Úvod

Windows Presentation Foundation (dále již jenom WPF) je relativně nový způsob ke tvorbě vizuálních aplikací (GUI). Je velmi podobný Windows forms (se kterými již máte zkušenosti) a snaží se jejich problémy vyřešit, například:

- problém s responzivitou - aplikace byly stále stejné a velmi omezené
- možnost tvořit vlastní vstupy od uživatele
- různé baličky a designové prvky

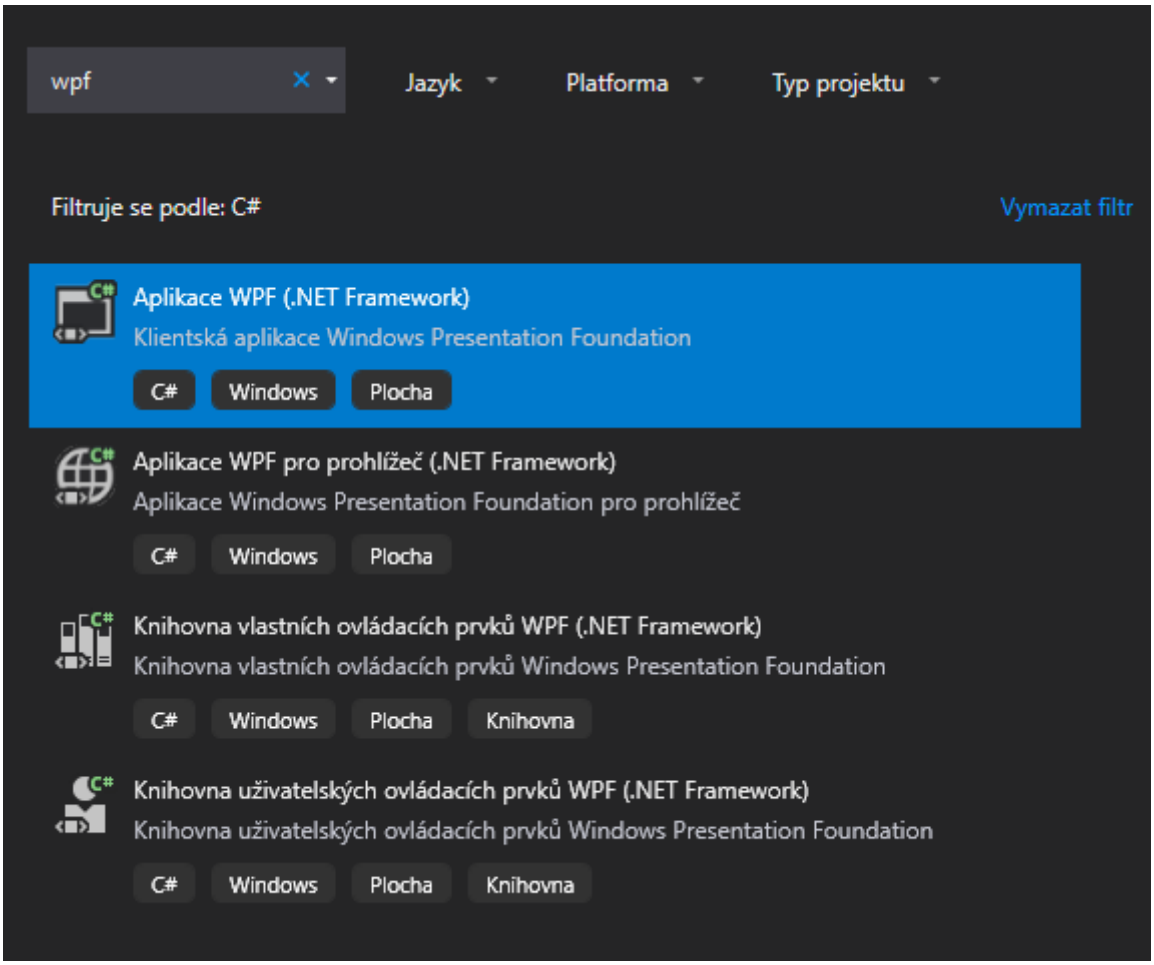
Stejně jako Forms je WPF omezen na platformu Windows a ani se nedají na jiných platformách vyvíjet. WPF si ukazujeme zejména z důvodů závěrečné práce, kde budete moci své aplikace ve WPF vytvářet.

Velmi velká výhoda WPF je to, že oproti Forms se nyní naše prostředí, které jsme si ve formuláři vytvořili, se negeneruje jako kód. Všechno nastavení, od pozic tlačítek až přes nastavení kliků, se ukládá do soborů typu XAML. To je velmi podobné XML a tedy i HTML, které znáte z WBA.

Při kompilaci projektu se nám stejně jako u konzolových aplikací vytváří exe soubor, který můžeme distribuovat mezi další lidi.

