*, ili *Project -> Add references*. Biramo *.dll* fajlove. Kada smo uspešno

dodali u references, moramo ga napomenuti i u kodu, tačnije u *USING* delu koda, da bi nam program prepoznao biblioteku (slika 6.)



Slika 6. Dodavanje biblioteke klasa KlasePodataka u aplikaciju

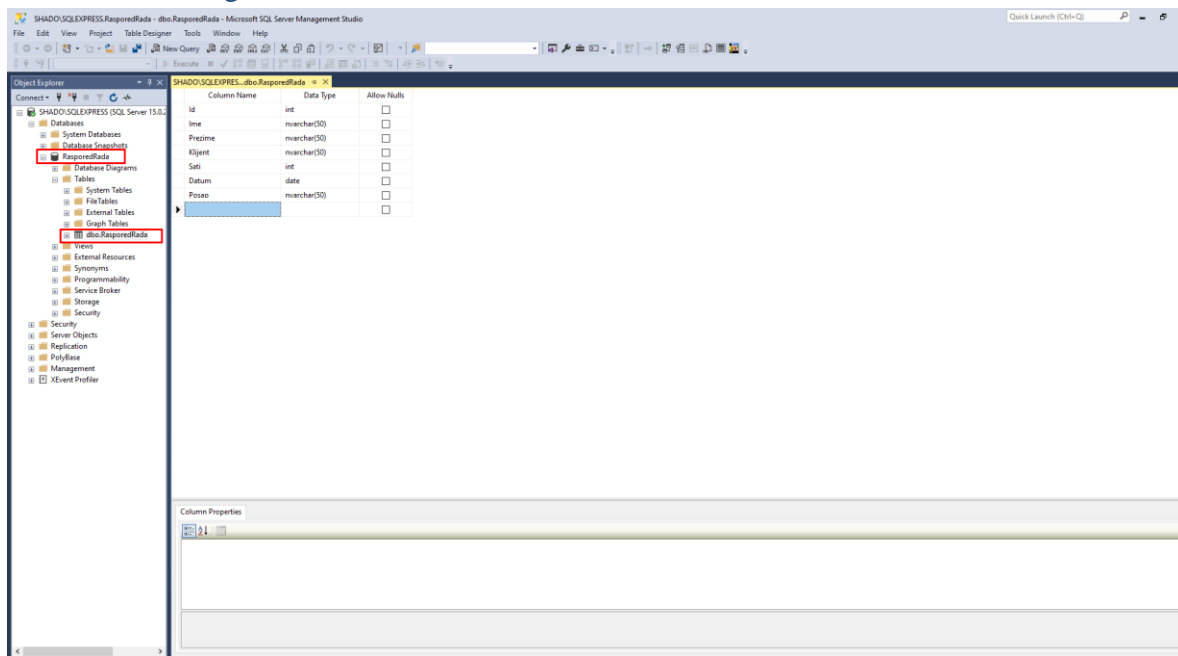
Biblioteku klasa *SQLDBUtils* dodajemo kao referencu (biblioteku) u projekat u kom se nalaza *KlasePodataka*, u njima se nalaze metode potrebne za konekciju sa bazom, čitanje iz baze, ažuriranje baze itd. Dok biblioteku *KlasePodataka* dodajemo kao referencu (biblioteku) u samoj aplikaciji.

3.2. Microsoft SQL Server Management Studio 18

3.2.1. Namena alata

SQL Server Management Studio (SSMS) je softverska aplikacija koja se koristi za konfigurisanje, upravljanje i administraciju svih komponenti u Microsoft SQL Serveru. Alat uključuje uređivače skripti i grafičke alate koji rade sa objektima i komponentama servera.

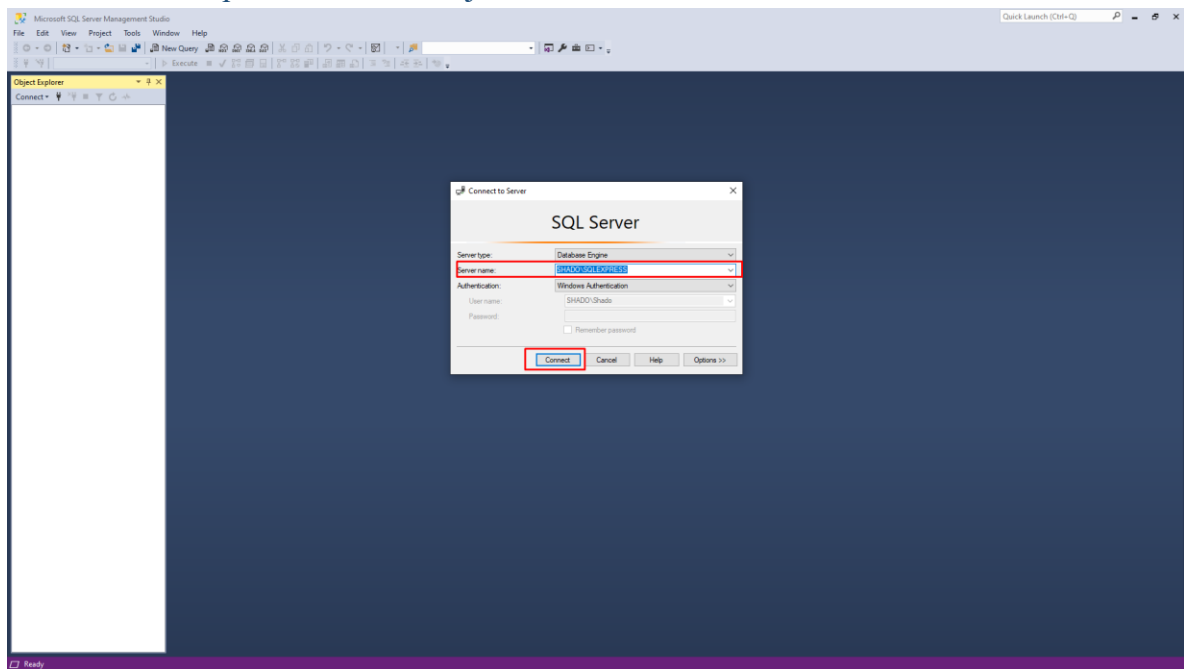
3.2.2. Izgled ekrana alata



Slika 7. Izgled ekrana alata Microsoft SQL Server Management Studio 18

Na slici 7. možemo videti izgled tabele RasporedRada, iz baze podataka RasporedRada, na slici se vidi kolone u datoj tabeli u sredini ekrana. Sa leve strane gore tekst predstavlja ime servera koji je potreban za konekciju naše aplikacije sa bazom podataka. Takođe sa leve strane se vidi baza podataka RasporedRada i tabela RasporedRada, uokvireno crvenom bojom.

