Tvorba desktopovej aplikácie v jazyku C#

Úvod

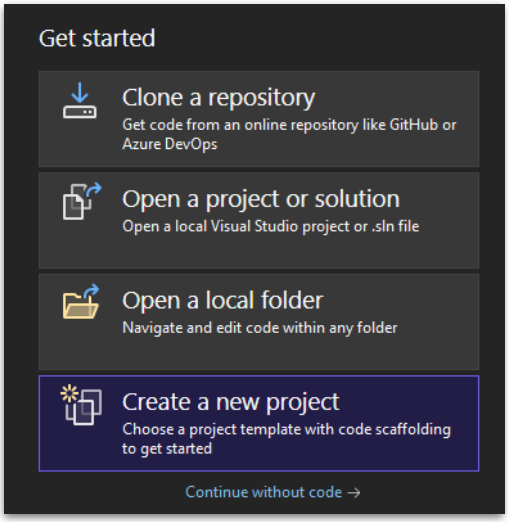
Sieťové programovanie umožňuje vytváranie množstva automatizačných nástrojov pre správu sieťových zariadení. Táto kapitola sa zaoberá ukážkou tvorby desktopovej aplikácie napísanej v jazyku C#, ktorá je schopná monitorovať sieťové zariadenia a do istej miery spravovať ich konfiguráciu. Manuál prevedie používateľov krokmi potrebnými k vytvoreniu používateľského rozhrania a vysvetlí prácu s modelovo riadeným prístupom správy konfigurácie využitím protokolu RESTCONF. Tento návod rovnako prevedie používateľov cez prácu so štandardnými protokolmi pre monitorovanie siete, akými sú protokoly SNMP a ICMP. Záverečná časť kapitoly sa venuje ukážkam práce so strojovým učením a spôsob jeho využitia v rámci kontroléra pre klasifikáciu dátovej prevádzky.

Požiadavky pre realizáciu úloh:

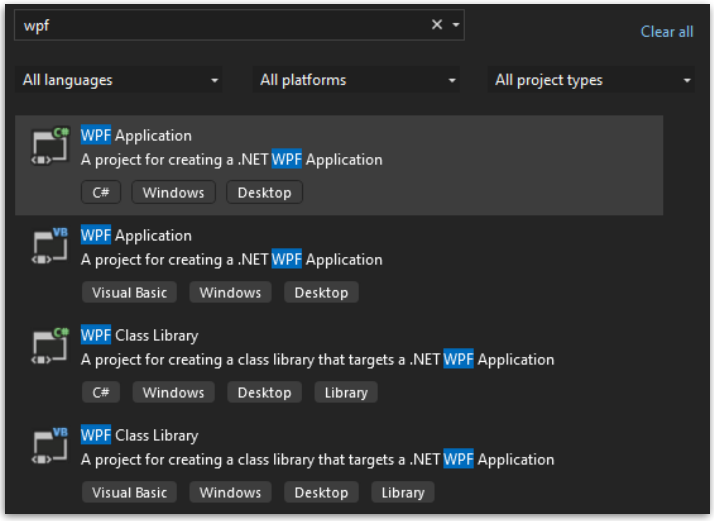
- Smerovač CSR1000v
- Visual Studio
- Nainštalovaný .NET

Ilustrácie v dokumente sa vzťahujú na použitie vývojového prostredia Visual Studio 2022. Dané úlohy je však možné riešiť aj v iných vývojových prostrediach, no ich výstupy sa nemusia presne zhodovať s výstupmi uvedenými v tomto dokumente.

Pred samotným riešením úloh súvisiacich so sieťovým programovaním a tvorbou kontroléra je najskôr potrebné vytvoriť nový projekt. Po nainštalovaní a spustení vývojového prostredia Visual Studio stačí zvoliť možnosť **Create a new project**. V nasledujúcom okne je následne možné v časti vyhľadávania zadať „*wpf*“, z vyhľadaných šablón zvoliť **WPF Application** a potvrdiť tlačidlom **Next**. Uvedené kroky sú zobrazené na obrázkoch 5.1 a 5.2:



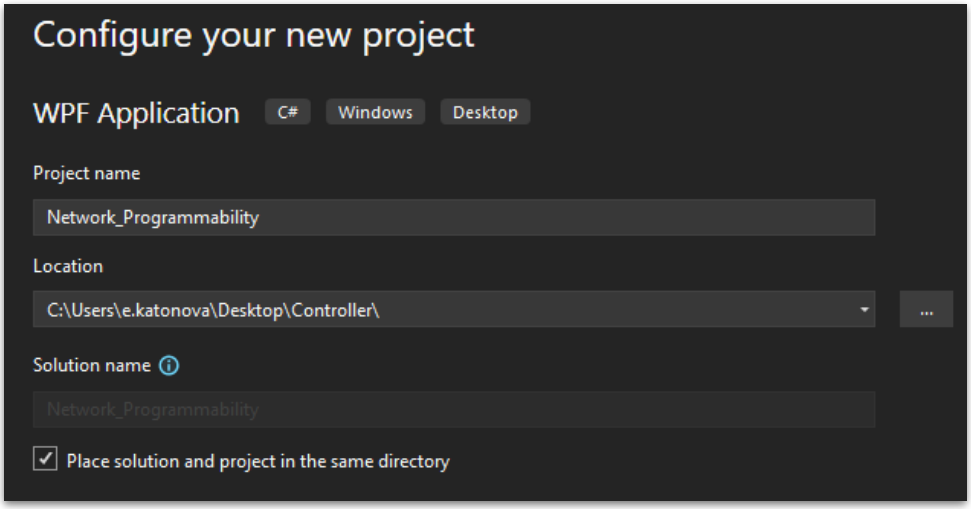
Obrázok 5.1 Vytvorenie projektu vo vývojovom prostredí Visual Studio



Obrázok 5.2 Zvolenie aplikačného rámca pre vývoj

V nasledujúcom okne je potrebné zvoliť názov projektu a jeho umiestnenie. To je následne potrebné znova potvrdiť tlačidlom **Next**, ako je možné vidieť na obrázku 5.3.

Posledným nastavením je voľba cieľového rámca. Projekt bol vyvíjaný vo verzii **.NET 5.0** preto je vhodné, pre funkčnosť kódu, zvoliť práve túto verziu a vytvoriť projekt stlačením tlačidla **Create**.



Obrázok 5.3 Finálne nastavenia pre vytvorenie projektu

Úloha 0: Oboznámenie sa s WPF

WPF (angl. *Windows Presentation Foundation*) je grafický rámec pre tvorbu desktopových aplikácií. Po vytvorení projektu sú automaticky vygenerované a otvorené dva súbory. Prvým je „*MainWindow.xaml*“, ktorý slúži na tvorbu používateľského okna a druhým je súbor „*MainWindow.xaml.cs*“, ktorý obsahuje C# kód definujúci funkcionalitu aplikácie. V rámci tejto práce je využitý „***event driven***“ programovací prístup. Tento prístup spočíva vo vykonaní akcie po vzniku udalosti. Napr. po stlačení tlačidla **Set hostname** sa spustí funkcia, ktorá zmení názov spravovaného zariadenia. V našom prípade bude udalosťou stlačenie tlačidla, po ktorej sa spustí príslušná obslužná funkcia. Štandardný kód pre hlavné okno aplikácie je zobrazený v nasledujúcom výpise:

```
<Window x:Class="Network_Programmability.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:Network_Programmability"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
    </Grid>
</Window>
```

Z uvedených údajov je možné zmeniť titulok, prípadne podľa potreby zmeniť šírku a výšku aplikácie. Vzhľad a rozloženie jednotlivých elementov aplikácie sa následne vkladá do tag-ov `<Grid></Grid>`. Grid nám umožňuje vytvárať pomyselnú mriežku, ktorá pomáha pri umiestňovaní elementov aplikácie, medzi ktoré patria tlačidlo, označenie, textové pole, menu a ďalšie. Elementy vytvárajúce mriežku je možné do seba vnárať a tým vytvárať podrobnejšie definície umiestnení. Tento prístup je analogický s rámcom Bootstrap, ktorý plní podobnú úlohu pri webových technológiách.

