5. Израда корисничког интерфејса: *Desktop* апликација (*Windows forms*) + рад са базама података (*SQL*)

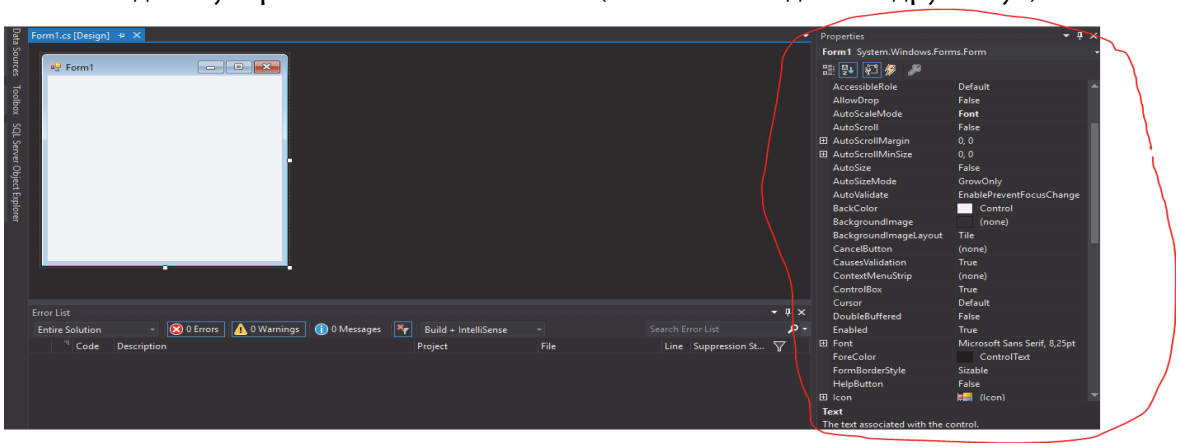
Desktop апликације у *.NET* окружењу су заправо *Windows Forms* апликације које се креирају за *Windows* оперативне системе са стандардним изгледом који функционише и на старијим верзијама система као што је *Windows 7* или чак и *Windows XP*. Поред класичних *Windows Forms* апликација постоје и неке новије као што су *WPF*[1] или пак *Windows Universal* које раде само на *Windows 10* оперативним системима (али и на *mobile Windows 10*).

5.1 *Windows Forms Application* - форме

У *Visual Studio* алату овај тип апликације се може креирати кликом на *New Project*, па под *Windows Classic Desktop* постоји *Windows Forms App*. Основне класе које се одмах користе када се направи пројекат јесу класа *Control* и класа *Form*. Заправо, када се креира пројекат креира се класа *Form1* која наслеђује системску класу *Form* и садржи сва њена својства и методе. Ту се налази и њен графички приказ за лакши визуелни рад. На ову класу се додају контроле које се наслеђују од системске класе *Control*.
Својства која се најчешће користе код класе *Form*:

- *Name* (назив инстанце класе *Form*)
- *Text* (текст који је исписан на насловној линији форме)
- *Left* и *Top* (координате положаја левог горњег угла форме)
- *BackColor* (боја позадине)
- *Width* (ширина)
- *Height* (висина)
- *Font* (фонт слова за текст)
- *MaximizeBox* (могућност максимизирања форме, *true-false*)
- *MinimizeBox* (могућност минимизирања форме, *true-false*)
- *MaximumSize* (маскимална могућа величина форме)
- *MinimumSize* (минимална могућа величина форме)
- …

Својства форме се могу видети и подесити и кроз *Properties* панел који се обично налази на десној страни *Visual Studio* алата (ако нисте подесили другачије).