3.2.3. Opis načina korišćenja

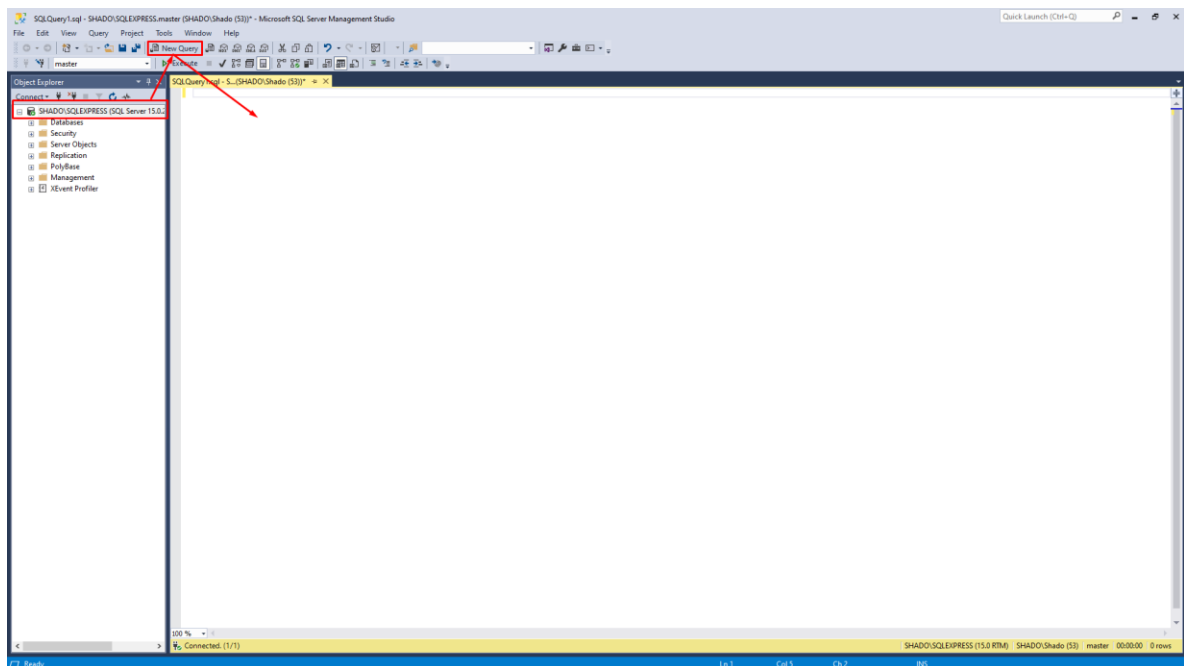


Slika 8. Početni ekran pokretanja Microsoft SQL Server Management Studio 18

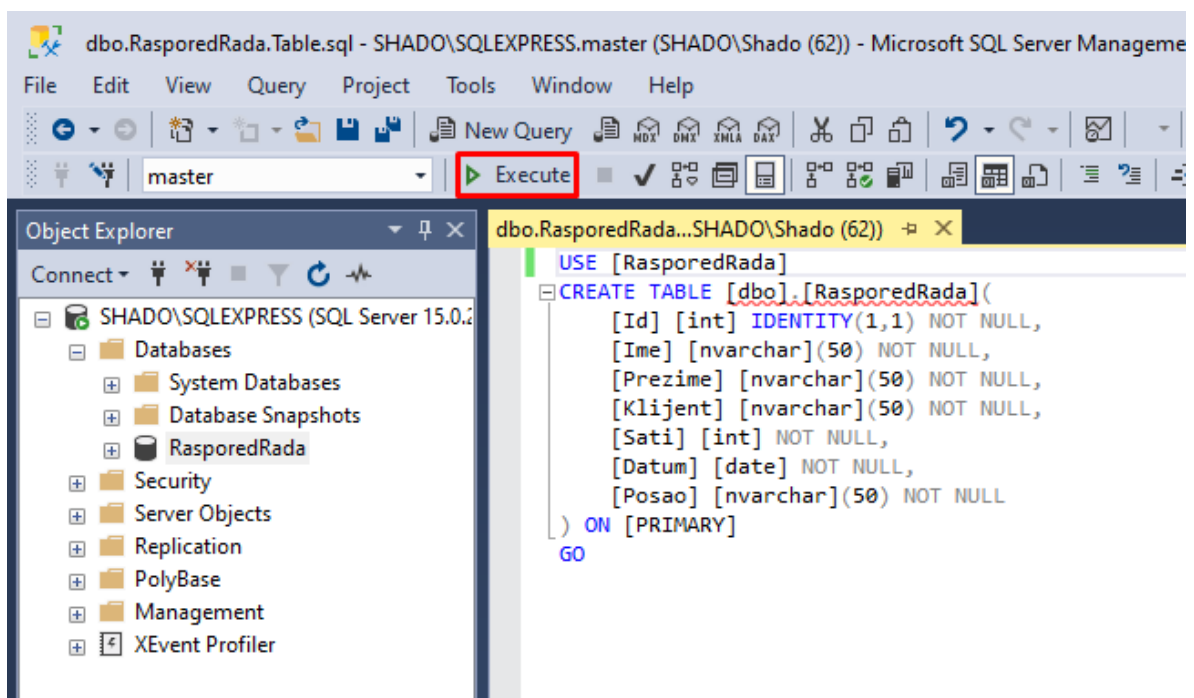
Pri pokretanju Microsoft SQL Server Management Studio-a dobija se prozor prikazan na slici 8. Pre kreiranja baze prvo se moramo konektovati na server, ukoliko je naziv servera dobar, konekcija će biti uspešna.

Zatim, kada se uspešno konektujemo, pozicioniramo se na naš naziv servera i kliknemo na *New Query*, dobićemo prazno polje prikazan na slici 9.

Jedan od načina pravljenja baze podataka je kucanje koda, koristeći sintaksu SQL jezika. Pisani kod pri kreiranju baze *Raspored Rada* dat je na slici 10. Kada je kod napisan pravilno klikne se na *Execute*, i ako takva baza ne postoji, baza podataka će biti uspešno napravljena (kao što se može videti na slici 7.).



Slika 9. Postupak pravljenja baze podataka



Slika 10. Kod za pravljenje baze podataka

4. Korisničko upustvo razvijene aplikacije

4.1. Namena aplikacije i osnovne funkcije

Osnovna namena desktop aplikacije je evidencija koliko sati koji zaposleni u IT firmi radi za kog klijenta na poslu kog tipa.

Osnovne funkcije aplikacije su:

- Tabelarni prikaz svih stavki rada u rasporedu rada (eng. *Timesheet*)
- Unos nove stavke rada
- Filtriranje podataka po imenu I prezimenu
- Odustajanje od unosa nove stavke rada.

4.2. Ekran i način korišćenja

Kada se pokrene aplikacija dobija se izgled sličan kao u dizajneru, osim što se automatski pri otvaranju aplikacije učitaju podaci iz baze podataka i popune u Data Grid. Početni ekran aplikacije može se videti na slici 11.

The screenshot shows a desktop application window titled "Raspored Rada". On the left is a form with fields for "Ime", "Prezime", "Klijent" (with a dropdown), "Posao" (with a dropdown), "Datum" (with a date picker showing "15"), and "Sati". Below the form are "Unesi" and "Poništi" buttons. On the right is a logo for "NB IT SOURCING AND SOLUTIONS" and a search section with the label "Ime i prezime:", a text input field, and "Filter" and "Resetuj" buttons. At the bottom is a table with 6 columns: "Ime", "Prezime", "Klijent", "Posao", "Datum", and "Sati".

Ime	Prezime	Klijent	Posao	Datum	Sati
Nikola	Bobinac	Activision	Development	9/10/2020 12:00:00	5
Aleksa	Cakic	EA Games	Bug fix	9/10/2020 12:00:00	3

Slika 11. Prikaz podataka pri otvaranju aplikacije

Zatim kada popunimo sve podatke koji se traže i kliknemo na opciju Unesi, dobija sa poruka o unosu podataka u bazu i tabela u aplikaciji se osveži sa novim informacijama, slika 12.

Podatak Klijent može da se unese kao tekst, ili odabere postojeći klijent iz dropdown liste koja se definiše u odvojenom tekstualnom fajlu.

Podatak Posao se učitava iz liste koja je definisana u odvojenom tekstualnom fajlu.

The screenshot shows the 'Raspored Rada' application window. On the left, there is a form with the following fields: 'Ime' (Nikola), 'Prezime' (Bobinac), 'Klijent' (TFZR), 'Posao' (PM Work), 'Datum' (9/11/2020), and 'Sati' (1). Below the form are buttons for 'Unesi' and 'Poništi'. On the right, there is a logo for 'NB IT SOURCING AND SOLUTIONS' and a section for 'Ime i prezime:' with a text box and 'Filter' and 'Resetuj' buttons. In the center, a small dialog box displays the message 'Uspešno uneta stavka!' with an 'OK' button. At the bottom, there is a table with 6 columns: 'Ime', 'Prezime', 'Klijent', 'Posao', 'Datum', and 'Sati'.

Ime	Prezime	Klijent	Posao	Datum	Sati
Nikola	Bobinac	Activision	Development	9/10/2020 12:00:00	5
Aleksa	Cakic	EA Games	Bug fix	9/10/2020 12:00:00	3
Nikola	Bobinac	TFZR	Communication	9/18/2020 12:00:00	5
Eduardo	Boieru	Blizzard	Bug fix	9/3/2020 12:00:00	1
Nikola	Bobinac	Test	Deployment	9/2/2020 12:00:00	2
Nikola	Bobinac	TFZR	PM Work	9/11/2020 12:00:00	1

Slika 12. Prikaz aplikacije sa popunjenim podacima i klikom na Unesi

Nakon što se klikne na *Ok* dugme u prozoru sa obaveštenjem podaci u text boxovima se resetuju na prazno, isto to se postiže i klikom na dugme *Poništi*, slika 13.

Raspored Rada

Ime

Prezime

Klijent

Posao

Datum

Sati

Odaberi datum

15

Unesi

Poništi

NB

IT SOURCING AND SOLUTIONS

Ime i prezime:

Filter

Resetuj

Ime	Prezime	Klijent	Posao	Datum	Sati
Nikola	Bobinac	Activision	Development	9/10/2020 12:00:00	5
Aleksa	Cakic	EA Games	Bug fix	9/10/2020 12:00:00	3
Nikola	Bobinac	TFZR	Communication	9/18/2020 12:00:00	5
Eduardo	Boieru	Blizzard	Bug fix	9/3/2020 12:00:00	1
Nikola	Bobinac	Test	Deployment	9/2/2020 12:00:00	2
Nikola	Bobinac	TFZR	PM Work	9/11/2020 12:00:00	1

Slika 13. Prikaz aplikacije nakon klika na dugme Poništi

Opcija *Filter* omogućava da se rezultati filtriraju po Imenu i Prezimenu onoga ko je radio neki posao. Potrebno je uneti ime i prezime kao dve reči razdvojene razmakom. Ukoliko unos nije takav izbacuje se poruka za grešku, slika 14.

Raspored Rada

Ime
Prezime
Klijent
Posao
Datum
Sati

×

Nije uneto ime i prezime, razdvojeno jednim razmakom

OK

ime i prezime:

Ime	Prezime	Klijent	Posao	Datum	Sati
Nikola	Bobinac	Activision	Development	9/10/2020 12:00:00	5
Aleksa	Cakic	EA Games	Bug fix	9/10/2020 12:00:00	3
Nikola	Bobinac	TFZR	Communication	9/18/2020 12:00:00	5
Eduardo	Boieru	Blizzard	Bug fix	9/3/2020 12:00:00	1
Nikola	Bobinac	Test	Deployment	9/2/2020 12:00:00	2
Nikola	Bobinac	TFZR	PM Work	9/11/2020 12:00:00	1

Slika 14. Prikaz greške pri filtriranju

Ukoliko se ime i prezime unese kako je traženo, i zatim odabere opcija *Filter*, tabela će se popuniti samo sa stavkama rada za tu osobu, slika 15.

Ime:

Prezime:

Klijent:

Posao:

Datum:

Sati:

Ime i prezime:

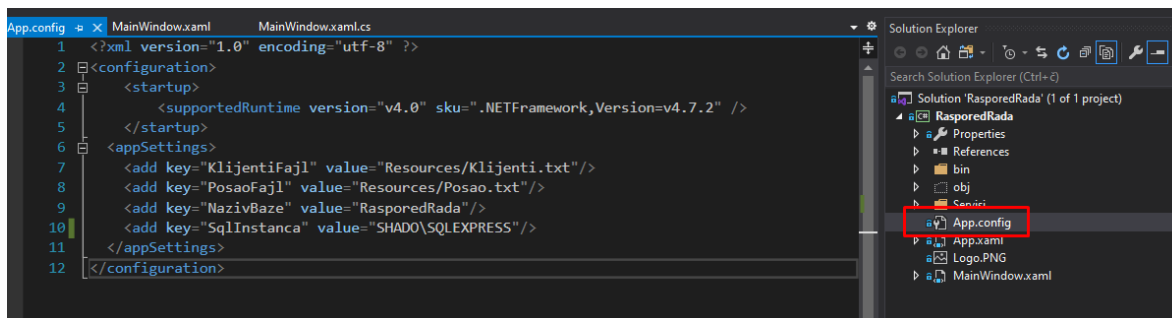
Ime	Prezime	Klijent	Posao	Datum	Sati
Nikola	Bobinac	Activision	Development	9/10/2020 12:00:00	5
Nikola	Bobinac	TFZR	Communication	9/18/2020 12:00:00	5
Nikola	Bobinac	Test	Deployment	9/2/2020 12:00:00	2
Nikola	Bobinac	TFZR	PM Work	9/11/2020 12:00:00	1

Slika 15. Prikaz uspešnog filtriranja

Opcija *Resetuj* briše vrednost iz polja za filtriranje i popunjava tabelu sa svim stavkama.