Ako je možné vidieť na nasledujúcich výpisoch, kód je veľmi jednoduchý a pozostáva z importovania základných modulov a triedy pre hlavné okno. Táto hlavná trieda obsahuje funkciu pre správu hlavného okna `MainWindow()`. Postupným riešením úloh budú pridávané nové funkcie práve do tejto hlavnej triedy.

Základný kód pre vytváranie funkcionality aplikácie znázorňuje nasledujúci výpis:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Network_Programmability
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();
      }
   }
}
```

Pre jednoduchosť a prehľadnosť najskôr upravíme vzhľad hlavného okna tak, aby obsahoval časť pre názov kontroléra, záložky pre jednotlivé úlohy a meno autora aplikácie. Vzorová ukážka kódu, ktorý je potrebné vložiť do súboru „*MainWindow.xaml*“ je uvedená v nasledujúcom výpise:

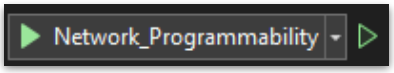
```
<Grid.ColumnDefinitions>
   <ColumnDefinition/>
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
   <RowDefinition Height="60px"/>
   <RowDefinition/>
   <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>

<Label Name="nazov_kontrolera" Content="NETWORK CONTROLLER" Grid.Row="0"
HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="40" Foreground="#4ebcff"
FontWeight="Bold"/>

<Label Name="Autor" Content="@Created by Erika Abigail Katonová" Grid.Row="2"
HorizontalAlignment="Center" Foreground="#2972b6"/>
<TabControl HorizontalAlignment="Center" Grid.Row="1">
   <TabItem Width="80" Header="GUI">
      <Grid>
      </Grid>
   </TabItem>
   <TabItem Width="100" Header="RESTCONF-GET">
      <Grid>
      </Grid>
   </TabItem>
   <TabItem Width="100" Header="RESTCONF-SET">
      <Grid>
      </Grid>
   </TabItem>
   <TabItem Width="100" Header="SNMP">
      <Grid>
      </Grid>
   </TabItem>
   <TabItem Width="100" Header="ICMP">
      <Grid>
      </Grid>
   </TabItem>
```

```
    <TabItem Width="110" Header="MachineLearning">
      <Grid>
      </Grid>
    </TabItem>
</TabControl>
```

Práca s `Grid` elementom vyžaduje definíciu počtu riadkov a stĺpcov v mriežke. V našom prípade bol vytvorený jeden stĺpec a tri riadky, pričom v prvom riadku bola jeho šírka nastavená staticky na hodnotu **60px**, druhý riadok nemá nastavený parameter šírky a posledný riadok sa prispôsobí veľkosti písma vloženého do daného riadka. Šírka druhého riadka bude automaticky prispôsobená zvyšku voľnej šírky pre celé okno.

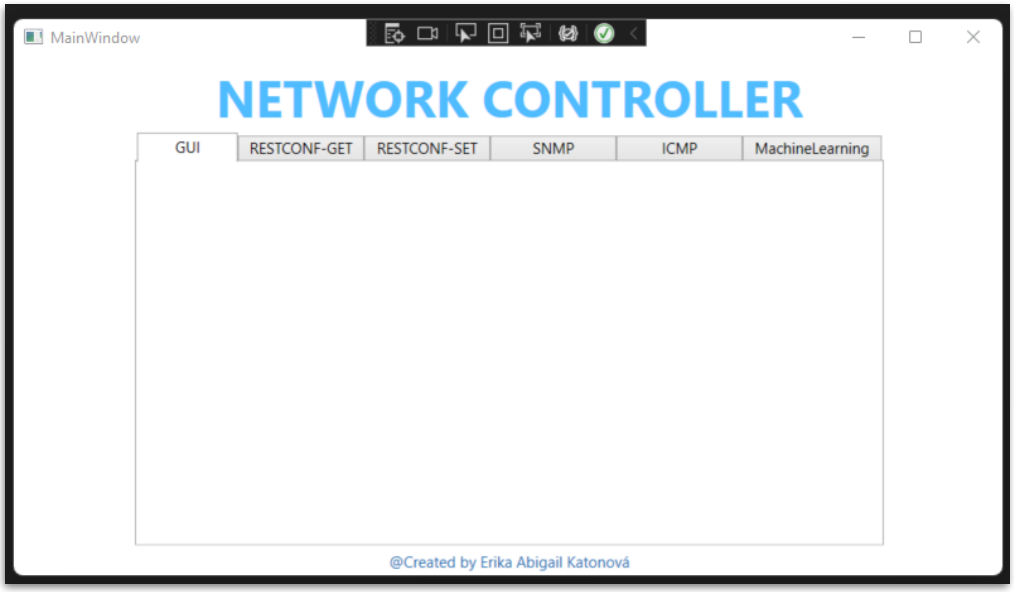
Element `Label` umožňuje pridávať do aplikácie text. Tento text je možné definovať atribútom `Content`. Atribút `Name` slúži na pomenovanie elementu pre neskoršiu prácu v kóde. `Grid.Row`/`Grid.Column` umožňujú definovať riadok a stĺpec, do ktorého chceme vložiť daný element. Atribút `Alignment` umožňuje nastaviť pozíciu elementu v rámci štvorca v mriežke. Atribút `FontSize` nastaví veľkosť písma, `FontWeight` štýl písma a `Foreground` farbu textu.

Element `TabControl` umožňuje vytvoriť klikateľné záložky, ktorých názov a obsah je definovaný vnoreným elementom `TabItem`. Pomenovať záložku je možné cez atribút `Header`. Väčšina elementov je párových, kde v prípade vkladania ďalších elementov do príslušnej vybranej sekcie stačí vkladať ďalšie elementy medzi začiatočný a koncový tag. Aktuálne, každá záložka obsahuje predpripravené elementy `Grid`, ktoré umožnia organizovať štruktúru okien pri riešení neskorších úloh. Po takto upravených kódoch je možné spustiť vytvorený kód stlačením tlačidla zobrazeného na obrázku 5.4:



Obrázok 5.4 Tlačidlo pre kompiláciu

Výsledný vzhľad vytvorenej desktopovej aplikácie je znázornený na obrázku 5.5:

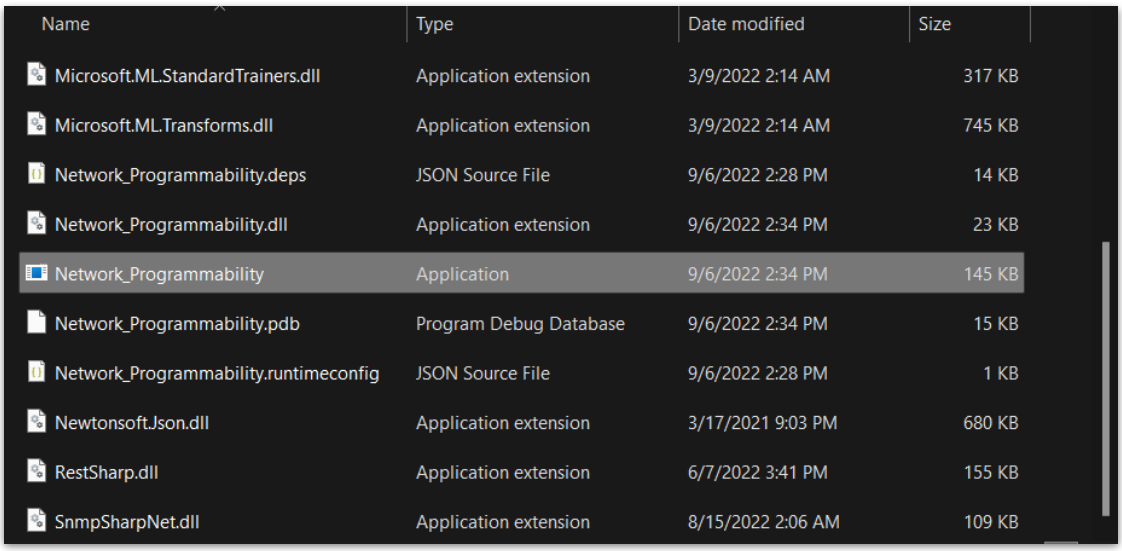


Obrázok 5.5 Grafické rozhranie aplikácie

Aktuálna verzia používateľského rozhrania umožňuje preklikávať sa medzi jednotlivými záložkami a zobraziť ich obsah. Keďže sú zatiaľ všetky záložky prázdne, tak sa zobrazuje len prázdne pole. Nasledujúce časti tohto dokumentu slúžia ako zbierka úloh, vyriešením ktorej sa používatelia postupne naučia vytvárať kontrolér pre správu sieťových zariadení využitím rôznych prístupov.

Aplikáciu je možné spustiť cez vytvorený spustiteľný súbor (na obrázku 5.6), ktorý je možné nájsť v:

Network_Programmability → bin → Debug → net5.0-windows → Network_Programmability



Name	Type	Date modified	Size
Microsoft.ML.StandardTrainers.dll	Application extension	3/9/2022 2:14 AM	317 KB
Microsoft.ML.Transforms.dll	Application extension	3/9/2022 2:14 AM	745 KB
Network_Programmability.deps	JSON Source File	9/6/2022 2:28 PM	14 KB
Network_Programmability.dll	Application extension	9/6/2022 2:34 PM	23 KB
Network_Programmability	Application	9/6/2022 2:34 PM	145 KB
Network_Programmability.pdb	Program Debug Database	9/6/2022 2:34 PM	15 KB
Network_Programmability.runtimeconfig	JSON Source File	9/6/2022 2:28 PM	1 KB
Newtonsoft.Json.dll	Application extension	3/17/2021 9:03 PM	680 KB
RestSharp.dll	Application extension	6/7/2022 3:41 PM	155 KB
SnmpSharpNet.dll	Application extension	8/15/2022 2:06 AM	109 KB

Obrázok 5.6 Umiestnenie spustiteľného súboru aplikácie

Úloha 1: Ukážka tvorby používateľského rozhrania

Zadanie: Podľa nasledujúcich požiadaviek vytvorte jednoduché používateľské rozhranie.

Rozdeľte plochu do dvoch stĺpcov:

a. V prvom stĺpci vytvorte dva riadky
   i. Prvý riadok nech obsahuje centrovaný text **`Insert Information`**.
   ii. Druhý riadok ďalej rozdeľte na dva riadky a dva stĺpce, pričom do prvého riadku a prvého stĺpca vložte text **`Insert text below:`**. Do prvého riadku a druhého stĺpca vložte pole pre vstup od používateľa. Na pozíciu druhého riadku a prvého stĺpca vložte tlačidlo s označením **`Display`**.

b. V druhom stĺpci vytvorte dva riadky
   i. Prvý riadok nech obsahuje centrovaný názov **`Display Output`**.
   ii. Druhý riadok nech obsahuje element, ktorý po stlačení tlačidla zobrazí text zadaný od používateľa.

Prvým krokom je rozdelenie celej plochy do dvoch stĺpcov, čo je možné definovať priamo v elemente `Grid`. Celý kód pre používateľské rozhranie sa bude aplikovať v rámci záložky pre prvú úlohu medzi hlavné `<Grid></Grid>` tagy, čo znázorňuje nasledujúci výpis:

```
<TabItem Width="80" Header="GUI">
   <Grid>
      ... Kód bude pridávaný do tejto sekcie ...
   </Grid>
</TabItem>
```

Rozdelenie plochy do dvoch stĺpcov je možné vykonať nasledujúcim kódom:

```
<Grid.ColumnDefinitions>
   <ColumnDefinition/>
   <ColumnDefinition/>
</Grid.ColumnDefinitions>
```

V tejto chvíli je možné s oboma stĺpcami pracovať oddelene. Začnime prácou s prvým stĺpcom, do ktorého je možné vložiť novú mriežku. Mriežkou vytvoríme dvojicu riadkov a priamo do prvého riadka vložíme požadovaný text **`Insert Information`**. Vzorový kód je znázornený v nasledujúcom výpise:

```
<Grid Grid.Column="0">
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition/>
   </Grid.RowDefinitions>

   <Label Name="Gui_Left_Column" Content="Insert Information" Grid.Row="0"
HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="20" Foreground="#4ebcff"
FontWeight="Bold"/>
</Grid>
```

Ďalej je možné rozdeliť druhý riadok na dva stĺpce a dva riadky. Do bunky v prvom riadku a prvom stĺpci následne vložíme element `Label`, v ktorom je potrebné nastaviť atribút `Content`, definovať súradnice pozície cez atribúty `Grid.Row`/`Grid.Column` a nakoniec vycentrovať výpis.

Ďalším elementom potrebným pre zadanie vstupu od používateľa je element `TextBox`. Dôležité atribúty tohto elementu sú atribút `Name`, cez ktorý bude možné získať v kóde hodnotu zadanú používateľom a atribúty pozície.

Posledným elementom, ktorý je potrebné pridať je element `Button`. Dôležité atribúty tohto elementu sú `Content`, ktorý definuje, čo bude na tlačidle vypísané, atribúty jeho pozície a najdôležitejší atribút `Click`, v ktorom sa definuje názov obslužnej funkcie:

```
<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="2*"/>
      <RowDefinition Height="2*"/>
      <RowDefinition Height="5*"/>
   </Grid.RowDefinitions>

   <Label Name="U1_Label_text" Content="Insert text below:" Grid.Row="0"
HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U1_user_input" Grid.Row="1" Height="20" Width ="150" VerticalAlignment="Top">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Button Content="Display" Grid.Row="2" Grid.Column="0" Height="20" Width="70"
BorderThickness="0" VerticalAlignment="Top" Click="U1_Button">
      <Button.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </Button.Effect>
   </Button>
</Grid>
```

Po vytvorení sa odporúča kliknúť na názov, ktorý bol zadaný do atribútu `Click` pravým tlačidlom a zvoliť možnosť **Go To Definition**, čo automaticky vytvorí novú funkciu s príslušným názvom v súbore „*MainWindow.xaml.cs*“. Vzorový kód vytvorenej prázdnej funkcie je nasledovný:

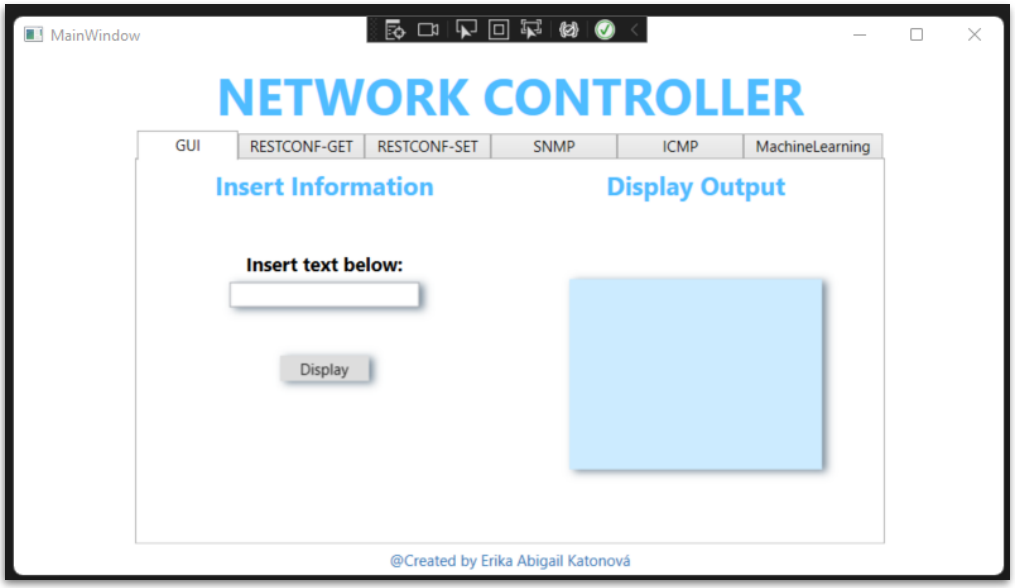
```
private void U1_Button(object sender, RoutedEventArgs e){}
```

Do tejto funkcie je možné vpisovať kód predstavujúci akcie, ktoré sa vykonajú po stlačení príslušného tlačidla. K vytvoreniu takého kódu sa ešte vrátime.

Poslednou časťou používateľského rozhrania je definovanie pravého stĺpca, v ktorom vytvoríme dvojicu riadkov, kde do prvého vložíme cez element `Label` text a v druhom riadku sa bude nachádzať element `ListBox`, ktorý po stlačení vytvoreného tlačidla zobrazí text zadaný od používateľa:

```
<Grid Grid.Column="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="*"/>
   </Grid.RowDefinitions>

   <Label Name="U1_Label_text_2" Content="Display Output" Grid.Row="0" Grid.Column="0"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20"
FontWeight="Bold"/>

   <ListBox Name="U1_user_output" Grid.Row="1" Height="150" Width="200" VerticalAlignment="Center"
Background="#ccebff" BorderThickness="0">
      <ListBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </ListBox.Effect>
   </ListBox>
</Grid>
```

Najdôležitejším atribútom elementu `ListBox` je atribút `Name`, ktorý umožní do tohto elementu vpísať zvolený text. Vzorové používateľské rozhranie po splnení úlohy 1 je zobrazené na obrázku 5.7:



Obrázok 5.7 Vzorové používateľské rozhranie po splnení prvej úlohy

Poslednou časťou tejto úlohy je vytvorenie tela funkcie, ktorá po stlačení tlačidla vypíše text zadaný od používateľa do modrého obdĺžnika. Z pohľadu kódu stačí vyčítať text obsahu elementu pre zadanie vstupu od používateľa a následne tento obsah pridať do elementu pre výpis. K elementom sa dostaneme cez ich atribút `Name`. Vzorový kód riešenia je zobrazený v nasledujúcom výpise:

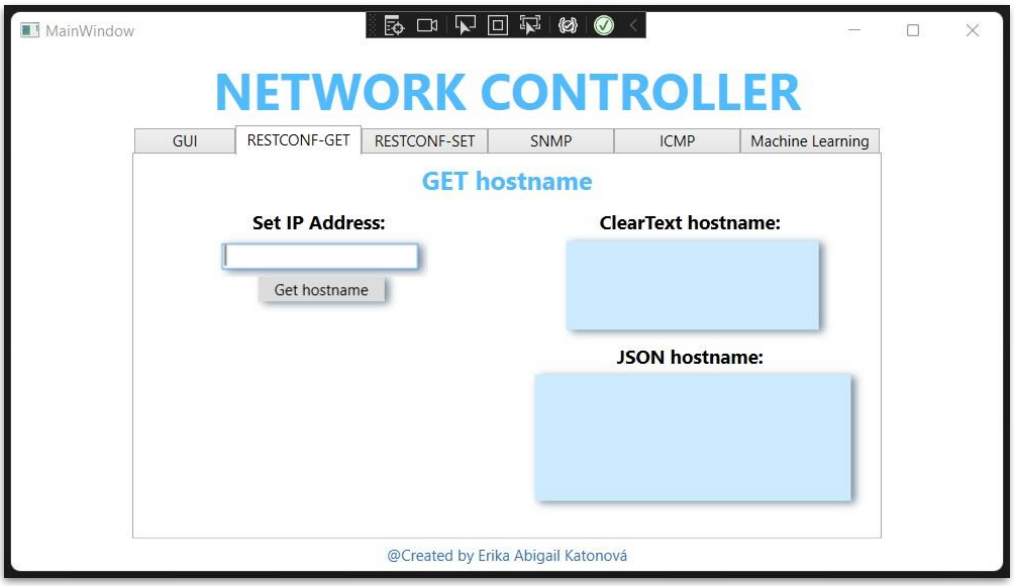
```
private void U1_Button(object sender, RoutedEventArgs e) {
   //získanie textu zadaného od používateľa
   string U_Input = U1_user_input.Text;

   //kontrolný výpis do Output konzoly
   System.Diagnostics.Debug.WriteLine(U_Input);

   //vypísanie získaného textu do List_Boxu
   U1_user_output.Items.Add(U_Input);

   //vymazanie textu zadaného používateľom
   U1_user_input.Clear();
}
```

Funkcia pre lepšie hľadanie chýb vypisuje načítaný reťazec od používateľa do výstupu konzoly. Pre krajšiu funkcionalitu bola pridaná aj vlastnosť, kedy sa po stlačení tlačidla automaticky vyčistí pole pre vstup od používateľa.

Úloha 2: RESTCONF – čítanie konfigurácie

Zadanie: Umožnite kontroléru vyčítať a v aplikácii zobraziť názov zariadenia využitím protokolu RESTCONF.

Táto úloha znázorní, ako je možné z desktopovej aplikácie vykonávať HTTPS GET požiadavky na externé aplikácie. Pred písaním samotného kódu funkcie je potrebné vytvoriť používateľské rozhranie, v ktorom používateľ špecifikuje IP adresu zariadenia, z ktorého chce názov čítať. Rozhranie tiež musí obsahovať tlačidlo, po ktorého stlačení sa vykoná príslušná akcia. Nakoniec musí tiež obsahovať element, ktorý vypíše názov zariadenia. Pre lepšie pochopenie vytvoríme dva takéto objekty, pričom do prvého sa vypíše len názov zariadenia a do druhého celý JSON obsahujúci názov zariadenia. Požadované používateľské rozhranie je zobrazené na obrázku 5.8:



Obrázok 5.8 Vzorové používateľské rozhranie po splnení druhej úlohy

Uvedené používateľské rozhranie umožní používateľovi zadať IP adresu cieľového zariadenia a po stlačení tlačidla **Get hostname** odoslať HTTPS GET požiadavku na sieťové zariadenie.

Vzorový kód pre vytvorenie uvedeného rozhrania je možné vidieť na nasledujúcich výpisoch:

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition/>
</Grid.RowDefinitions>

<Label Name="U2_Label_Title" Content="GET hostname" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20" FontWeight="Bold"/>

<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="3*"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="5*"/>
   </Grid.RowDefinitions>
```

```
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <Label Name="U2_Label_IP_address" Content="Set IP Address: " Grid.Row="0" Grid.Column="0"
HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="16" FontWeight="Bold"/>

    <TextBox Name="U2_IP_address" Grid.Row="1" Grid.Column="0" Height="20" Width="150"
VerticalAlignment="top" HorizontalAlignment="Center">
        <TextBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </TextBox.Effect>
    </TextBox>

    <Button Content="Get hostname" Grid.Row="1" Grid.Column="0" Height="20" Width="100"
BorderThickness="0" VerticalAlignment="Center" HorizontalAlignment="Center" Click="U2_Button">
        <Button.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </Button.Effect>
    </Button>

    <Label Name="U2_Label_Clear_Hostname" Content="ClearText hostname: " Grid.Row="0"
Grid.Column="2" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="16"
FontWeight="Bold"/>

    <ListBox Name="U2_ClearText_hostname" Grid.Row="1" Grid.Column="2" Height="70" Width="200"
VerticalAlignment="Top" Background="#ccebff" BorderThickness="0">
        <ListBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </ListBox.Effect>
    </ListBox>

    <Label Name="U2_Label_JSON_Hostname" Content="JSON hostname: " Grid.Row="2" Grid.Column="2"
HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="16" FontWeight="Bold"/>

    <ListBox Name="U2_JSON_Hostname" Grid.Row="3" Grid.Column="2" Height="100" Width="250"
VerticalAlignment="Top" Background="#ccebff" BorderThickness="0">
        <ListBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </ListBox.Effect>
    </ListBox>
</Grid>
```

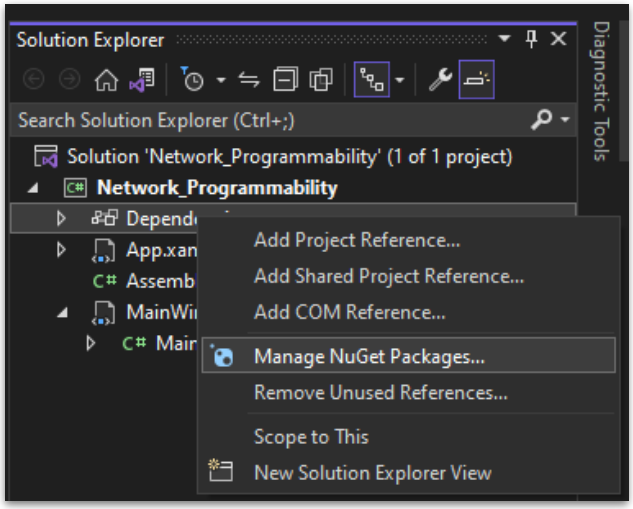
Podobne ako v prvej úlohe aj v tomto prípade je po vytvorení tlačidla potrebné kliknúť pravým tlačidlom myši na atribút `Click` pre účely vygenerovania obslužnej funkcie. Funkcia vyžaduje nastavenie štandardných atribútov pre HTTPS komunikáciu, medzi ktoré radíme URI, hlavičky pre `Content-Type`, `Accept` a `Authorization`.