[1] *Windows Presentation Foundation*

Поред својстава, у класи *Form* постоје и догађаји које можемо позивати када се дешавају неке акције корисника на форми. У оквиру догађаја дефинишемо шта ће се догодити када корисник изврши неку акцију (нпр. клик на дугме). Основни догађаји класе *Form*:

- *Load* (врло често се користи, реакција на учитавање форме)
- *MouseClick* (реакција на клик дугмета миша)
- *MouseDown*, *KeyDown* (реакција на притисак дугмета миша или тастатуре)
- *MouseUp*, *KeyUp* (реакција на отпуштање дугмета миша или тастатуре)
- *MouseMove* (реакција на померање миша)
- *Resize* (реакција на промену величине форме)
- …

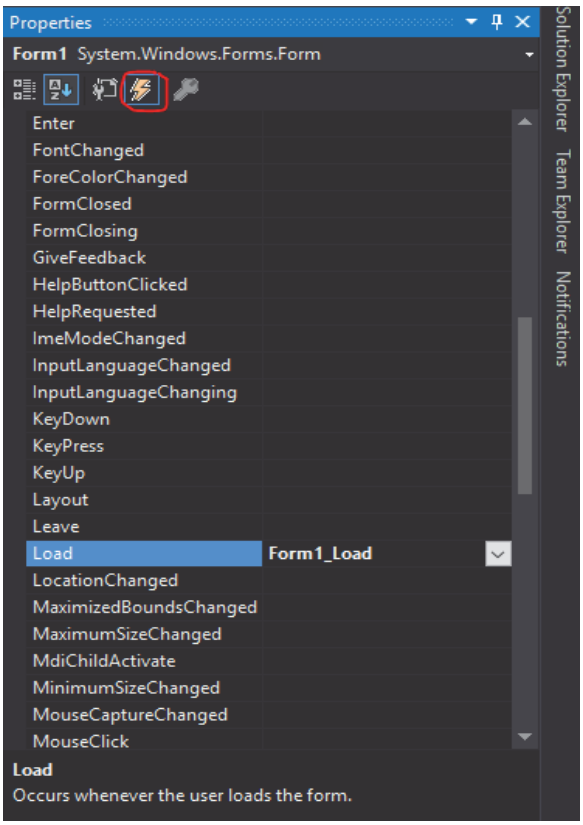
Подешавање ових догађаја се може вршити директно на форми (нпр. двоклик на форму креираће догађај *Load*), или такође у *Properties* прозору где се могу видети сви догађаји које форма пружа и то кликом на малу иконицу у облику муње:

У првој колони се приказују сви могући догађаји, а у другој они који су креирани у кôду.