4.3. Podešavanja aplikacija

Kroz App.config moguće je promeniti SQL Instancu, naziv baze podataka, kao i putanje do tekstualnih fajlova za Klijente i Poslove.



Slika 16. Konfiguracije aplikacije

5. Ključni elementi implementacije

5.1. SQL Script

```
CREATE DATABASE [RasporedRada]
GO
USE [RasporedRada]
CREATE TABLE [dbo].[RasporedRada](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Ime] [nvarchar](50) NOT NULL,
    [Prezime] [nvarchar](50) NOT NULL,
    [Klijent] [nvarchar](50) NOT NULL,
    [Sati] [int] NOT NULL,
    [Datum] [date] NOT NULL,
    [Posao] [nvarchar](50) NOT NULL
) ON [PRIMARY]
GO
```

Tabela 1. Skripta za kreiranje baze i tabele

5.2. Ključni delovi programskog koda sa objašnjenjem

5.2.1. Uključivanje biblioteke klasa

Ako je referenca na .dll fajl pravilno podešena namespace-ovima iz biblioteke se može pristupiti sa sledećom linijom:

```
using KlasePodataka;
```

5.2.2. Početni događaj pri otvaranju aplikacije

Pri pokretanju WPF aplikacije poziva se MainWindow i njegov konstruktor. Main window takođe ima i privatne attribute za naziv baze i sql instancu koje vuče iz konfiguracije, tabela 2.

```
private string NazivBaze => ConfigurationManager.AppSettings["NazivBaze"];
private string SqlInstanca => ConfigurationManager.AppSettings["SqlInstanca"];

public MainWindow()
{
    InitializeComponent();
    BinDataGrid();
}
```

Tabela 2. Deo klase MainWindow koji sadrži konstruktor i privatne attribute

5.2.3. Klijent i Posao servisi

Statičke klase KlijentServis, tabela 3, i PosaoServis, tabela 4, zadužene su da iz tekstualnog fajla podešenog preko konfiguracije pretvore Klijente i Poslove u liste.

```
namespace RasporedRada.Servisi
{
    public static class KlijentiServis
    {
        static string PutanjaFajla =>
        ConfigurationManager.AppSettings["KlijentiFajl"];
    }
}
```

```

public static IEnumerable<string> DajKlijenteIzFajla()
{
    IEnumerable<string> klijenti = Enumerable.Empty<string>();

    if (!string.IsNullOrEmpty(PutanjaFajla))
    {
        try
        {
            klijenti = System.IO.File.ReadAllLines(PutanjaFajla);
        }
        catch (Exception e)
        {
            MessageBox.Show("Putanja fajla za klijente neispravna!");
        }
    }
    else
    {
        MessageBox.Show("Putanja fajla za klijente je prazna!");
    }

    return klijenti;
}
}

```

Tabela 3. Statička klasa KlijentiServis

```

namespace RasporedRada.Servisi
{
    public static class PosaoServis
    {
        static string PutanjaFajla =>
        ConfigurationManager.AppSettings["PosaoFajl"];

        public static IEnumerable<string> DajPosloveIzFajla()
        {
            IEnumerable<string> poslovi = Enumerable.Empty<string>();

            if (!string.IsNullOrEmpty(PutanjaFajla))
            {
                try
                {
                    poslovi = System.IO.File.ReadAllLines(PutanjaFajla);
                }
                catch (Exception e)
                {
                    MessageBox.Show("Putanja fajla za poslove je neispravna!");
                }
            }
            else
            {
                MessageBox.Show("Putanja fajla za poslove je prazna!");
            }

            return poslovi;
        }
    }
}

```

```

    }
}
}

```

Tabela 4. Statička klasa PosaoServis

U slučaju greške sa putanjama do fajlova izbacuje se prozor sa porukom da je putanja neispravna, a u slučaju da u konfiguraciji ne postoji putanja fajla izbacuje se poruka da je putanja prazna. U oba slučaja vraća se prazna string lista, a ako su putanje u redu onda popunjena string lista.

Ove dve statičke klase i metode pozivaju combo box-evi za Posao i Klijent prilikom učitavanja aplikacije, tabela 5 i tabela 6.

```

private void cbxKlijenti_Ucitaj(object sender, RoutedEventArgs e)
{
    List<string> klijenti = KlijentiServis.DajKlijenteIzFajla().ToList();

    if (klijenti != null && klijenti.Any())
    {
        foreach (string klijent in klijenti)
        {
            if (!string.IsNullOrWhiteSpace(klijent))
            {
                cbxKlijenti.Items.Add(klijent);
            }
        }
    }
    else
    {
        cbxKlijenti.Opacity = 0;
    }
}

```

Tabela 5. Metoda za učitavanje vrednosti iz liste u combo box za Klijente

```

private void cbxPosao_Ucitaj(object sender, RoutedEventArgs e)
{
    List<string> poslovi = PosaoServis.DajPosloveIzFajla().ToList();

    if (poslovi != null && poslovi.Any())
    {
        foreach (string posao in poslovi)
        {
            if (!string.IsNullOrWhiteSpace(posao))
            {
                cbxPosao.Items.Add(posao);
            }
        }
    }
}

```

```

        else
        {
            cbxPosao.Opacity = 0;
        }
    }

```

Tabela 6. Metoda za učitavanje vrednosti iz liste u combo box za Poslove

Ove dve metode pri učitavanju pozivaju odgovarajuće servise i učitavaju u dizajn vrednosti iz liste, ili ako je lista prazna učine to polje nevidljivim.

5.2.4. Popunjavanje tabele

Za popunjavanje tabele je zadužena BinDataGrid metoda. Ona kreira novi objekat tipa clsRasporedRadaDB koji je iz biblioteke KlasePodataka i služi za komunikaciju sa bazom, zatim iz baze učitava sve stavke i popunjava ih u data grid na dizajnu, tabela 7.

```

private void BinDataGrid()
{
    clsRasporedRadaDB rasporedRadaDB = new clsRasporedRadaDB(SqlInstanca, NazivBaze);
    DataGrid.ItemsSource =
    rasporedRadaDB.DajSveIzRasporeda().Tables["RasporedRada"].DefaultView;
}

```

Tabela 7. BinDataGrid metoda za popunjavanje tabele na dizajnu

5.2.5. Snimanje podataka

Prilikom snimanja podataka vrši se provera da li je svako polje popunjeno, potom se kreira objekat klase clsRasporedRada iz biblioteke KlasePodataka, objekat se popunjava unešenim podacima i radi se pokušaj unosa podataka u bazu, tabela 8.

Ako je pokušaj uspešan tabela sa stavkama iz baze se osvežava, izbacuje se poruka o uspešnom unosu i prazne se polja na formi.

Ako pokušaj nije uspešan, prikazuje se prozor sa porukom greške.

```

private void unesi_Klik(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrWhiteSpace(txtIme.Text))
    {
        MessageBox.Show("Ime je prazno!");
        return;
    }

    if (string.IsNullOrWhiteSpace(txtPrezime.Text))
    {
        MessageBox.Show("Prezime je prazno!");
    }

    if (string.IsNullOrWhiteSpace(txtKlijent.Text))

```

```

        {
            if (cbxKlijenti.SelectedItem == null)
            {
                MessageBox.Show("Klijent nije odabran!");
            }
        }

        if (string.IsNullOrEmpty(txtSati.Text))
        {
            bool rezultat = double.TryParse(txtSati.Text, out double iSati);
            if (rezultat == false)
            {
                MessageBox.Show("Sati nisu uneti kao broj!");
                return;
            }

            if (iSati <= 0)
            {
                MessageBox.Show("Sati ne mogu biti manji ili jednaki 0");
            }
        }

        if (cbxPosao.SelectedItem == null)
        {
            MessageBox.Show("Posao nije odabran!");
        }

        clsRasporedRada objRasporedRada = new clsRasporedRada();

        string klijent = string.Empty;

        if (!string.IsNullOrEmpty(txtKlijent.Text))
        {
            klijent = txtKlijent.Text;
        }
        else
        {
            klijent = cbxKlijenti.SelectedItem.ToString();
        }

        objRasporedRada.Ime = txtIme.Text;
        objRasporedRada.Prezime = txtPrezime.Text;
        objRasporedRada.Klijent = klijent;
        objRasporedRada.Posao = cbxPosao.SelectedItem.ToString();
        objRasporedRada.Sati = double.Parse(txtSati.Text);
        objRasporedRada.Datum = (DateTime)dpDatum.SelectedDate;
        clsRasporedRadaDB rasporedRadaDB = new clsRasporedRadaDB(SqlInstanca,
NazivBaze);
        rasporedRadaDB.SnimiNoviUnosRada(objRasporedRada, out bool uspeh, out
string greska);

        if (!uspeh)
        {
            MessageBox.Show(greska);
        }
        else
        {
            BinDataGrid();
        }
    }
}

```

```

        MessageBox.Show("Uspešno uneta stavka!");
        ponistiFormu();
    }
}

```

Tabela 8. Metoda za proveru i unos podataka u bazu

5.2.6. Poništavanje vrednosti iz forme

Klikom na opciju *Poništi* poziva se metoda koja poziva drugu metodu za poništavanje forme. Ovo je odrađeno na ovaj način kako bi se izbeglo dupliciranje koda jer metoda za snimanje podataka takođe mora nakon uspešnog snimanja da očisti vrednosti iz forme, tabela 9 i tabela 10.

```

private void ponisti_Klik(object sender, RoutedEventArgs e)
{
    ponistiFormu();
}