URI adresa, cez ktorú je možné získať názov sieťového zariadenia vyzerá nasledovne:

https://ip_add /restconf/data/native/hostname

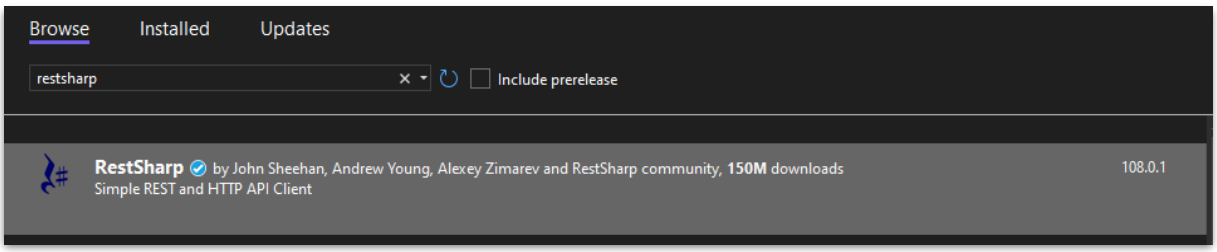
Keďže sa bude pracovať s dátami v JSON formáte a s protokolom HTTPS, je potrebné do projektu doinštalovať balíčky pre prácu s JSON formátom a REST službami. Tieto balíčky je potom po inštalácii potrebné zavolať v rámci zdrojového kódu.

Inštaláciu balíčka je možné realizovať nasledujúcim spôsobom. Najskôr je potrebné v časti **Solution Explorer** (v pravej časti Visual Studio okna) kliknúť pravým tlačidlom na **Dependencies** projektu a zvoliť možnosť **Manage NuGet Packages**, čo zobrazuje obrázok 5.9:



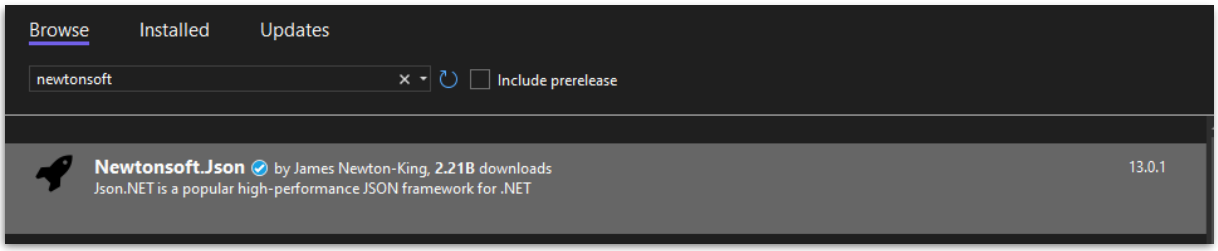
Obrázok 5.9 Cesta ku správe balíčkov projektu

V novo-otvorenom okne treba následne zvoliť **Browse** a do vyhľadávacieho poľa zadať „***restsharp***“. VisualStudio ponúkne množinu balíčkov, z ktorých si vyberieme ten potrebný (na obrázku 5.10) a stlačíme **Install**:



Obrázok 5.10 Vyhľadanie požadovanej verzie balíčka RestSharp

Pre prácu s JSON objektami je potrebné nainštalovať balíček **Newtonsoft.Json** (obrázok 5.11):



Obrázok 5.11 Vyhľadanie požadovanej verzie balíčka Newtonsoft.Json

Po inštalácii je možné využitím nasledujúceho kódu pripojiť balíčky a implementovať obslužnú funkciu:

```
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;
using RestSharp;
using RestSharp.Authenticators;
using Newtonsoft.Json.Converters;
```

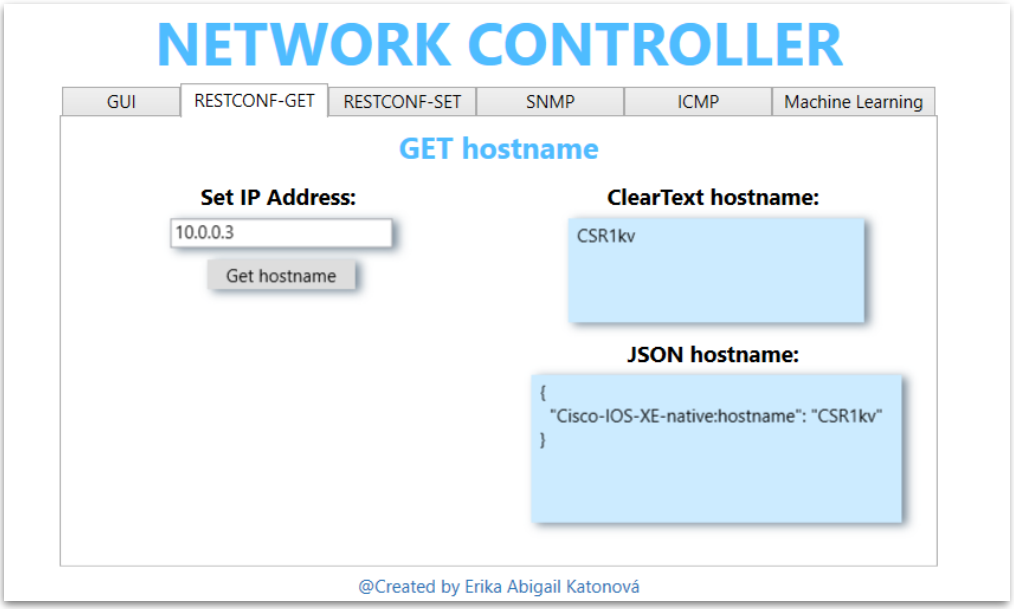
```
private void U2_Button(object sender, RoutedEventArgs e) {
   //získanie IP adresy spravovaného zariadenia
   string ip_add = U2_IP_address.Text;

   //inicializácia klienta a priradenie URI identifikátora
   var client = new RestClient("https://" + ip_add + "/restconf/data/native/hostname");
   client.RemoteCertificateValidationCallback = (sender, certificate, chain, sslPolicyErrors) =>
true;
   client.Timeout = -1;

   //definovanie typu požiadavky a vytvorenie príslušných hlavičiek
   var request = new RestRequest(Method.GET);
```

```
    request.AddHeader("Content-Type", "application/yang-data+json");
    request.AddHeader("Accept", "application/yang-data+json");
    request.AddHeader("Authorization", "Basic Y2lzY286Y2lzY28xMjMh");

    //odoslanie požiadavky
    IRestResponse response = client.Execute(request);
    //vypísanie do debug outputu kód úspešnosti vykonania požiadavky – pre prípad hľadania chýb
    System.Diagnostics.Debug.WriteLine(response.StatusCode);
    System.Diagnostics.Debug.WriteLine(response.Content);

    //vypísanie prijatého obsahu do aplikácie v JSON formáte
    U2_JSON_Hostname.Items.Add(response.Content);
    //vytvorenie JSON objektu a jeho rozparsovanie
    JObject json_object = JObject.Parse(response.Content);

    //uloženie hodnoty názvu zariadenia v stringu do premennej a jej vloženie do aplikácie
    string device_hostname = json_object["Cisco-IOS-XE-native:hostname"].ToString();
    U2_ClearText_hostname.Items.Add(device_hostname);
}
```

Ak by sme chceli urobiť aplikáciu odolnú voči chybným vstupom, bolo by potrebné overovať správnosť reťazca zadaného od používateľa (teda či obsahuje 4 hodnoty v rozsahu od 0 do 255 oddelené bodkami). Následne, ak by bola zadaná IP adresa v správnom formáte, tak by sa overovala návratová hodnota HTTPS odpovede a v prípade, ak by Status kód nebol v rozsahu 200, aplikácia by nemala pokračovať vo vykonávaní funkcie.