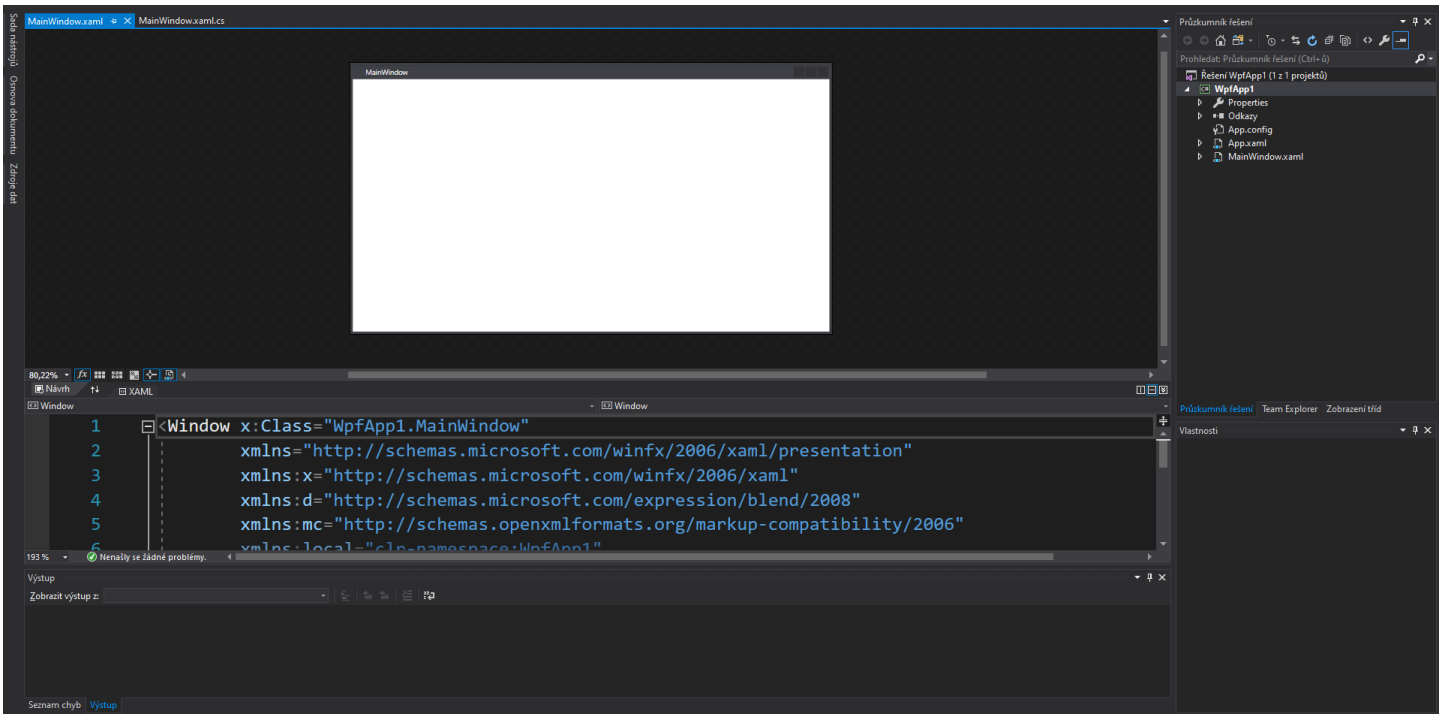
Založení projektu a prostředí

Při vybírání projektu máme možnost vybrat WPF z následujících možností. My vždy budeme vybírat možnost první:

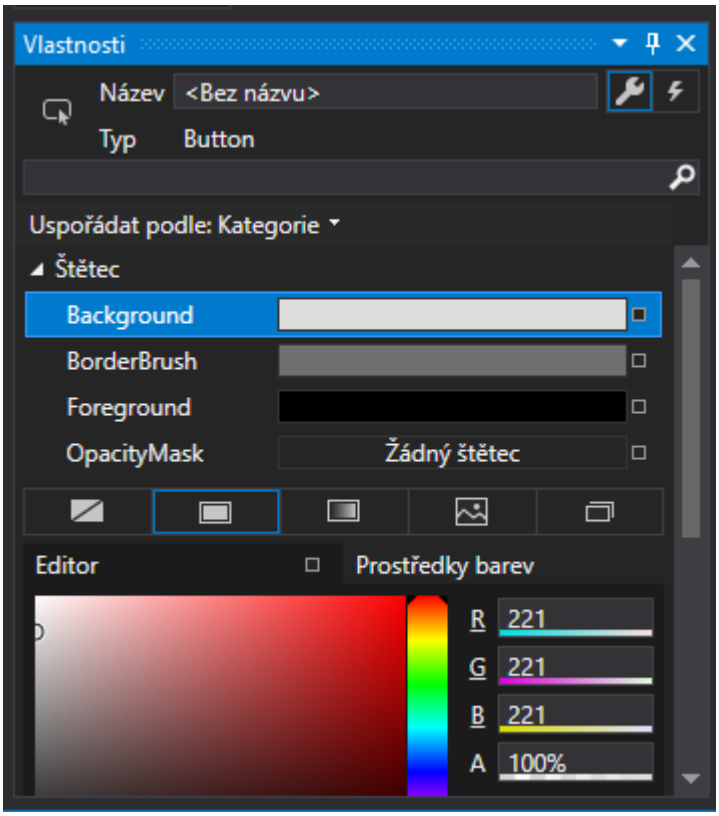


Zbytek můžeme vyplnit dle našich preferencí - jedná se o lokaci projektu a jeho jméno.