```

Tabela 9. Metoda kada se klikne na opciju Poništii

```

private void ponistiFormu()
{
    txtIme.Text = string.Empty;
    txtPrezime.Text = string.Empty;
    txtSati.Text = "0";
    txtKlijent.Text = string.Empty;
    cbxPosao.SelectedIndex = -1;
    cbxKlijenti.SelectedIndex = -1;
    dpDatum.SelectedDate = null;
}

```

Tabela 10. Metoda za čišćenje vrednosti iz forme

5.2.7. Menjanje watermark vrednosti za Date Picker

Podatak Datum na formi koristi tip Date Picker, kroz properties za Date Picker nije moguće promeniti text „*Select a date*“ i zbog toga se moraju napisati odvojene metode koje će to odraditi, kako bi i vrednost na Date Pickeru bila na srpskom, kao ostatak aplikacije. Ova metoda se poziva pri učitavanju elementa za koji je povezana (u ovom slučaju Date Picker u XAML), tabela 11.

```

/*
 * Metoda za menjanje default texta na Date Picker-u iz "Select a date" u
 "Odaberi datum" pri učitavanju elementa
 * Preuzeto sa: https://social.msdn.microsoft.com/Forums/sqlserver/en-US/9eec87e0-4d12-430d-83fd-ce13dd96776b/datepicker-hide-quotselect-datequot-placeholder-or-change-it?forum=wpf
 * Metode Datum_Ucitan i FindVisualChild
 */
private void Datum_Ucitan(object sender, RoutedEventArgs e)
{
    DatePicker datePicker = sender as DatePicker;
}

```



```

        if (datePicker != null)
        {
            System.Windows.Controls.Primitives.DatePickerTextBox
datePickerTextBox =
FindVisualChild<System.Windows.Controls.Primitives.DatePickerTextBox>(datePicker);
            if (datePickerTextBox != null)
            {
                ContentControl watermark =
datePickerTextBox.Template.FindName("PART_Watermark", datePickerTextBox) as
ContentControl;
                if (watermark != null)
                {
                    watermark.Content = "Odaberi datum";
                }
            }
        }
    }

    private T FindVisualChild<T>(DependencyObject dependencyObject) where T :
DependencyObject
    {
        if (dependencyObject != null)
        {
            for (int i = 0; i <
VisualTreeHelper.GetChildrenCount(dependencyObject); i++)
            {
                DependencyObject child =
VisualTreeHelper.GetChild(dependencyObject, i);
                T result = (child as T) ?? FindVisualChild<T>(child);
                if (result != null)
                {
                    return result;
                }
            }
        }
        return null;
    }
}

```

Tabela 11. Metode za menjanje watermark teksta na Date Pickeru

5.2.8. Ograničavanje textbox-a za sate na decimalne numeričke vrednosti

Textbox inače prima bilo kakve alfa-numerički i simbolske karaktere. Za polje *Sati* bilo je potrebno ograničiti da se mogu uneti samo brojevi i decimalni brojevi, ova provera radi se preko Regexa, tabela 12.

```

/*
    * Provera i ograničenje za textbox za sate preko Regex-a kako bi se samo
    decimalni brojevi mogli uneti
    */
    private void SamoBrojevi(System.Object sender,
System.Windows.Input.TextCompositionEventArgs e)
    {
        e.Handled = DaLiJeTekstBroj(e.Text);
    }
}

```

```

private static bool DaLiJeTekstBroj(string str)
{
    System.Text.RegularExpressions.Regex reg = new
System.Text.RegularExpressions.Regex("[^0-9.-]+");
    return reg.IsMatch(str);
}

```

Tabela 12. Metoda za proveru da li je vrednost polja za sate broj

5.2.9. Filtriranje po imenu i prezimenu i reset filtera

Za filtriranje po imenu i prezimenu proverava se da li su unešene dve reči, koje predstavljaju ime i prezime, i zatim se iz objekta tipa `clsRasporedRadaDB` poziva odgovarajuća metoda za filtriranje i Data Grid se popunjava sa rezultatom, tabela 13.

```

private void filter_Klik(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(txtFilter.Text))
    {
        MessageBox.Show("Uneti parametar za filter");
    }

    if (txtFilter.Text.Split(' ').Length != 2)
    {
        MessageBox.Show("Nije uneto ime i prezime, razdvojeno jednim
razmakom");
    }
    else
    {
        clsRasporedRadaDB rasporedRadaDB = new
clsRasporedRadaDB(SqlInstanca, NazivBaze);
        DataGrid.ItemsSource =
rasporedRadaDB.DajRasporedPremaKorisniku(txtFilter.Text).Tables["RasporedRada"].De
faultView;
    }
}

```

Tabela 13. Metoda koja se poziva prilikom klika na opciju Filter

Opcija *Reset* prazni polje za filter i puni Data Grid sa svim stavkama iz baze, tabela 14.

```

private void resetuj_Klik(object sender, RoutedEventArgs e)
{
    PonistiFilter();
    BinDataGrid();
}

private void PonistiFilter()
{
    txtFilter.Text = string.Empty;
}

```

}

Tabela 14. Metoda za opciju resetovanja filtera

6. Greške programskog koda

Postoje tri glavne greške koje kompajler može detektovati, a to su:

1. Sintaksa greška
2. Leksička greška
3. Sematička greška

Postoji i takozvana run-time greška, koju kompajler ne može da detektuje, ali ona može da se uvidi pri pokretanju koda, obično je to pogrešna putanja do baze, nepostojeća tabela, ili kolona u tabeli i slično. U slučaju ove aplikacije do greške može doći i zbog nepravilnih putanja do tekstualnih fajlova i zato su odrađene provere unete kako aplikacija ne bi „pukla“.

Runtime greške se mogu umanjiti adekvatnim korišćenjem try-catch blokova koda.

6.1. Segmenti programskog koda sa greškama i korekcijama

U ovom segmentu biće predstavljene gore pomenute greške, ispravljen kod i objašnjenje greške.

6.1.1. Sintaksne Greške

Sintaksne greške se najčešće javlja zbog nedostatka ; na kraju reda ili () pri pozivu metode, tabela 15.

NEISPRAVNO	ISPRAVNO	OBJAŠNJENJE
BinDataGrid()	BinDataGrid();	Nedostaje ;
BinDataGrid;	BinDataGrid();	Potrebno je imati () pri pozivu metode

Tabela 15. Primeri sintaksnih grešaka

6.1.2. Leksičke greške

Leksičke greške se javljaju kada se ne napiše dobro tip podatka ili ime promenljive. Ove greške se često javljaju ako je programer navikao da koristi Tab dugme na tastaturi za završavanje onoga što je počeo da piše, u okruženjima koja implementiraju IntelliSense, ali okruženje na trenutak prikoči i ne dovrši tekst a programer ne primeti, tabela 16.

NEISPRAVNO	ISPRAVNO	OBJAŠNJENJE
str porukaGreske = string.Empty;	string porukaGreske = string.Empty;	Nije dobro napisan tip podataka string

txtIm.Text;	txtIme.Text;	txtIm promenljiva ne postoji.
-------------	--------------	-------------------------------

Tabela 16. Primeri leksičkih grešaka

6.1.3. Semantičke greške

Ove greške se pojavljuju kada promenljivoj pokušamo dodati vrednost koja nije istog tipa ili ako pokušamo da pristupimo neinicijalizovanoj promenljivoj, tabela 17.

NEISPRAVNO	ISPRAVNO	OBJAŠNJENJE
txtSati.Text = 0;	txtSati.Text = "0";	Promenljiva je tipa string i ne može da prima tip int.
objRaspored = new clsRasporedRada();	clsRasporedRada objRaspored = new clsRasporedRada();	Korišćenje promenljive koja nije inicijalizovana

Tabela 17. Primeri semantičkih grešaka

6.1.4. Run-time greške

Da bi videli run-time greške, treba uneti pogrešne podatke unutar koda korisničkog interfejsa. Kompajler se neće buniti, jer je sve dobro napisano i nema gore navedenih grešaka, ali kod će “pući” kada pokušamo da unesemo pogrešan put do baze podataka ili kada pokušamo da upišemo u tabelu koja se ne nalazi u datoj bazi podataka. Na sledećim primerima prikazan je ispravan i neispravan kod. Kako ova aplikacija vuče ovakve podatke iz konfiguracije, neki od primera će biti iz App.Config fajla.

Neispravno:

```
<add key="SqlInstanca" value="SQLEXPRESS"/>
```

Ispravno:

```
<add key="SqlInstanca" value="SHADO\SQLEXPRESS"/>
```

U programskom kodu biblioteke klasa, *KlasaPodataka*, u klasi *clsRasporedRadaDB* postavljena je fiksna putanja za konekciju sa bazom. Promenljiva *objSqlTabela* predstavlja objekat koje se instancira pomoću konstruktora koji prima dva parametra, konekciju ka bazi podataka i naziv tabele u datoj bazi.

Da bi napravili run-time grešku daćemo pogrešan naziv tabele:

Neispravno:

```
objSqlTabela = new clsSqlTabela(objSqlKonekcija, "Raspored");
```

Ispravno:

```
objSqlTabela = new clsSqlTabela(objSqlKonekcija, "RasporedRada");
```

6.2. Pravila I EBNF prikaz gramatičko ispravnog programskog koda

PRAVILO: Evidencija uvek ima ; na kraju.

EBNF:

```
<naredba>::=<telo_naredbe>";"
```

PRAVILO: Procedura ima zagrade.

EBNF:

```
<procedura>::=<modifikator_pristupa><tip_naziv_procedure>("[<parameter>]")"
```

PRAVILO: String promenljiva uvek dobija string vrednost.