Upozornenie: Uvedený kód má za úlohu demonštrovať základnú prácu s HTTPS protokolom a prácou s dátami v JSON formáte. Ukážka predpokladá korektný formát vstupných dát. Teda používateľ musí zadať IP adresu v správnom formáte.

Aplikácia by mala po stlačení tlačidla **Get hostname** vyprodukovať nasledujúci výstup (obrázok 5.12):



Obrázok 5.12 Výstup aplikácie po naprogramovaní druhej úlohy

Úloha 3: RESTCONF – zmena konfigurácie

<u>Zadanie</u>: Využitím protokolu RESTCONF vykonajte zmenu názvu zariadenia.

Podobná úloha ako v predchádzajúcom prípade, no teraz je po vyplnení IP adresy potrebné zadať aj nový názov zariadenia, ktorý sa použije na zmenu pôvodného názvu. Rovnako ako v predchádzajúcom protokole, aj teraz bude na zmenu využitý protokol RESTCONF. Grafické rozhranie zobrazené na obrázku 5.13 s príslušným kódom v nasledujúcich výpisoch by mohlo vyzerať nasledovne:

Obrázok 5.13 Vzorové používateľské rozhranie po splnení tretej úlohy

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition/>
</Grid.RowDefinitions>
<Label Name="U3_Label_Title" Content="SET hostname" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20" FontWeight="Bold"/>

<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="4*"/>
   </Grid.RowDefinitions>
   <Label Name="U3_Label_IP_address" Content="Set IP Address: " Grid.Row="0"
HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U3_IP_address" Grid.Row="1" Height="20" Width="150" VerticalAlignment="Top"
HorizontalAlignment="Center">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Label Name="U3_Label_New_Hostname" Content="Insert hostname: " Grid.Row="2"
HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U3_New_hostname" Grid.Row="3" Height="20" Width="150" VerticalAlignment="Top"
HorizontalAlignment="Center">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>
```

```
    <Button Content="Set hostname" Grid.Row="4" Height="20" Width="100" VerticalAlignment="Top"
Click="U3_Button">
        <Button.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </Button.Effect>
    </Button>
</Grid>
```

Zdrojový kód k funkcii na zmenu názvu zariadenia bude podobný ako v predchádzajúcej úlohe. Jedinou zmenou bude HTTPS metóda zmenená z GET na PUT a tiež je potrebné pridať telo správy, ktoré bude obsahovať daný nový názov zariadenia v správnom JSON formáte.

Vzorové riešenie v podobe komentovaného zdrojového kódu je zobrazené v nasledujúcom výpise:

```
private void U3_Button(object sender, RoutedEventArgs e) {
    //získanie IP adresy zariadenia a nový názov pre spravované zariadenie
    string ip_add = U3_IP_address.Text;
    string hostname = U3_New_hostname.Text;

    //overenie, či bola zadaná nejaká hodnota v poli pre nový názov zariadenia
    if (!string.IsNullOrWhiteSpace(hostname))
    {
        //inicializácia klienta a priradenie URI
        var client = new RestClient("https://" + ip_add + "/restconf/data/native/hostname");
        client.RemoteCertificateValidationCallback = (sender, certificate, chain, sslPolicyErrors) =>
true;
        client.Timeout = -1;

        //vyčistenie formulára pre zadanie názvu zariadenia
        U3_New_hostname.Clear();

        //vytvorenie objektu pre uloženie JSON objektu
        dynamic obj = new JObject();

        //priradenie načítaného názvu zariadenia
        obj["Cisco-IOS-XE-native:hostname"] = hostname;

        //serializácia JSON objektu do stringu - na odoslanie
        var jsonString = Newtonsoft.Json.JsonConvert.SerializeObject(obj);

        //vytvorenie PUT požiadavky s príslušnými hlavičkami a telom správy
        var request = new RestRequest(Method.PUT);
        request.AddHeader("Content-Type", "application/yang-data+json");
        request.AddHeader("Accept", "application/yang-data+json");
        request.AddHeader("Authorization", "Basic Y2lzY286Y2lzY28xMjMh");
        request.AddParameter("application/yang-data+json", jsonString, ParameterType.RequestBody);

        //odoslanie požiadavky
        IRestResponse response = client.Execute(request);

        //vypísanie prijatého obsahu ako odpovede do debug výstupu
        System.Diagnostics.Debug.WriteLine(response.Content);
    }
}
```

Upozornenie: Realizáciou predchádzajúcich úloh je demonštrovaný modelovo riadený prístup správy konfigurácie sieťových zariadení. Ďalšie úlohy implementovať netreba – zmena akejkoľvek konfigurácie je iná len tým, že sa pracuje s inými JSON objektmi.

Úloha 4: SNMP – čítanie údajov

<u>Zadanie</u>: Do kontroléra implementujte funkcionalitu, ktorá po kliknutí na príslušné tlačidlo vyčíta využitím protokolu SNMP zo zariadenia: základný popis, ako dlho je spustené, názov a vyťaženie procesora za posledných 5s a poslednú 1min.

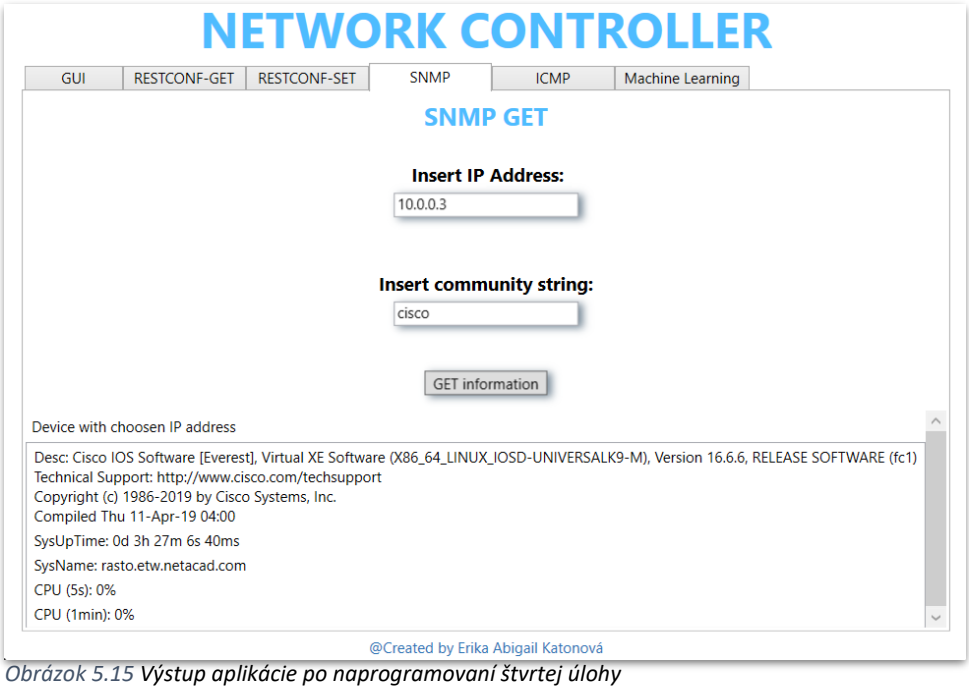
Práca s protokolom SNMP vyžaduje doinštalovať balíček **SnmpSharpNet** a pridať ho medzi používané balíčky príkazom:

```
using SnmpSharpNet;
```

Používateľské rozhranie by malo obsahovať možnosť zadať IP adresu zariadenia spolu s komunitným reťazcom. Vzorovú ukážku rozhrania znázorňuje obrázok 5.14:

Obrázok 5.14 Vzorové používateľské rozhranie po splnení štvrtej úlohy

Aplikácia by po stlačení tlačidla **GET information** mala vypísať požadované údaje, pričom po opätovnom klikaní na tlačidlo by sa tieto informácie mali aktualizovať. Očakávaný výstup po naprogramovaní obslužnej funkcie je uvedený na obrázku 5.15:



Obrázok 5.15 Výstup aplikácie po naprogramovaní štvrtej úlohy

Skript pre vytvorenie uvedeného používateľského rozhrania vyzerá nasledovne:

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition/>
</Grid.RowDefinitions>
<Label Name="U4_Label_Title" Content="SNMP GET" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20" FontWeight="Bold"/>

<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="4*"/>
   </Grid.RowDefinitions>
   <Label Content="Insert IP Address:" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U4_IP_address" Grid.Row="1" Height="20" Width ="150" HorizontalAlignment="Center"
VerticalAlignment="Top">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Label Content="Insert community string:" Grid.Row="2" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U4_Comm_String" Grid.Row="3" Height="20" Width ="150"
HorizontalAlignment="Center" VerticalAlignment="Top">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Button Content="GET information" Grid.Row="4" Height="20" Width="100"
VerticalAlignment="Center" Click="U4_Button">
      <Button.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </Button.Effect>
   </Button>

   <ScrollViewer Grid.Row="5" Grid.ColumnSpan="2" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">
      <StackPanel Name="U4_SNMP_output"/>
   </ScrollViewer>
```

Kód obsahuje štandardné objekty, s ktorými sa pracovalo aj v predchádzajúcich úlohách. Pribudli len dva nové elementy a to `ScrollViewer` a `StackPanel`. `ScrollViewer` umožní do seba vkladať objekty pričom v prípade presahu cez okno sa vytvorí scrollovacia lišta, ktorá umožní pracovať s rozsiahlejšími výpismi. Druhým novým elementom je element `StackPanel`, do ktorého je možné pridávať nové a nové výpisy, kde tieto výpisy budú ukladané postupne pod seba. Tento element je neskôr využitý v kóde, kde do neho budeme vkladať príslušné údaje čítané cez protokol SNMP.

Doteraz sa výpisy do aplikácie realizovali vypĺňaním obsahu do pred vytvorených elementov. Rámec WPF ale umožňuje priamo v zdrojovom kóde funkcie vytvárať nové elementy a vkladať ich do hlavného okna. V tomto prípade budú vytvárané elementy `Label` a `ListBox`, ktoré budú vložené ako vnorené elementy (*Children*) pre element `StackPanel`, na ktorý budú naviazané cez jeho atribút `Name`.

Vzorový komentovaný zdrojový kód pre prácu s protokolom SNMP vyzerá nasledovne:

```
private void U4_Button(object sender, RoutedEventArgs e) {
   //načítanie IP adresy a komunitného stringu
   string IP_address = U4_IP_address.Text;
   string Community_string_app = U4_Comm_String.Text;
   //zmazanie zobrazeného zoznamu v aplikácii
   U4_SNMP_output.Children.Clear();

   //vyplnenie potrebných údajov a vytvorenie SNMP paketu
      //SNMP community string
   OctetString community = new OctetString(Community_string_app);
      //definovanie parametrov pre agenta realizujúceho pripojenie
   AgentParameters param = new AgentParameters(community);
      //definovanie verzie SNMP
   param.Version = SnmpVersion.Ver1;
      //nastavenie IP adresy
   IpAddress agent = new(IP_address);
      //nastavenie cieľovej požiadavky - definovanie adresy, UDP portu ...
   UdpTarget target = new UdpTarget((System.Net.IPAddress)(IpAddress)agent, 161, 2000, 1);

      //vytvorenie pdu, ktoré bude obsahovať typ akcie a OID objektov
   Pdu pdu = new Pdu(PduType.Get);
   pdu.VbList.Add("1.3.6.1.2.1.1.1.0");          //sysDescr
   pdu.VbList.Add("1.3.6.1.2.1.1.3.0");          //sysUpTime
   pdu.VbList.Add("1.3.6.1.2.1.1.5.0");          //sysName
   pdu.VbList.Add("1.3.6.1.4.1.9.2.1.56.0");     //CPU usage last 5s
   pdu.VbList.Add("1.3.6.1.4.1.9.2.1.57.0");     //CPU usage last 1min

      //odoslanie požiadavky
   SnmpV1Packet result = (SnmpV1Packet)target.Request(pdu, param);
   //podmienka overenia prijatia odpovede
   if (result != null) {
      //ak odpoveď obsahuje chybu, tak sa info o nej vypíše do konzoly
      if (result.Pdu.ErrorStatus != 0) {
         System.Diagnostics.Debug.WriteLine("Error in SNMP reply. Error {0} index {1}",
result.Pdu.ErrorStatus, result.Pdu.ErrorIndex);
      }
      else {
         //vytvorenie značky a poľa pre vypísanie získaných údajov
         Label r = new Label();
         r.Content = "Device with choosen IP address";
         ListBox info = new ListBox();
         info.Items.Add("Desc: "+result.Pdu.VbList[0].Value.ToString());
         info.Items.Add("SysUpTime: "+result.Pdu.VbList[1].Value.ToString());
         info.Items.Add("SysName: " +result.Pdu.VbList[2].Value.ToString());
         info.Items.Add("CPU (5s): "+result.Pdu.VbList[3].Value.ToString() + "%");
         info.Items.Add("CPU (1min): "+result.Pdu.VbList[4].Value.ToString() + "%");
         //do okna aplikácie pridá nový Label a ListBox a vypíše príslušné údaje
         U4_SNMP_output.Children.Add(r);
         U4_SNMP_output.Children.Add(info);
      }
   }
   else {
      System.Diagnostics.Debug.WriteLine("No response received from SNMP agent.");
   }
   target.Close();
}
```

Úloha 5: ICMP – test dostupnosti

<u>Zadanie</u>: Doplňte funkcionalitu pre testovanie dostupnosti zadanej IP adresy využitím protokolu ICMP, pričom informáciu o úspešnosti/neúspešnosti vypíšte používateľovi na obrazovku.

Úloha vyžaduje z pohľadu používateľského rozhrania zadanie IP adresy, pre ktorú chceme overiť dostupnosť, tlačidlo, ktoré spustí test a pole pre vypísanie výsledku.

Príklad výsledného rozhrania (na obrázku 5.16) a k nemu príslušný zdrojový kód:

Obrázok 5.16 Vzorové používateľské rozhranie po splnení piatej úlohy

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition/>
</Grid.RowDefinitions>
<Label Name="U5_Label_Title" Content="ICMP test" Grid.Row="0" Grid.ColumnSpan="4"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20"
FontWeight="Bold"/>

<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="2*"/>
      <RowDefinition Height="1*"/>
      <RowDefinition Height="3*"/>
   </Grid.RowDefinitions>
   <Label Content="Insert IP Address:" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontSize="15" FontWeight="Bold"/>

   <TextBox Name="U5_IP_address" Grid.Row="1" Height="20" Width ="150" HorizontalAlignment="Center"
VerticalAlignment="Top">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Button Content="Ping device" Grid.Row="2" Height="20" Width="100" VerticalAlignment="Center"
Click="U5_Button">
      <Button.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </Button.Effect>
   </Button>

   <Label Content="Result:" Grid.Row="3" HorizontalAlignment="Center" VerticalAlignment="Bottom"
FontSize="15" FontWeight="Bold"/>
```