Po vytvoření projektu se nám zobrazí nové rozložení VS, které dovoluje tyto aplikace tvořit:



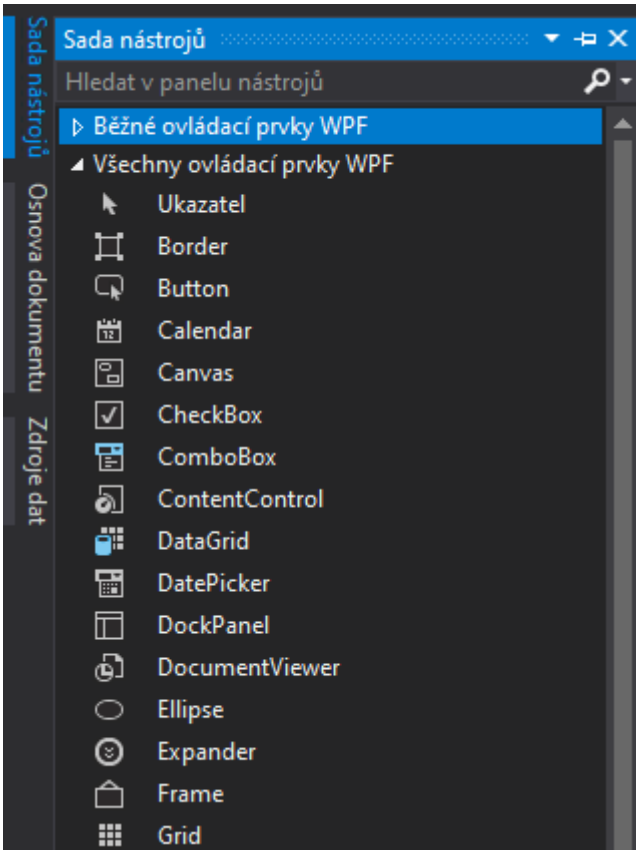
Jednotlivé části si nyní popíšeme. V pravé části obrazovky máme naše obvyklé věci - zobrazení souborů projektu. Vpravo dole máme vlastnosti. Ve vlastnostech se nám budou později zobrazovat vlastnosti věcí, které jsme přidali do projektu, např.:



V dolní části poté máme samotný XAML kód, který jsme si již vysvětlili, že je náš kód pro formulář. Najdeme tam různé definice, které budeme moci ručně psát místo toho, abychom měnili pozice a další věci přes vlastnosti či vizuální editor.

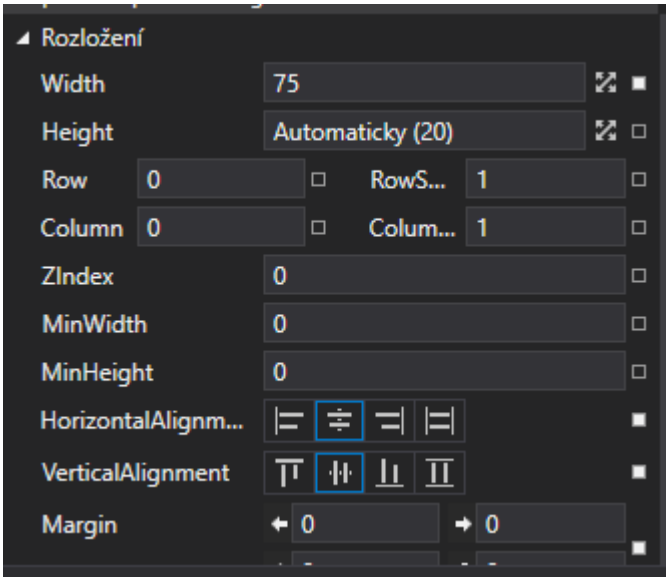
Zbytek obrazovky je vizuální editor, kde můžeme posouvat věci a určovat jim pozice a resp. jednotlivé prvky vybírat.