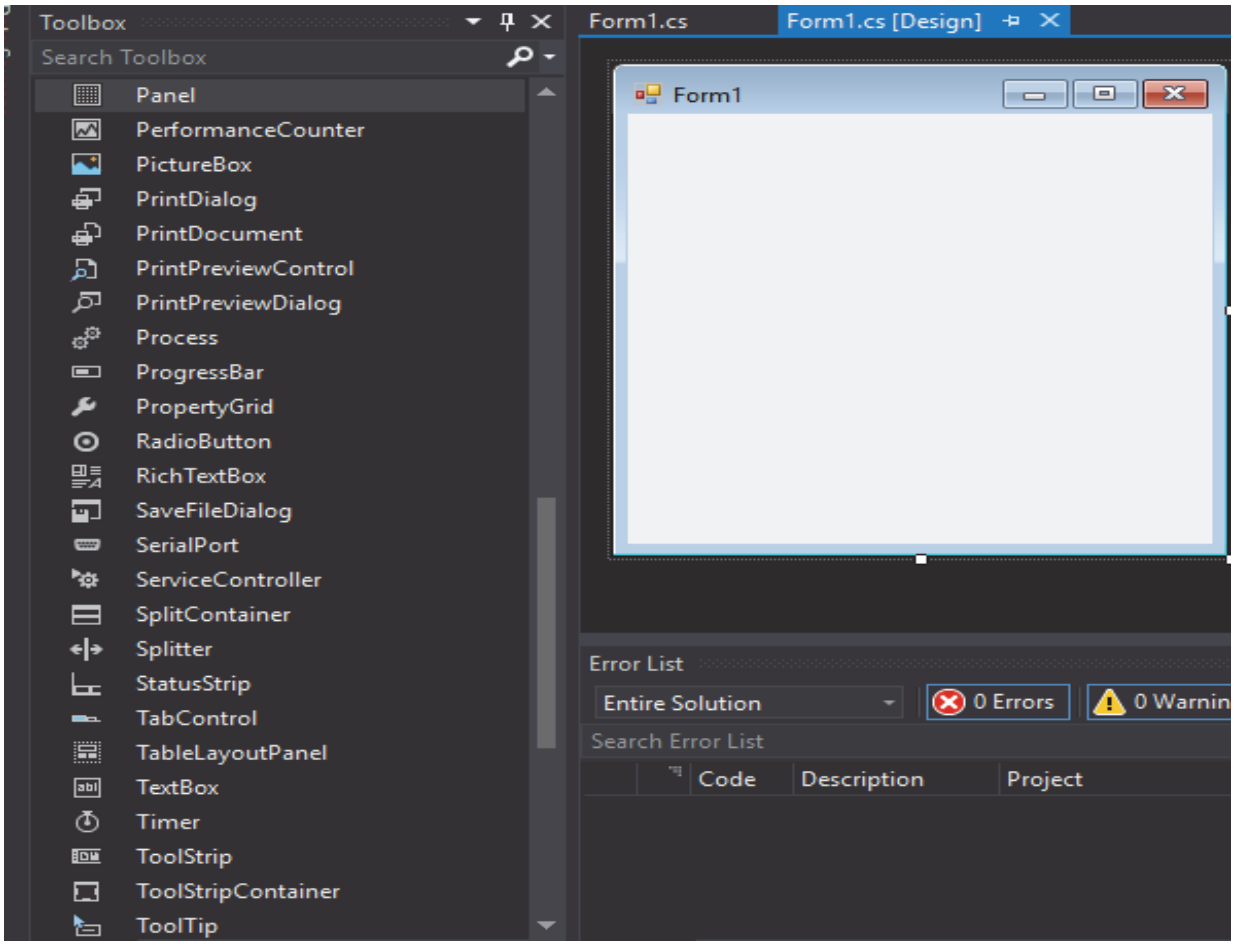
5.2 Контроле

.NET окружење пружа заиста огроман број контрола за рад са *Windows Forms* апликацијама. Најчешће коришћене:

- *Button* (дугме)
- *Label* (лабела у којој се може исписати неки текст)
- *TextBox* (поље за унос текста)
- *RadioButton* (избор једне од више могућих опција)
- *CheckBox* (избор ниједне, једне или више опција од већег броја могућих избора)
- *ListBox* (листа из које корисник може изабрати неку ставку)
- *ComboBox* (*drop-down* или падајућа листа из које корисник може изабрати ставку)
- *GroupBox* (груписање више контрола у једну целину)
- *DataGridView* (табеларни приказ података)
- *DateTimePicker* (избор датума)
- *PictureBox* (контрола за приказивање слике)
- *Panel* (груписање колекције контрола помоћу панела)
- *TabControl* (могућност креирања табова који приказују различите контроле)
- *TreeView* (“дрво” преглед, хијерахријски преглед лабела)
- …



Све контроле које пружа *.NET* окружење за *Windows Forms* апликације се могу видети у *Toolbox* прозору који се обично налази са леве стране (уколико корисник није подесио другачије). Контроле су груписане у неколико група ради лакшег проналажења а постоји и поље за претрагу.



Контроле се могу поставити на форму простим кликом на контролу која ће затим бити изабрана а затим клик на форму где желимо да поставимо контролу. Друга могућност је једноставно превлачење.

5.3 Својства и догађаји контрола

Као форме, и контроле имају своја својства и догађаје. У зависности од врсте контроле та својства и догађаји могу бити различити, али постоји одређена група која је заједничка за све контроле а неке од основних својствава су:

- *BackColor* (боја позадине контроле)
- *Bottom*, *Left*, *Right*, *Top* (растојање од одређене ивице прозора)
- *Enabled* (“укључена” или ”искључена” контрола, *true-false*)
- *Height* (висина контроле)
- *Width* (ширина контроле)
- *Name* (име контроле, за референцирање у кôду)
- *Text* (текст који је придружен контроли)
- *Visible* (видљивост контроле, *true-false*)

Најчешће коришћени догађаји:

- *Click* (реакција када се кликне на контролу)
- *Double Click* (реакција када се два пута кликне на контролу)
- *KeyDown*, *KeyPress*, *KeyUp* (реакција када је контрола у фокусу а када се притисне тастер на тастатури, ови догађаји се иницирају редом како су написани)
- *MouseDown*, *MouseMove*, *MouseUp* (реакције које се дешавају редом како су написани у зависности од акције са мишем)

Ова својства и догађаји се могу наћи у истом прозору где смо их налазили и за форму, *Properties*.