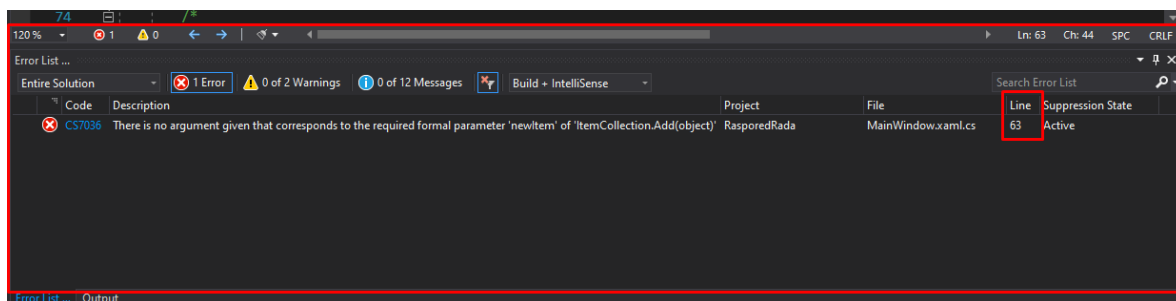
EBNF:

```
<string_promenljiva>::=" " "<vrednost_promenljive>" " "
```

7. Prikaz primene alata u detektovanju grešaka

7.1. Ekranski prikaz izveštaja kompajlera nad neispravnim kodom i nakon popravke

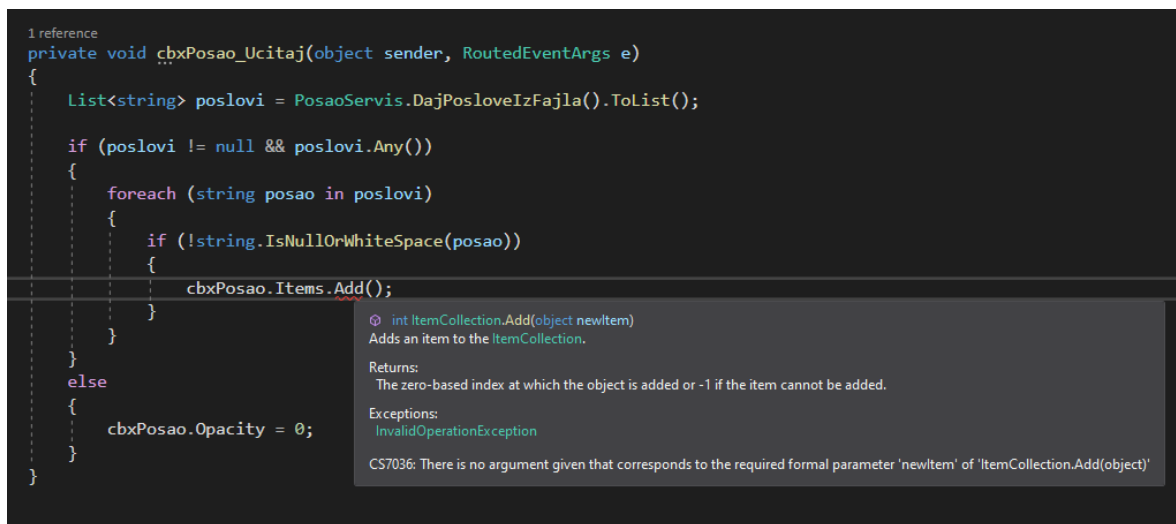
Prilikom pokretanja Build opcije iz Menu-a (*Build -> Build Solution*), dobijamo izveštaj kompajlera u okviru *Error List*, kao što je prikazano na slici 17.



Slika 17. Error List izveštaj

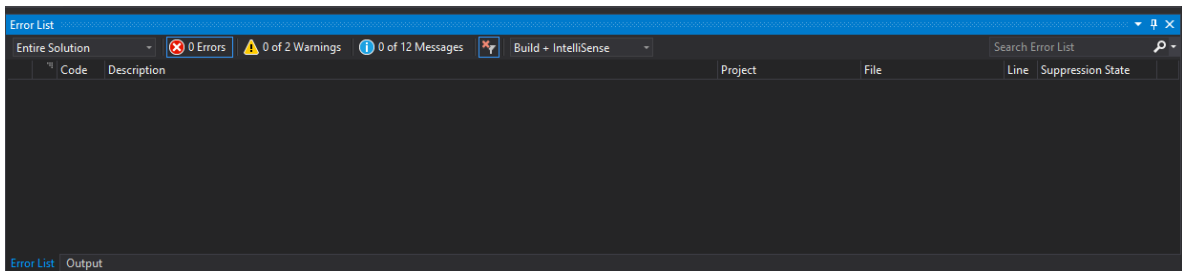
Error lista pored detaljnog opisa greške daje nam tačno mesto gde se greška pojavila, znači tačan naziv projekta, fajl u kome se javila i liniju koda. Time nam olakšava njeno traženje, a samim tim i njeno otklanjanje.

Postoji i još jedan način za pronalazak greške u kodu, a to je, da prilikom greške, kod će biti podvučen drugom bojom kako bi bio vidljiv da se tu nalazi greška. I ako s pređe mišem preko te greške, sam program nam nudi potencijalne ispravke. Pore toga, dobijamo detaljan opis greške (slika 18.).

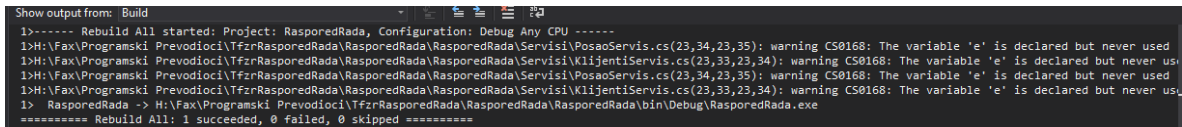


Slika 18. Mouse-over izveštaj greške

Nakon popravki grešaka i ponovnog pokušaja buildovanja projekta, dobijamo Izveštaj: Error = 0, Build succeeded, slika 19 i slika 20.



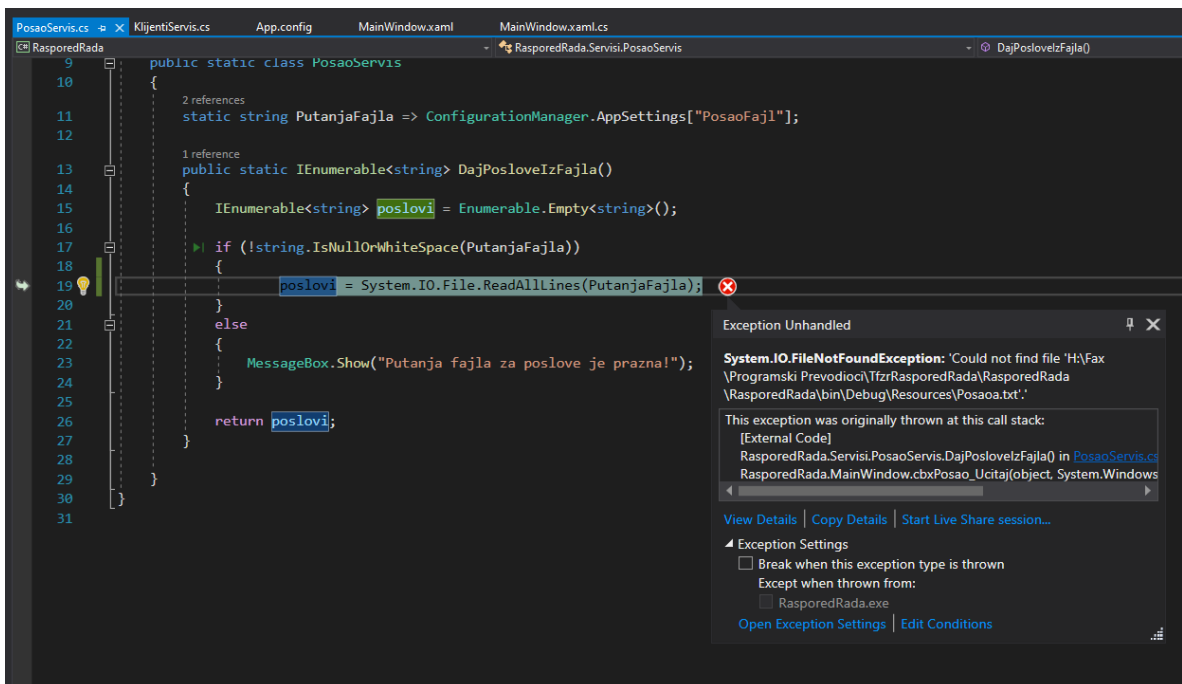
Slika 19. Error List izveštaj bez grešaka



Slika 20. Build proces koji je prošao bez grešaka

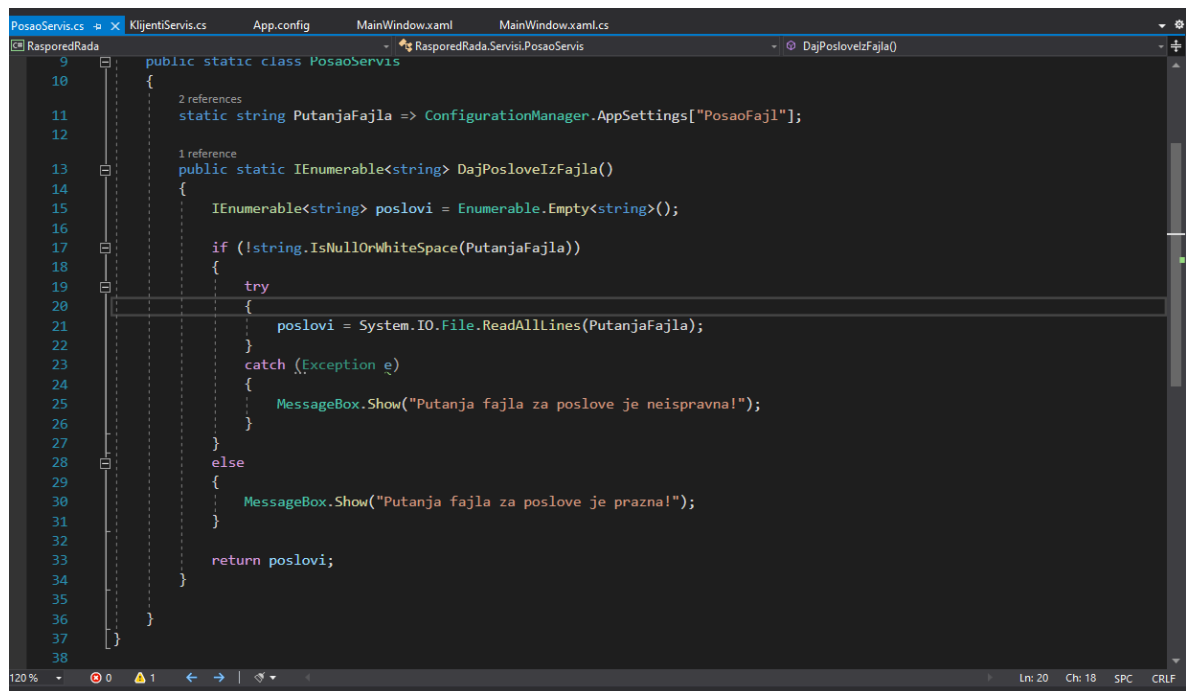
7.2. Ekranski prikaz reakcije na runtime greške