V levé části obrazovky máme ještě záložky. Nejvíce nás zajímá záložka Sada nástrojů (angl. toolbox). V této záložce nalezneme veškeré prvky, které můžeme vkládat do formulářů, např.:



Vlastnosti a události

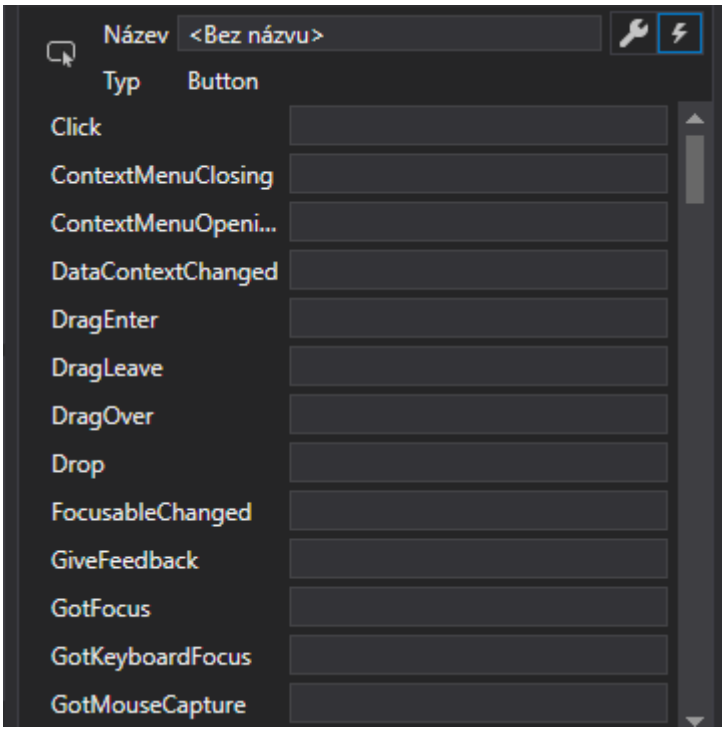
Jak jsme si již říkali, jednotlivým prvkům lze nastavovat vlastnosti pomocí záložka vlastností, případně uvnitř XAML. Tyto vlastnosti jsou uvnitř záložky většinou různě uskupené. Nejvíce nás bude zajímat rozložení, kde jednotlivým prvkům můžeme velmi jednoduše nastavovat, kde se mají nacházet uvnitř místa, které jim bylo vytvořeno:



Při modifikaci se nám vždy upraví XAML, po úpravách viz. výše se nám v XAML objeví:

```
<Grid>
    <Button Content="Button" Margin="0" VerticalAlignment="Center" Width="75" HorizontalAlign
</Grid>
```

Události jsou způsob jak uvnitř WPF můžeme předávat informace o tom, že se na prvku něco stalo. Např. když jsme klikli na tlačítko či když jsme přes něj přejeli myší. Události nastavujeme úplně stejně jako vlastnosti, musíme pouze v horní části vybrat blesk:

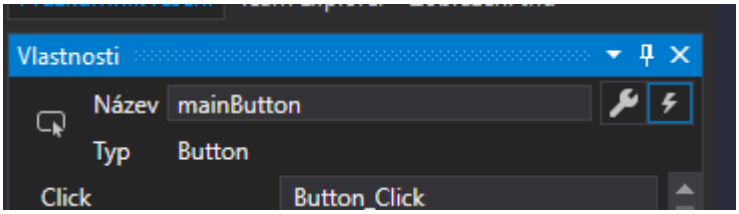


Při dvojkliku na nějakou událost se stane několik věcí:

- v XAML se prvek upraví tak, že se mu nastaví daná událost a předá se tam jméno funkce / metody, která se zavolá, když událost nastane
- vytvoří se daná funkce / metoda uvnitř dané třídy pro formulář
- daný soubor s funkcí / metodou se otevře

Uvnitř této metody poté můžeme dělat různé věci, např. modifikovat vlastnosti již vytvořený komponent.

Jednotlivé prvky ve formulář nejsou při vytvoření pojmenované. V záložce vlastnosti můžeme v horní části prvek přejmenovat a poté jej používat např. v kódu:



```
private void Button_Click(object sender, RoutedEventArgs e)
{
    this.mainButton.Content = "Klikl";
}
```

Dynamické přidávání objektů

Někdy máme ve WPF takové prvky formuláře, které by měli existovat až po nějaké době po vytvoření formuláře. Dynamické přidávání objektů je způsob, jak tvoříme jednotlivé prvky (textbox, label) za běhu programu. Jednotlivé prvky můžeme úplně stejně modifikovat (velikosti, barvy, rozložení, ...) již při vytvoření v kódu a poté je můžeme vložit do formuláře.