5.4 "Позивање" (приступање) контрола и других елемената

Када желимо да променимо вредност неке контроле, односно неко својство у оквиру програмског кôда (нпр. у оквиру догађаја *button click*) тој контроли можемо приступити преко њеног имена, нпр:

```
private void button1_Click(object sender, EventArgs e)
 {
     // promena vrednosti svojstva tekst klikom na dugme
     label1.Text = "Kliknuo sam na dugme"; // label1 je vrednost svojstva Name
 }
```

На исти начин се приступа било којој контроли која се налази на форми.

5.5 Навигација

За навигацију кроз *Windows Forms* апликацију се могу користити бројне контроле: *MenuStrip* (стандардан *Microsoft* мени), *TabControl* (креирање посебних табова), *ToolStrip* (нешто слично класичним *Toolbar* контролама) итд. Све ове контроле пружају могућност да се на истом делу прозора прикаже различит садржај у зависности од изабране опције. Понекад је за ове потребе потребно заправо отворити нову форму "преко" ове почетне, већ отворене. Следећи кораци су потребни за такву акцију:

- Прво је потребно креирати нову форму (десни клик на пројекат, *Add* > *Windows Form…*) и назвати је по жељи

Креираће се потпуно иста форма која је већ креирана када смо направили пројекат

- На одређени догађај у оквиру почетне форме (*menu item click*, *button click*, *label click…*) треба написати следећи кôд:

```
private void button1_Click(object sender, EventArgs e)
 {
     // u okviru nekog dogadjaja (u ovom slučaju klika na dugme)
     test newForm = new test(); // kreiranje instance klase nove forme (test je
              nova forma)
     newForm.Show(); // prikaz nove forme
 }
```

Када користимо методу *Show()* над инстанцом неке форме, отвара се нова форма али је могуће вратити се и на претходну (простим кликом на другу форму). Да би се ово спречило, може се користити и метода ***ShowDialog()*** која нам неће дозволити да пређемо на неку другу форму док тренутну не затворимо.

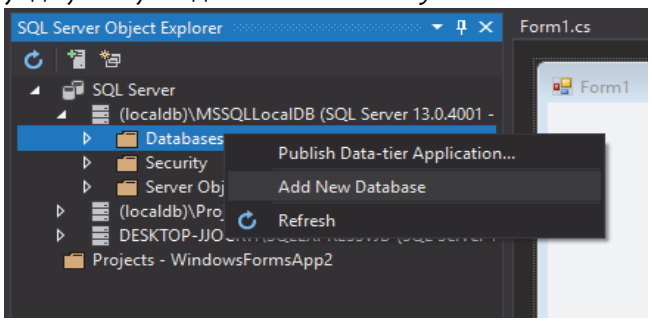
5.6 Локална база података (*LocalDB*)

Visual Studio развојно окружење поред рада са бројним програмским језицима пружа такође доста добру подршку и за рад са базама података. Приликом инсталације *VS*-а постоји могућност да се изабере и опција *LocalDB* која нам омогућава да креирамо локални сервер за базу података (*SQL Server*) који можемо користити приликом развоја апликација.

У *VS*-у, у главном менију, под *View* имамо опцију *SQL Server Object Explorer* која нам отвара прозор за преглед повезаних *SQL Server*-а, а један од њих је и *(localdb)* који можемо користити у било ком тренутку па чак и кад *VS* није отворен.