Greška se nalazi u nepravilnoj putanji do fajla za poslove, slika 21:



Slika 21. Primer runtime greške

Zato je ubačen try-catch blok koda kako bi se izbegla ova greška i korisniku dala adekvatna poruka pri pokretanju aplikacije (slika 22):



```
9 public static class PosaoServis
10 {
11     2 references
12     static string PutanjaFajla => ConfigurationManager.AppSettings["PosaoFajl1"];
13
14     1 reference
15     public static IEnumerable<string> DajPosloveIzFajla()
16     {
17         IEnumerable<string> poslovi = Enumerable.Empty<string>();
18
19         if (!string.IsNullOrEmpty(PutanjaFajla))
20         {
21             try
22             {
23                 poslovi = System.IO.File.ReadAllLines(PutanjaFajla);
24             }
25             catch (Exception e)
26             {
27                 MessageBox.Show("Putanja fajla za poslove je neispravna!");
28             }
29         }
30         else
31         {
32             MessageBox.Show("Putanja fajla za poslove je prazna!");
33         }
34
35         return poslovi;
36     }
37 }
38
```

Slika 22. Korišćenje try-catch bloka kako bi se izbegla run-time greška

8. Zaključak

Cilj ovog rada je da predstavi izradu Windows Presentation Foundation aplikacije u okruženju Microsoft Visual Studio 2019 Community Edition koristeći .NET framework, kao i povezivanje aplikacije sa SQLExpress bazom podataka. Za izradu baze korišćeni su Microsoft SQL Server i SQL Server Management Studio, međutim i sam Visual Studio ima opciju da se konektuje na SQL Server i unutar okruženja da se napravi baza, tako da SQL Server Management Studio nije neophodan. Poseban akcenat ovog rada je na objektno-orijentisanom programiranju.

Visual Studio ima opcije i za druge jezike koje rade sa .NET Framework-om, za ovu aplikaciju su korišćeni XAML za dizajn i C# za funkcionalnost.

Rad je rađen iz predmeta Programski Prevodioci, i kao takav sadrži primere grešaka koje nam prijavljuje kompajler (prevodilac), u radu su prikazani primeri grešaka i kako da se uklone, dok su u krajnjoj aplikaciji te greške već uklonjene.

Raspored rada (*eng, Timesheet*) je vrsta aplikacije koja se koristi u IT industriji, ali i u drugim sferama poslovanja kako bi se mogla voditi evidencija koliko vremena zaposleno lice provede na određenom zadatku. Iz ličnog iskustva ovo je bolje rešenje nego da se koristi *clocking* sistem kako bi se postarali da zaposleni rade svoje puno radno vreme.

Aplikacija bi se mogla doraditi tako što bi zaposleni, klijenti i poslovi odvojili u zasebne tabele unutar baze. Da se omoguće filteri po klijentima, ubaci kalendarski prikaz koliko je zaposleni kog dana radio i još mnoštvo drugih promena.

9. Literatura

[1] http://www.tfzr.uns.ac.rs/Content/files/0/99_PRAKTIKUM_sa13poglavlja.pdf

[2]

<http://www.tfzr.uns.ac.rs/Content/files/0/PripremniMaterijalZaKolokvijumDOPUNJENO3.pdf>

[3] https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio

[4] https://en.wikipedia.org/wiki/SQL_Server_Management_Studio

10. Listing

10.1. Listing korisničkog interfejsa

10.1.1. Dizajn MainWindow.xaml

```
<Window x:Class="RasporedRada.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:RasporedRada"
        mc:Ignorable="d"
        Title="Raspored Rada" Height="500" Width="699.745">
    <Grid>
        <DataGrid Name="DataGrid" HorizontalAlignment="Left" Height="200"
            Margin="10,259,10,10" VerticalAlignment="Top" AutoGenerateColumns="False">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding Ime}"
                    Header="Ime" Width="*"/>
                <DataGridTextColumn Binding="{Binding Prezime}"
                    Header="Prezime" Width="*"/>
                <DataGridTextColumn Binding="{Binding Klijent}"
                    Header="Klijent" Width="*"/>
                <DataGridTextColumn Binding="{Binding Posao}"
                    Header="Posao" Width="*"/>
                <DataGridTextColumn Binding="{Binding Datum}"
                    Header="Datum" Width="*"/>
                <DataGridTextColumn Binding="{Binding Sati}"
                    Header="Sati" Width="*"/>
            </DataGrid.Columns>
        </DataGrid>
        <Label Content="Ime" HorizontalAlignment="Left" Margin="32,34,0,0"
            VerticalAlignment="Top"/>
        <Label Content="Prezime" HorizontalAlignment="Left"
            Margin="10,63,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.519,1.154"/>
        <Label Content="Klijent" HorizontalAlignment="Left"
            Margin="19,93,0,0" VerticalAlignment="Top"/>
        <Label Content="Datum" HorizontalAlignment="Left"
            Margin="16,148,0,0" VerticalAlignment="Top"/>
        <Label Content="Sati" HorizontalAlignment="Left" Margin="33,176,0,0"
            VerticalAlignment="Top" RenderTransformOrigin="0.414,1.192"/>
        <TextBox x:Name="txtIme" HorizontalAlignment="Left" Height="23"
            Margin="67,37,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
        <TextBox x:Name="txtPrezime" HorizontalAlignment="Left" Height="23"
            Margin="67,66,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
        <TextBox x:Name="txtKlijent" HorizontalAlignment="Left" Height="23"
            Margin="67,94,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
        <TextBox x:Name="txtSati" HorizontalAlignment="Left" Height="23"
            Margin="67,176,0,0" TextWrapping="Wrap" PreviewTextInput="SamoBrojevi"
            VerticalAlignment="Top" Width="120"/>
        <ComboBox x:Name="cbxKlijenti" HorizontalAlignment="Left"
            Margin="192,93,0,0" VerticalAlignment="Top" Width="120"
            Loaded="cbxKlijenti_Ucitaj" Height="24"/>
        <DatePicker x:Name="dpDatum" HorizontalAlignment="Left"
            Margin="67,151,0,0" VerticalAlignment="Top" FirstDayOfWeek="Monday" Text="Odaberi
            Datum" Loaded="Datum_Ucitan" />
        <Button Content="Unesi" HorizontalAlignment="Left"
            Margin="67,234,0,0" VerticalAlignment="Top" Width="75" Click="unesi_Klik"/>
    </Grid>
</Window>
```

```

        <Button Content="Poništi" HorizontalAlignment="Left"
Margin="147,234,0,0" VerticalAlignment="Top" Width="75" Click="ponisti_Klik"/>
        <Button Content="Filter" HorizontalAlignment="Left"
Margin="399,218,0,0" VerticalAlignment="Top" Width="75" Click="filter_Klik"/>
        <TextBox x:Name="txtFilter" HorizontalAlignment="Left" Height="23"
Margin="399,190,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="229"/>
        <Label Content="Ime i prezime:" HorizontalAlignment="Left"
Margin="399,164,0,0" VerticalAlignment="Top"/>
        <Label Content="Posao" HorizontalAlignment="Left"
Margin="20,120,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.414,1.192"/>
        <ComboBox x:Name="cbxPosao" HorizontalAlignment="Left"
Margin="67,122,0,0" VerticalAlignment="Top" Width="120" Loaded="cbxPosao_Ucitaj"
Height="24"/>
        <Button Content="Resetuj" HorizontalAlignment="Left"
Margin="479,218,0,0" VerticalAlignment="Top" Width="75" Click="resetuj_Klik"/>
        <Image HorizontalAlignment="Left" Height="175" Margin="444,0,0,0"
VerticalAlignment="Top" Width="184" Source="Logo.PNG"/>

    </Grid>
</Window>

```

10.1.2. Funkcionalnost MainWindow.xaml.cs

```

using KlasePodataka;
using RaspoRedRada.Servisi;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace RaspoRedRada
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private string NazivBaze => ConfigurationManager.AppSettings["NazivBaze"];
        private string SqlInstanca =>
ConfigurationManager.AppSettings["SqlInstanca"];

        public MainWindow()
        {
            InitializeComponent();
            BinDataGrid();
        }
        private void BinDataGrid()
        {
            clsRaspoRedRadaDB raspoRedRadaDB = new clsRaspoRedRadaDB(SqlInstanca,
NazivBaze);
            DataGrid.ItemsSource =
raspoRedRadaDB.DajSveIzRaspoReda().Tables["RaspoRedRada"].DefaultView;
        }

        private void cbxKlijenti_Ucitaj(object sender, RoutedEventArgs e)