Kdybychom chtěli vytvořit tlačítko právě tehdy, když se načte formulář, v kódu bychom měli:

```
private void Window_Loaded(object sender, RoutedEventArgs e) // metodu musíme spojit v XAML
{
    Button button = new Button(); // vytvoř tlačítko
    button.Content = "Klikni"; // nastav mu text na klikni
    button.Width = 100; // nastav šířku
    button.Height = 100; // nastav výšku

    this.panel.Children.Add(button); // přidej do jiného prvku - panelu
}
```

Přidaným prvkům často chceme nastavovat i události. Události přidáme pomocí +=, kde ukazujeme na metodu, která daný kód zavolá:

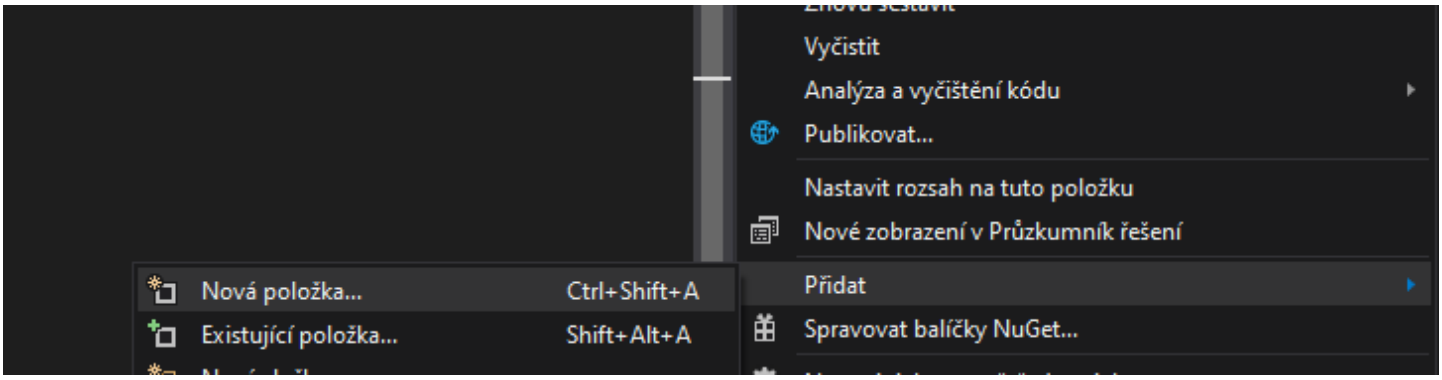
```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    Button button = new Button();
    button.Content = "Klikni";
    button.Width = 100;
    button.Height = 100;

    button.Click += Button_Click;
    button.MouseEnter += Button_MouseEnter;

    this.panel.Children.Add(button);
}
```

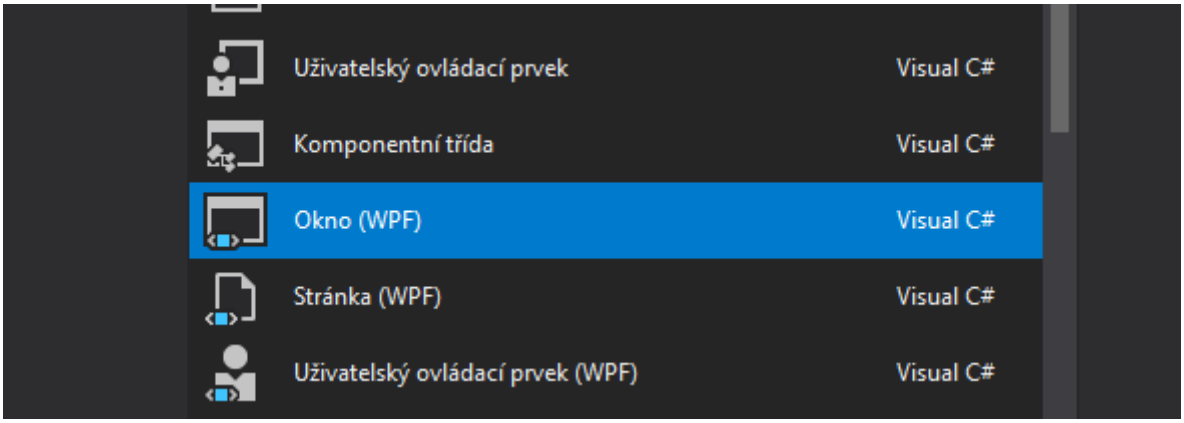
Více formulářových oken

Aplikace se často netvoří pouze z jednoho formuláře, ale z více, které se vzájemně otevírají a zavírají. Před otevřením nějakého formuláře se musí daný formulář založit, a to velmi podobně, jako např. struktura. Samotný formulář můžeme otevřít pomocí volání `.Show()`. V jeden moment může být otevřeno více formulářů.

Formulář zavřeme pomocí volání `.Close()`. Formulář zavřením ztratí veškeré zapsaná data.

Nový formulář založíme tím, že v průzkumníkovi projektu zaklikneme pravým tlačítkem myši a vybere Přidat > Nový > Okno. Tedy:





Jak tento formulář pojmenujeme, tak ho budeme muset používat v kódu - doporučuji tedy sémanticky dobře pojmenovaný formulář.

Daný formulář tedy můžeme otevřít pomocí:

```
Window1 newForm = new Window1();

newForm.Show();
```

Velmi často se používá taktéž otevření `.ShowDialog()`. Který formulář otevře, ale ten, který jej otevřel (tj. původní) bude uzamknutý (nepůjde používat) do té doby, než se nový uzavře.