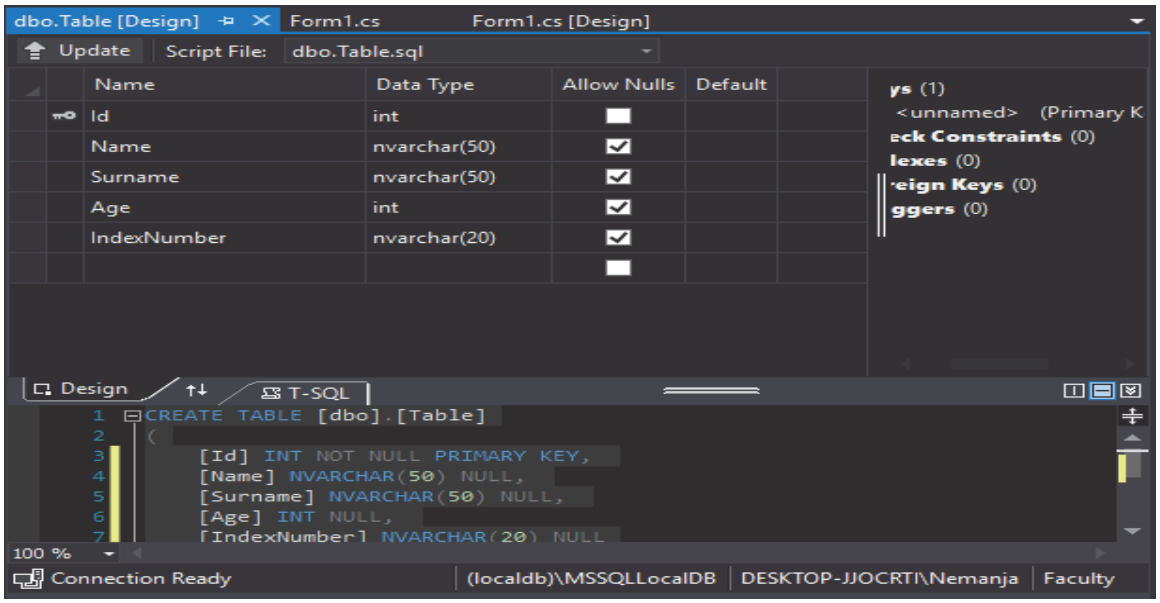
Нова база података у оквиру постојећих *SQL Server* инстанци (једна од *localdb* ставки) се може креирати десним кликом на *Databases* па *Add New Database*.

- Креираћемо једну базу под називом *Faculty*



Аутоматски се креирају подразумевани фолдери као што су *Tables*, *Views*, *Security*... Табеле у бази података се могу креирати десним кликом на *Tables* па *Add New Table*.. што нам отвара нови прозор (*Design*) у оквиру којег можемо "дизајнирати" табелу без употребе било каквог *SQL* кôда (о *SQL*-у мало више у следећем подпоглављу). У једном делу се чак може и видети какав се *SQL* у позадини креира.

- Креирати табелу *Students* са колонама: *Id*, *Name*, *Surname*, *Age*, *IndexNumber*



Кликом на *Update* (па *Update Database*) се може *publish*-овати табела на нашу локалну базу података.

5.7 *SQL* основе

Structured Query Language је стандардан језик за чување, манипулацију и добијање података из релационих база података као што је *SQL Server*. У *Visual Studio* алату, десним кликом на базу података па избором опције *New Query* отвориће се прозор где се могу вршити упити над базом података. Кликом на *Execute* (мало зелено дугме) може се извршити упид над базом. Постоји велики број упита који се могу користити у *SQL*-у, а најосновнији су:

- *SELECT* (користи се за селектовање података из базе података)

```
SELECT * FROM dbo.Students -- selektovanje svih kolona svih redova iz tabele Students

SELECT Name, Surname -- selektovanje kolona Name i Surname ..
FROM dbo.Students -- .. svih redova gde je..
WHERE Name LIKE 'Nemanja' -- .. ime Nemanja
ORDER BY Surname -- .. sortirano prema prezimenu
```

- *INSERT INTO* (унос података у табелу у бази података)

```
INSERT INTO dbo.Students (Id, Name, Surname, Age, IndexNumber) -- definisanje kolona
VALUES (1, 'Nemanja', 'Vićović', 25, '852/2015') -- definisanje vrednosti u koje unosimo
u kolone
```

- *UPDATE* (модификовање реда или редова у табели)

```
UPDATE dbo.Students
SET age = 26, IndexNumber = '852/2016'
WHERE IndexNumber LIKE '852/2015
```

- *DELETE* (брисање редова у табели)

```
DELETE FROM dbo.Students -- obriši redove iz tabele Students
WHERE Id = 1 -- gde je Id jednak vrednosti 1
```

5.8 Конекција са базом података (*ADO.NET*)

ADO.NET пружа конзистентан приступ изворима података као што је *SQL Server*. У оквиру *.NET*-а он заправо пружа бројне класе за конектовање и манипулацију података из базе података у нашу апликацију, односно *C#* програмски кôд.