```

```

    {
        List<string> klijenti = KlijentiServis.DajKlijenteIzFajla().ToList();

        if (klijenti != null && klijenti.Any())
        {
            foreach (string klijent in klijenti)
            {
                if (!string.IsNullOrEmpty(klijent))
                {
                    cbxKlijenti.Items.Add(klijent);
                }
            }
        }
        else
        {
            cbxKlijenti.Opacity = 0;
        }
    }

    private void cbxPosao_Ucitaj(object sender, RoutedEventArgs e)
    {
        List<string> poslovi = PosaoServis.DajPosloveIzFajla().ToList();

        if (poslovi != null && poslovi.Any())
        {
            foreach (string posao in poslovi)
            {
                if (!string.IsNullOrEmpty(posao))
                {
                    cbxPosao.Items.Add(posao);
                }
            }
        }
        else
        {
            cbxPosao.Opacity = 0;
        }
    }

    /*
    * Metoda za menjanje default texta na Date Picker-u iz "Select a date" u
    "Odaberi datum" pri učitavanju elementa
    * Preuzeto sa: https://social.msdn.microsoft.com/Forums/sqlserver/en-US/9eec87e0-4d12-430d-83fd-ce13dd96776b/datepicker-hide-quotselect-datequot-placeholder-or-change-it?forum=wpf
    * Metode Datum_Ucitan i FindVisualChild
    */
    private void Datum_Ucitan(object sender, RoutedEventArgs e)
    {
        DatePicker datePicker = sender as DatePicker;
        if (datePicker != null)
        {
            System.Windows.Controls.Primitives.DatePickerTextBox
datePickerTextBox =
FindVisualChild<System.Windows.Controls.Primitives.DatePickerTextBox>(datePicker);
            if (datePickerTextBox != null)
            {

```

```

        ContentControl watermark =
datePickerTextBox.Template.FindName("PART_Watermark", datePickerTextBox) as
ContentControl;
        if (watermark != null)
        {
            watermark.Content = "Odaberi datum";
        }
    }
}

private T FindVisualChild<T>(DependencyObject dependencyObject) where T :
DependencyObject
{
    if (dependencyObject != null)
    {
        for (int i = 0; i <
VisualTreeHelper.GetChildrenCount(dependencyObject); i++)
        {
            DependencyObject child =
VisualTreeHelper.GetChild(dependencyObject, i);
            T result = (child as T) ?? FindVisualChild<T>(child);
            if (result != null)
            {
                return result;
            }
        }
    }
    return null;
}

private void filter_Klik(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(txtFilter.Text))
    {
        MessageBox.Show("Uneti parametar za filter");
    }

    if (txtFilter.Text.Split(' ').Length != 2)
    {
        MessageBox.Show("Nije uneto ime i prezime, razdvojeno jednim
razmakom");
    }
    else
    {
        clsRasporedRadaDB rasporedRadaDB = new
clsRasporedRadaDB(SqlInstanca, NazivBaze);
        DataGrid.ItemsSource =
rasporedRadaDB.DajRasporedPremaKorisniku(txtFilter.Text).Tables["RasporedRada"].De
faultView;
    }
}

private void resetuj_Klik(object sender, RoutedEventArgs e)
{

```

```

        PonistiFilter();
        BinDataGrid();
    }

    private void PonistiFilter()
    {
        txtFilter.Text = string.Empty;
    }

    private void ponisti_Klik(object sender, RoutedEventArgs e)
    {
        ponistiFormu();
    }

    private void ponistiFormu()
    {
        txtIme.Text = string.Empty;
        txtPrezime.Text = string.Empty;
        txtSati.Text = "0";
        txtKlijent.Text = string.Empty;
        cbxPosao.SelectedIndex = -1;
        cbxKlijenti.SelectedIndex = -1;
        dpDatum.SelectedDate = null;
    }

    private void unesi_Klik(object sender, RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty(txtIme.Text))
        {
            MessageBox.Show("Ime je prazno!");
            return;
        }

        if (string.IsNullOrEmpty(txtPrezime.Text))
        {
            MessageBox.Show("Prezime je prazno!");
        }

        if (string.IsNullOrEmpty(txtKlijent.Text))
        {
            if (cbxKlijenti.SelectedItem == null)
            {
                MessageBox.Show("Klijent nije odabran!");
            }
        }

        if (string.IsNullOrEmpty(txtSati.Text))
        {
            bool rezultat = double.TryParse(txtSati.Text, out double iSati);
            if (rezultat == false)
            {
                MessageBox.Show("Sati nisu uneti kao broj!");
                return;
            }

            if (iSati <= 0)
            {

```



```

        MessageBox.Show("Sati ne mogu biti manji ili jednaki 0");
    }
}

if (cbxPosao.SelectedItem == null)
{
    MessageBox.Show("Posao nije odabran!");
}

clsRasporedRada objRasporedRada = new clsRasporedRada();

string klijent = string.Empty;

if (!string.IsNullOrEmpty(txtKlijent.Text))
{
    klijent = txtKlijent.Text;
}
else
{
    klijent = cbxKlijenti.SelectedItem.ToString();
}

objRasporedRada.Ime = txtIme.Text;
objRasporedRada.Prezime = txtPrezime.Text;
objRasporedRada.Klijent = klijent;
objRasporedRada.Posao = cbxPosao.SelectedItem.ToString();
objRasporedRada.Sati = double.Parse(txtSati.Text);
objRasporedRada.Datum = (DateTime)dpDatum.SelectedDate;
clsRasporedRadaDB rasporedRadaDB = new clsRasporedRadaDB(SqlInstanca,
NazivBaze);
rasporedRadaDB.SnimiNoviUnosRada(objRasporedRada, out bool uspeh, out
string greska);

if (!uspeh)
{
    MessageBox.Show(greska);
}
else
{
    BinDataGrid();
    MessageBox.Show("Uspešno uneta stavka!");
    ponistiFormu();
}

}

/*
 * Provera i ograničenje za textbox za sate preko Regex-a kako bi se samo
decimalni brojevi mogli uneti
 */
private void SamoBrojevi(System.Object sender,
System.Windows.Input.TextCompositionEventArgs e)
{
    e.Handled = DaLiJeTekstBroj(e.Text);
}

```

```

        private static bool DaLiJeTekstBroj(string str)
        {
            System.Text.RegularExpressions.Regex reg = new
System.Text.RegularExpressions.Regex("[^0-9.-]+");
            return reg.IsMatch(str);
        }
    }
}

```

10.1.3. Konfiguracioni fajl App.config

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <appSettings>
        <add key="KlijentiFajl" value="Resources/Klijenti.txt"/>
        <add key="PosaoFajl" value="Resources/Posao.txt"/>
        <add key="NazivBaze" value="RasporedRada"/>
        <add key="SqlInstanca" value="SHADO\SQLEXPRESS"/>
    </appSettings>
</configuration>

```

10.1.4. Funkcionalnost KlijentiServis.cs

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Windows;

namespace RasporedRada.Servisi
{
    public static class KlijentiServis
    {
        static string PutanjaFajla =>
ConfigurationManager.AppSettings["KlijentiFajl"];

        public static IEnumerable<string> DajKlijenteIzFajla()
        {
            IEnumerable<string> klijenti = Enumerable.Empty<string>();

            if (!string.IsNullOrEmpty(PutanjaFajla))
            {
                try
                {
                    klijenti = System.IO.File.ReadAllLines(PutanjaFajla);
                }
                catch (Exception e)
                {
                }
            }
        }
    }
}

```

```

        MessageBox.Show("Putanja fajla za klijente neispravna!");
    }
}
else
{
    MessageBox.Show("Putanja fajla za klijente je prazna!");
}

return klijenti;
}
}
}

```

10.1.5. Funkcionalnost PosaoServis.cs

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Windows;

namespace RaspoRedRada.Servisi
{
    public static class PosaoServis
    {
        static string PutanjaFajla =>
        ConfigurationManager.AppSettings["PosaoFajl"];

        public static IEnumerable<string> DajPosloveIzFajla()
        {
            IEnumerable<string> poslovi = Enumerable.Empty<string>();

            if (!string.IsNullOrEmpty(PutanjaFajla))
            {
                try
                {
                    poslovi = System.IO.File.ReadAllLines(PutanjaFajla);
                }
                catch (Exception e)
                {
                    MessageBox.Show("Putanja fajla za poslove je neispravna!");
                }
            }
            else
            {
                MessageBox.Show("Putanja fajla za poslove je prazna!");
            }

            return poslovi;
        }
    }
}

```

10.2. Listing biblioteke KlasaPodataka

10.2.1. Model clsRaspredRada.cs

```
using System;

namespace KlasaPodataka
{
    public class clsRasporedRada
    {
        // NAMENA: Klasa ciji objekat koja odgovara jednom zapisu iz baze
        // podataka, ima samo set-get metode

        // atributi
        private string _Ime;
        private string _Prezime;
        private string _Klijent;
        private string _Posao;
        private double _Sati;
        private DateTime _Datum;

        // konstruktor
        public clsRasporedRada()
        {
            // inicijalizacija atributa
            _Ime = string.Empty;
            _Prezime = string.Empty;
            _Klijent = string.Empty;
            _Posao = string.Empty;
            _Sati = 0;
            _Datum = DateTime.Now;
        }

        // public
        public string Ime
        {
            get { return _Ime; }
            set { _Ime = value; }
        }

        public string Prezime
        {
            get { return _Prezime; }
            set { _Prezime = value; }
        }

        public string Klijent
        {
            get { return _Klijent; }
            set { _Klijent = value; }
        }

        public string Posao
        {
            get { return _Posao; }
            set { _Posao = value; }
        }
    }
}
```

```

        public double Sati
        {
            get { return _Sati; }
            set { _Sati = value; }
        }

        public DateTime Datum
        {
            get { return _Datum; }
            set { _Datum = value; }
        }
    }
}

```

10.2.2. Funkcionalnost clsRasporedRadaDB.cs

```

using SqlDBUtils;
using System;
using System.Data;
namespace KlasePodataka
{
    public class clsRasporedRadaDB
    {
        //atributi
        private clsSqlKonekcija objSqlKonekcija;
        private clsSqlTabela objSqlTabela;
        private DataSet dsRasporedRada;

        //konstruktor
        public clsRasporedRadaDB(string MSSQLInstanca, string NazivBazePodataka)
        {
            objSqlKonekcija = new clsSqlKonekcija(MSSQLInstanca, "",
NazivBazePodataka);
            objSqlKonekcija.OtvoriKonekciju();
            objSqlTabela = new clsSqlTabela(objSqlKonekcija, "RasporedRada");
            dsRasporedRada = new DataSet();
        }

        public DataSet DajSveIzRasporeda()
        {
            dsRasporedRada = objSqlTabela.DajPodatke("SELECT * FROM
RasporedRada");
            return dsRasporedRada;
        }

        public DataSet DajRasporedPremaKorisniku(string ImeIPrezime)
        {
            //Razdvojiti ime i prezime u niz stringova na osnovu razmaka
            string[] imeIPrezime = ImeIPrezime.Split(' ');
            dsRasporedRada = objSqlTabela.DajPodatke($"SELECT * FROM RasporedRada
WHERE Ime = '{imeIPrezime[0]}' AND Prezime = '{imeIPrezime[1]}'");
            return dsRasporedRada;
        }

        public void SnimiNoviUnosRada(clsRasporedRada objRasporedRada, out bool
uspeh, out string tekstGreske)
        {

```