Při vytvoření formuláře máme k dispozici všechny veřejné proměnné z daného formuláře a můžeme je mezi sebou vyměňovat. Tedy, když vytvoříme formulář, můžeme mu přidat hodnotu, kterou si zobrazí např. při načtení formuláře, např.:

```
public partial class Window1 : Window
{
    public string name; // < nase promenne, pozor na public!
    public List<ushort> grades;

    public Window1()
    {
        InitializeComponent();
    }

    public Window_Loaded() {
        this.nameLabel.Text = name;

        foreach(ushort grade in grades)
        {
            this.grades.Text += grade + ", ";
        }
    }
}
```

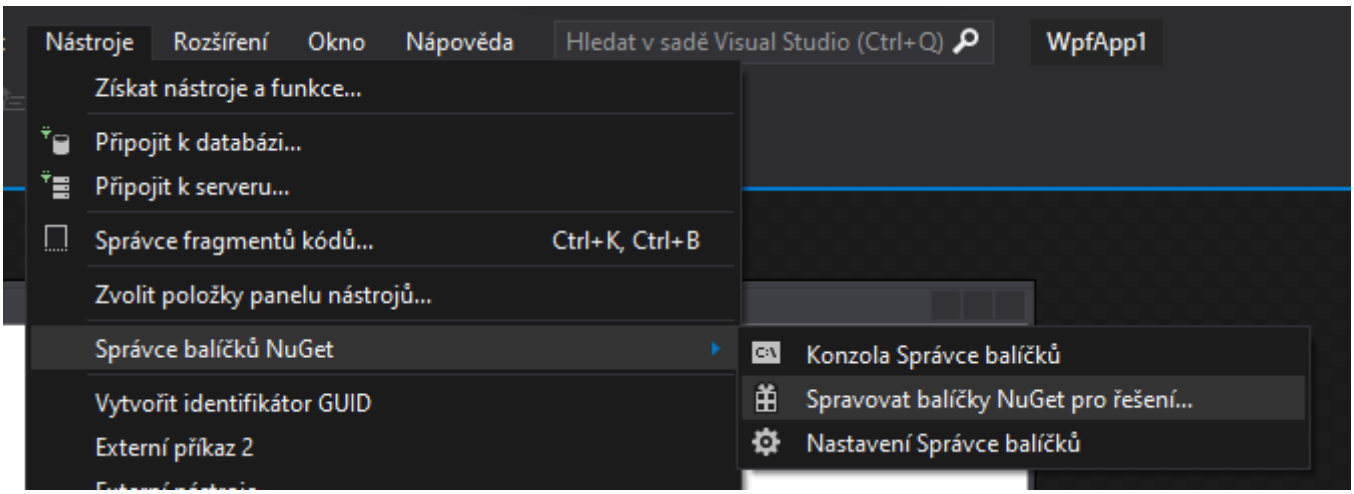
```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        Window1 newForm = new Window1();
```

```
            newForm.name = "test";
            newForm.grades = new List<ushort> {3, 2, 1};


            newForm.Show();
        }
    }
```