Конекцију и прикупљање података из базе података можемо сместити у посебну класу (нпр. *DataConnection*).

Да бисмо се конектовали на неку базу података прво нам је потребан конекциони стринг:

```
string connString = "Integrated Security=true;Initial Catalog=Diskont;Data
Source=(localdb)\\MSSQLLocalDB";
```

Значења:

- *Integrated Security*: да ли је приступ за кориснике унутар мреже (активног директоријума)
- *Initial Catalog*: назив базе података са којом радимо
- *Data Source*: сервер, односно инстанца *SQL Server*-а коју користимо

Конекциони стринг се неретко чува и у апликационом конфигурационом фајлу а може да садржи и корисничко име и лозинку у ситуацију када се качимо на неку базу података која се налази на удаљеној локацији (нпр. на неком другом серверу).

Треба пратити следеће кораке приликом креирања конекције ка бази података и купљењу података у *ADO.NET* класу *SqlDataReader* из које касније можемо читати податке:

1. Креирати *sql* конекцију:

```
SqlConnection dataConnection = new SqlConnection(); // inicijalizacija konekcije ka bazi podataka
```

2. Поставити конекциони стринг конекције и отварање конекције

```
dataConnection.ConnectionString = connString;
dataConnection.Open();
```

3. Креирање *SQL* команде, постављање конекције команде и *SQL* упита:

```
SqlCommand command = new SqlCommand(); // kreiranje komande
command.Connection = this.dataConnection; //setovanje konekcije komande
command.CommandText = "SELECT * FROM Students"; // setovanje SQL upita koji će se izvršiti nad bazom podataka
```

4. Извршавање упита:

```
SqlDataReader dataReader = command.ExecuteReader();
```

Овде је битно нагласити да команда може бити извршена са различитим методама у зависности од упита који треба да или врати неке вредности, или изврши промене података у бази. Од коришћенијих ту су још *ExecuteQuery* метода која се обично користи за *INSERT*, *UPDATE* и *DELETE SQL* изразе али и *ExecuteScalar* која се такође користи за исте упите али враћа број редова на које је упит утицао (изменио неку вредност).

5. Купљење података из *SqlDataReader*-а који заправо садржи у себи табелу односно резултат који враћа неки упит:

```
while (dataReader.Read())
{
	Student student = new Student();
	// za svaki red se uzima vrednost određene kolone (0 - prva kolona)
	student.GetSetName = dataReader.GetString(1);
	student.surname = dataReader.GetString(2);
	student.age = dataReader.GetInt32(3);
	student.GetSetIndexNumber = dataReader.GetString(4);

	listToReturn.Add(student); // svaki student se na kraju može ubaciti u neku listu
}
```

Уколико резултат упита садржи само један ред, онда уместо *while* можемо користити и *if*.

6. Затварање конекције:

```
dataConnection.Close();
```

Добра пракса:

- Креирање конекције, команде и извршавање упита би требало да се налази у оквиру ***try*** блока а затим треба додати *finally* блок у оквиру којег ће се затворити конекција (*finally* се извршава без обзира да ли је *try* прошао или не)
- Друго решење за горњу ситуацију јесте коришћење израза *using*:

```
using (SqlConnection dataConnection = new SqlConnection(connString))
{ // ovde se zatim izvršavaju ostale operacije (kreiranje komande, izvršavanje
upita...)
} // ovde se ne zahteva zatvaranje konekcije već će se to automatski dogoditi
```

Додатак:

Постоји одређено поклапање између типова на *SQL Server* базама података и *.NET* типовима, нпр. *nvarchar* и *string* типови су они који се "поклапају" што значи да их можемо мапирати. Са друге стране, не можемо мапирати нпр. тип *nvarchar* у *int*. Више информација:

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-data-type-mappings