```

        string strDatum =
        $"{objRasporedRada.Datum.Month}/{objRasporedRada.Datum.Day}/{objRasporedRada.Datum
        .Year}";

        string sqlInsertUpit = $"INSERT INTO RasporedRada
        VALUES('{objRasporedRada.Ime}', '{objRasporedRada.Prezime}', '{objRasporedRada.Klije
        nt}', '{objRasporedRada.Sati}', '{strDatum}', '{objRasporedRada.Posao}')";

        try
        {
            uspeh = objSqlTabela.IzvrsiAzuriranje(sqlInsertUpit);
            tekstGreske = string.Empty;
        }
        catch (Exception greska)
        {
            uspeh = false;
            tekstGreske = greska.Message;
        }
    }
}

```

10.3. Listing biblioteke SQLDBUtils

10.3.1. clsSqlKonekcija

```

//
using System.Data.SqlClient;

namespace SqlDBUtils
{
    public class clsSqlKonekcija
    {
        /* ODGOVORNOST: Konekcija na celinu baze podataka, SQL server tipa */

        #region Atributi
        private SqlConnection pKonekcija;
        //
        private string pPutanjaSQLBaze;
        private string pNazivBaze;
        private string pNazivSQL_DBMSInstance;
        private string pStringKonekcije;

        #endregion

        #region Konstruktor
        public clsSqlKonekcija(string nazivSQL_DBMSInstance, string
        putanjaSqlBaze, string NazivBaze)
        {
            // preuzimanje vrednosti u privatne attribute
            pPutanjaSQLBaze = putanjaSqlBaze;
            pNazivBaze = NazivBaze;
            pNazivSQL_DBMSInstance = nazivSQL_DBMSInstance;
            // pripremanje stringa konekcije, dodato 22.3.2020.
            pStringKonekcije = "";
            pStringKonekcije = FormirajStringKonekcije();
        }
    }
}

```

```

        // preklapajući konstruktor, kada dobijemo spolja string konekcije
        public clsSqlKonekcija(string noviStringKonekcije)
        {
            pStringKonekcije = noviStringKonekcije; // u ovom slucaju se ne poziva
metoda FormirajStringKonekcije
        }

        // preklapajući konstruktor, bez stringa konekcije
        public clsSqlKonekcija()
        {

        }

        #endregion

        #region Privatne metode
        private string FormirajStringKonekcije() // promenjen naziv
        {
            string mStringKonekcije = "";
            if (pPutanjaSQLBaze.Length.Equals(0) || pPutanjaSQLBaze == null)
            {
                mStringKonekcije = "Data Source=" + pNazivSQL_DBMSInstance + "
;Initial Catalog=" + pNazivBaze + ";Integrated Security=True";
            }
            else
            {
                mStringKonekcije = "Data Source=.\\" + pNazivSQL_DBMSInstance +
";AttachDbFilename=" + pPutanjaSQLBaze + "\\" + pNazivBaze + ";Integrated
Security=True;Connect Timeout=30;User Instance=True";
            }
            return mStringKonekcije;
        }
        #endregion

        #region Javne metode

        public string StringKonekcije
        {
            get { return pStringKonekcije; }
            set { pStringKonekcije = value; }
        }

        public bool OtvoriKonekciju()
        {
            bool uspeh;
            pKonekcija = new SqlConnection();
            pKonekcija.ConnectionString = pStringKonekcije; // 22.3.2020. s
obzirom da se na konstruktoru formira i smesta string konekcije u
pStringKonekcije, onda se ovde samo otvara
            // ovo je bolje resenje nego da se ovde pozove i kreiranje stringa
konekcije, s obzirom na SOLID princip Single Responsibility i a sto jezgrovitije
radimo posao koji je dat (da se ne radi ono sto nije u nazivu)

            try
            {
                pKonekcija.Open();
            }
            catch { }
        }
    }
}

```

```

        uspeh = true;
    }
    catch
    {
        uspeh = false;
    }
    return uspeh;
}

public SqlConnection DajKonekciju()
{
    return pKonekcija;
}

public string DajStringKonekcije() // dodato 22.3.2020.
{
    return pStringKonekcije;
}

public void ZatvoriKonekciju()
{
    pPutanjaSQLBaze = "";
    pKonekcija.Close();
    pKonekcija.Dispose();
}

#endregion

}
}

```

10.3.2. clsSqlTabela

```

using System.Collections.Generic;
//
using System.Data.SqlClient;
using System.Linq;

namespace SqlDBUtils
{
    public class clsSqlTabela
    {
        #region Atributi

        private string pNazivTabele;
        private clsSqlKonekcija pKonekcija;
        private SqlDataAdapter pAdapter;
        private System.Data.DataSet pDataSet;

        #endregion

        #region Konstruktor

        public clsSqlTabela(clsSqlKonekcija Konekcija, string NazivTabele)
        {
            pKonekcija = Konekcija;
            pNazivTabele = NazivTabele;
        }
    }
}

```



```

#endregion

#region Privatne metode

    private void KreirajAdapter(string SelectUpit, string InsertUpit, string
DeleteUpit, string UpdateUpit)
    {
        SqlCommand mSelectKomanda, mInsertKomanda, mDeleteKomanda,
mUpdateKomanda;

        mSelectKomanda = new SqlCommand();
        mSelectKomanda.CommandText = SelectUpit;
        mSelectKomanda.Connection = pKonekcija.DajKonekciju();

        mInsertKomanda = new SqlCommand();
        mInsertKomanda.CommandText = InsertUpit;
        mInsertKomanda.Connection = pKonekcija.DajKonekciju();

        mDeleteKomanda = new SqlCommand();
        mDeleteKomanda.CommandText = DeleteUpit;
        mDeleteKomanda.Connection = pKonekcija.DajKonekciju();

        mUpdateKomanda = new SqlCommand();
        mUpdateKomanda.CommandText = UpdateUpit;
        mUpdateKomanda.Connection = pKonekcija.DajKonekciju();

        pAdapter = new SqlDataAdapter();
        pAdapter.SelectCommand = mSelectKomanda;
        pAdapter.InsertCommand = mInsertKomanda;
        pAdapter.UpdateCommand = mUpdateKomanda;
        pAdapter.DeleteCommand = mDeleteKomanda;
    }

    private void KreirajDataset()
    {
        pDataSet = new System.Data.DataSet();
        pAdapter.Fill(pDataSet, pNazivTabele);
    }

    private System.Data.DataTable KreirajDataTable(System.Data.DataSet
noviDataSet)
    {
        return noviDataSet.Tables[0];
    }

    private void ZatvoriAdapterDataset()
    {
        pAdapter.Dispose();
        pDataSet.Dispose();
    }

#endregion

#region Javne metode

    public System.Data.DataSet DajPodatke(string SelectUpit)

```

```

// izdvaja podatke u odnosu na dat selectupit
{
    KreirajAdapter(SelectUpit, "", "", "");
    KreirajDataset();
    return pDataSet;
}

public int DajBrojSlogova()
{
    int BrojSlogova = pDataSet.Tables[0].Rows.Count;
    return BrojSlogova;
}

public bool IzvrsiAzuriranje(string Upit)
// izvrzava azuriranje unos/brisanje/izmena u odnosu na dati i upit
{
    //
    bool uspeh = false;
    SqlConnection mKonekcija;
    SqlCommand Komanda;
    SqlTransaction mTransakcija = null;
    try
    {
        mKonekcija = pKonekcija.DajKonekciju();
        // aktivan kod

        // povezivanje
        Komanda = new SqlCommand();
        Komanda.Connection = mKonekcija;
        Komanda = mKonekcija.CreateCommand();
        // pokretanje
        // NE TREBA OPEN JER DOBIJAMO OTVORENU KONEKCIJU KROZ KONSTRUKTOR
        // mKonekcija.Open();
        mTransakcija = mKonekcija.BeginTransaction();
        Komanda.Transaction = mTransakcija;
        Komanda.CommandText = Upit;
        Komanda.ExecuteNonQuery();
        mTransakcija.Commit();
        uspeh = true;
    }
    catch
    {
        mTransakcija.Rollback();
        uspeh = false;
    }
    return uspeh;
}

// preklapajuca metoda kada dobijemo vise upita da se izvrsi u transakciji
public bool IzvrsiAzuriranje(List<string> ListaUpita)
// izvrzava azuriranje unos/brisanje/izmena
// moze se dodeliti kao parametar lista od vise upita
// sada transakcija ima smisla, jer izvrsava vise upita u paketu
{
    //
    bool uspeh = false;

```

```

SqlConnection mKonekcija;
SqlCommand Komanda;
SqlTransaction mTransakcija = null;
try
{
    mKonekcija = pKonekcija.DajKonekciju();
    // aktivan kod

    // povezivanje
    Komanda = new SqlCommand();
    Komanda.Connection = mKonekcija;
    Komanda = mKonekcija.CreateCommand();
    // pokretanje
    // NE TREBA OPEN JER DOBIJAMO OTVORENU KONEKCIJU KROZ KONSTRUKTOR
    // mKonekcija.Open();
    string Upit = "";
    mTransakcija = mKonekcija.BeginTransaction();
    Komanda.Transaction = mTransakcija;
    for (int i = 0; i < ListaUpita.Count(); i++)
    {
        Upit = ListaUpita[i];
        Komanda.CommandText = Upit;
        Komanda.ExecuteNonQuery();
    }
    mTransakcija.Commit();
    uspeh = true;
}
catch
{
    mTransakcija.Rollback();
    uspeh = false;
}
return uspeh;
}

#endregion

}
}

```