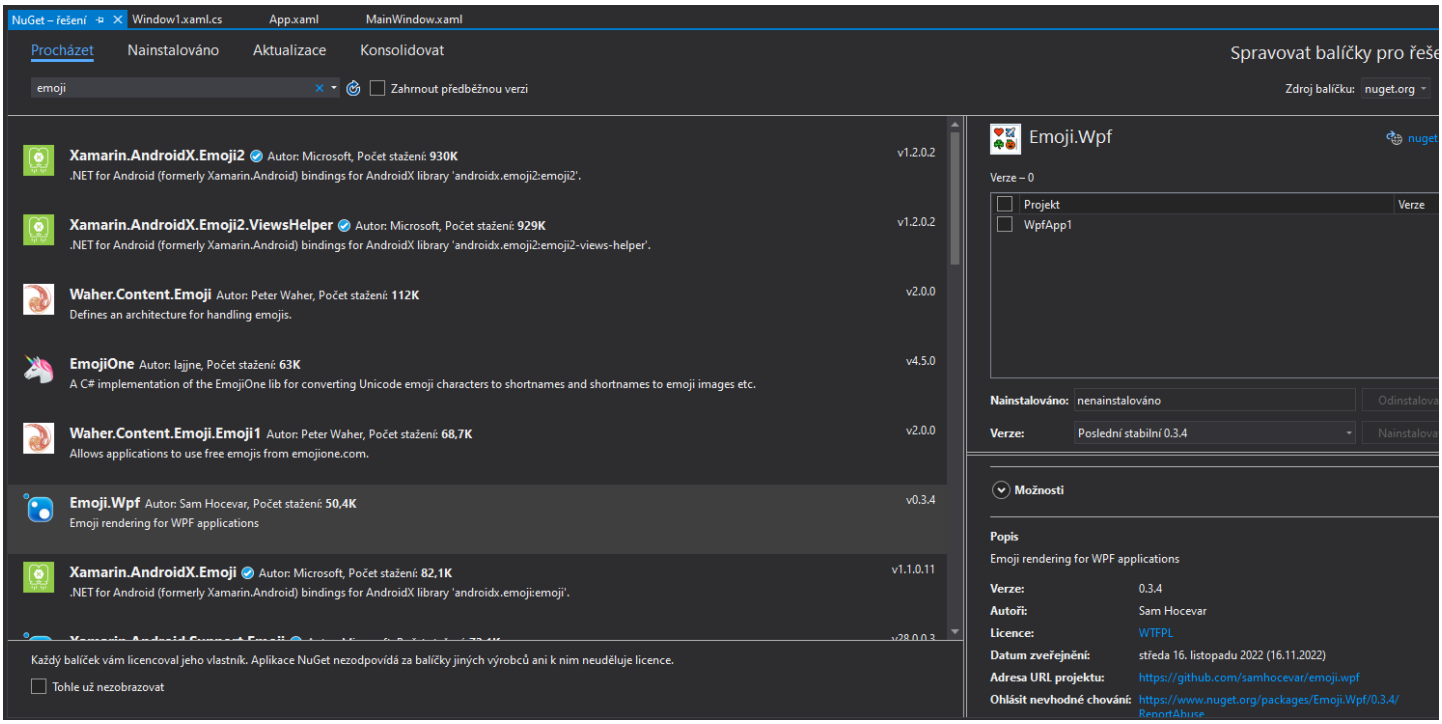
Knihovny

Uvnitř jakýchkoliv projektů napsaných C# máme možnost importovat a instalovat knihovny. Knihovny jsou nějaké části kódu od (často jiného) vývojáře, která nám ulehčují práci. Pro WPF se jedná např. o vlastní ovládací prvky či speciální UI.

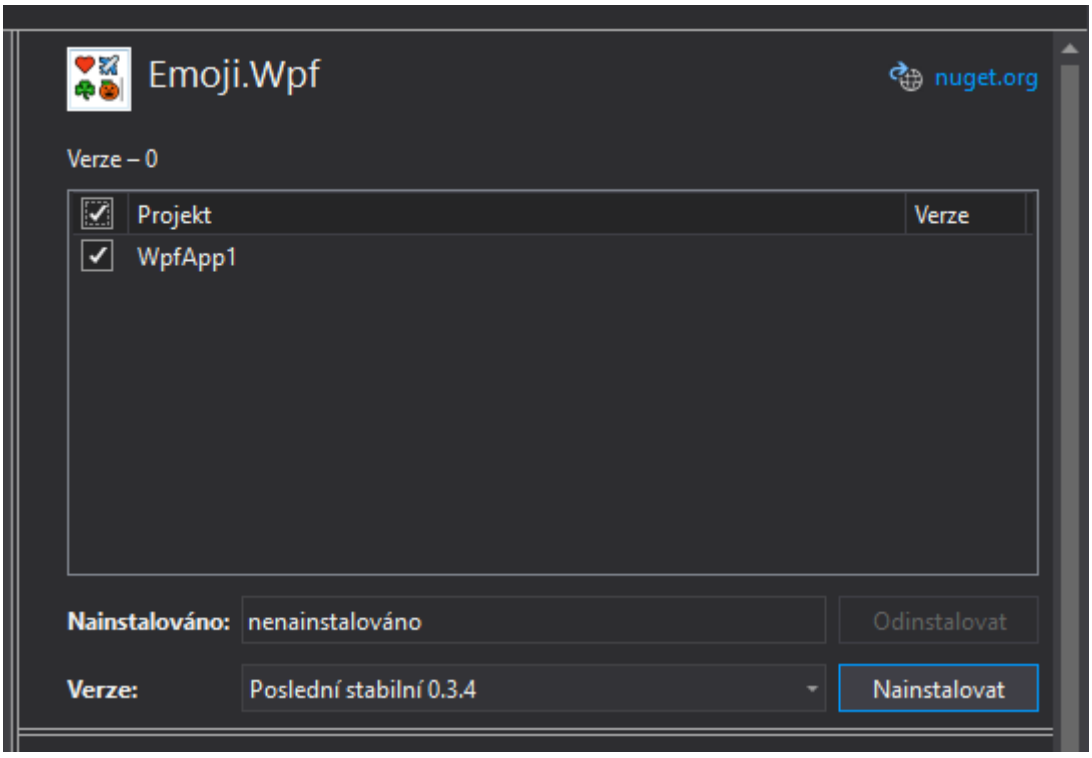
Balíčky pro C# instalujeme pomocí nuget. Nuget je balíčkovací systém, který nám balíčky automaticky spravuje. Nuget je k dispozici uvnitř VS:



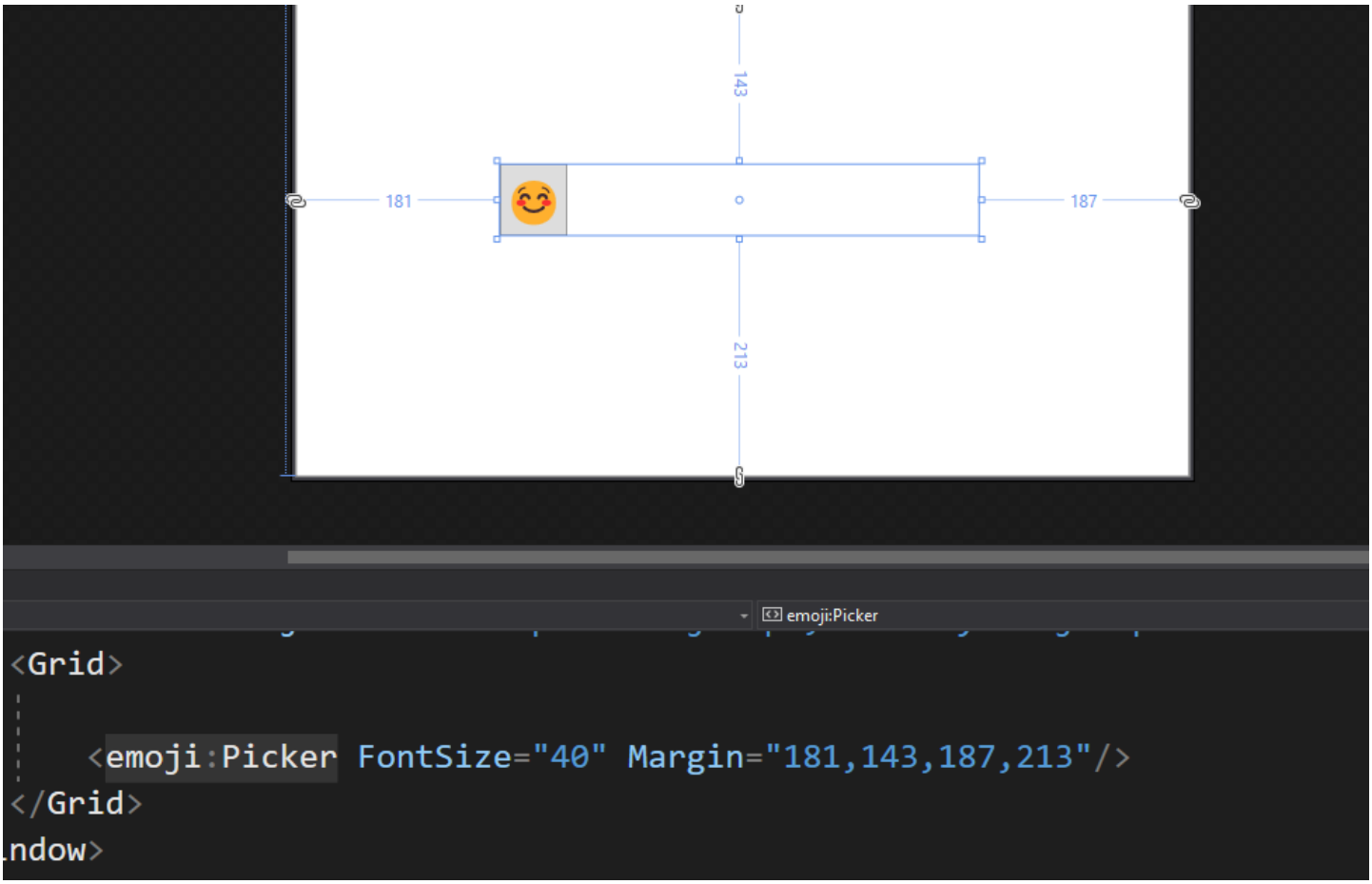
Při otevření tohoto nastavení se nám zobrazí okno s balíčky, kde můžeme balíčky vyhledávat:



Po vyhledání balíčku musíme vpravo označit náš projekt a verzi nainstalovat:



Po instalaci mám k dispozici dané prvky či daný kód, který můžeme dle dokumentace knihovny používat. Například výše nainstalované:



Základní design WPF není velmi moderní a proto je nutné často pro aplikace, které vidí uživatele a nejedná se přímo o pomůcky pro programátory, nutné používat nějaký design systém. Těchto systému existuje spousta a dokonce si můžete navrhnout i nějaký vlastní. Nejpoužívanější je Material Design od Google, který se používá skoro všude. Jedná se o přehledný design, který je komunitou přístupný i ve WPF.

Mezi další design systémy patří:

- https://github.com/handyOrg/HandyControl
- https://github.com/fluentribbon/Fluent.Ribbon
- https://github.com/benruehl/adonis-ui