```
    <ListBox Name="U5_result" Grid.Row="4" Height="20" Width ="100" HorizontalAlignment="Center"
VerticalAlignment="Top">
        <ListBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </ListBox.Effect>
    </ListBox>
</Grid>
```

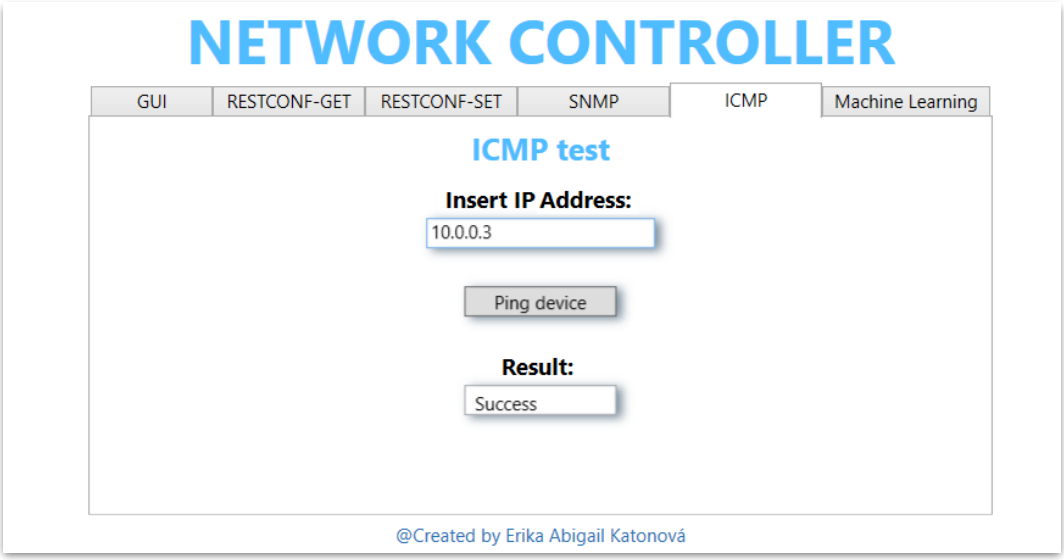
Pred tým, ako začneme implementovať funkcionalitu pre túto úlohu, je potrebné importovať nový modul pre prácu s ICMP protokolom a to konkrétne zadaním:

```
using System.Net.NetworkInformation;
```

Komentovaný kód funkcie pre generovanie ICMP požiadaviek na zadanú IP adresu je uvedený v nasledujúcom výpise:

```
private void U5_Button(object sender, RoutedEventArgs e) {
    //zmazanie starého výstupu
    U5_result.Items.Clear();

    //načítanie zadanej IP adresy
    string IP_address = U5_IP_address.Text;

    //vytvorenie objektu pre odoslanie ICMP echo-request požiadavky
    Ping ping = new Ping();

    //odoslanie ICMP požiadavky na príslušnú adresu
    PingReply reply = ping.Send(IP_address);

    //zobrazenie prijatej odpovede na príslušné miesto v aplikácii
    U5_result.Items.Add(reply.Status);
}
```

Ako je možné vidieť, tak v tejto úlohe stačí použiť vytvorené základné funkcie a objekty. Obrázok 5.17 znázorňuje výstup aplikácie po zadaní IP adresy a stlačení tlačidla:



Obrázok 5.17 Výstup aplikácie po naprogramovaní piatej úlohy

Úloha 6: Integrácia strojového učenia

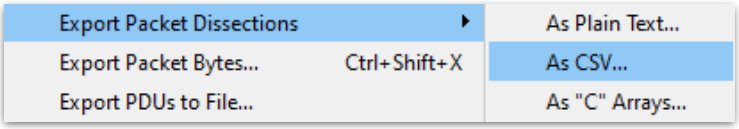
<u>Zadanie</u>: Integrujte do topológie model strojového učenia umožňujúci klasifikovať pakety.

Strojové učenie je možné v počítačových sieťach využiť na množstvo účelov. Najčastejšie sa jedná o klasifikáciu dát, detekciu rôznych neštandardných situácií, prípadne na detekciu útokov. V rámci tejto ukážky je uvedený postup, ako pridať do kontroléra funkcionalitu, ktorá na základe údajov o odchytenom pakete zadaných od používateľa dokáže klasifikovať paket do kategórie, do ktorej patrí. Cieľom je, aby daný klasifikátor dokázal rozoznať štandardné typy paketov pre protokoly ICMP, RIP, EIGRP a OSPF. Z pohľadu implementácie strojového učenia je použitá .NET knižnica **ML.net**.

Prvým krokom potrebným pre prácu s modelmi strojového učenia je existencia dátovej sady, ktorú je možné využiť na trénovanie modelov. Tento manuál využíva prístup vytvorenia vlastnej dátovej sady, vhodnej pre predikciu uvedených protokolov. Dátové sady sa štandardne vytvárajú v CSV formáte, ktorý pozostáva z množiny údajov oddelených čiarkou. Dátovú sadu je možné vytvoriť využitím nástroja *Wireshark*, ktorý dokáže exportovať záznam o všetkých odchytených paketoch. Pre jednoduchosť stačí v emulátore GNS3 vytvoriť topológiu prepájajúcu dvojicu smerovačov (obrázok 5.18), a na tomto prepojení spustiť odchytávanie paketov nástrojom *Wireshark*. Ďalej je potrebné nakonfigurovať smerovače tak, aby medzi sebou komunikovali využitím všetkých potrebných protokolov.

Obrázok 5.18 Spustenie odchytávania paketov nástrojom Wireshark na prepoji 2 smerovačov

Po odchytení dostatočného množstva paketov je potrebné v nástroji *Wireshark* zastaviť odchytávanie a exportovať záznamy (obrázok 5.19) o odchytených paketoch do CSV súboru (cez záložku **File**).



Obrázok 5.19 Export záznamov o odchytených paketoch do CSV súboru

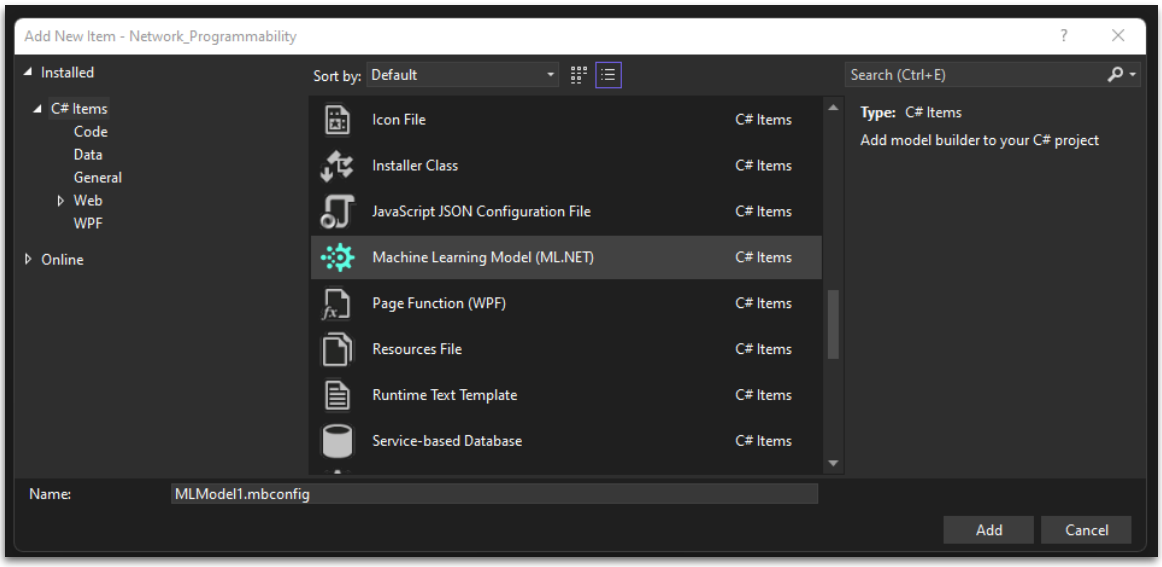
Takto vytvorená dátová sada obsahuje stĺpce [No, Time, Source, Destination, Protocol, Length, Info]. Pre efektívne spracovanie modelmi strojového učenia je ešte potrebné daný dokument upraviť a ponechať v ňom len stĺpce, ktoré sú potrebné pre klasifikáciu. V našom prípade to sú stĺpce **`Source`**, **`Destination`**, **`Protocol`** a **`Length`**. Pre jednoduchú úpravu je možné využiť napr. online nástroj dostupný na: https://extendsclass.com/csv-editor.html, kde po načítaní daného .csv súboru je možné odstrániť nepotrebné stĺpce. Očakávaný výstup je znázornený na obrázku 5.20:

```
1  Source,Destination,Protocol,Length
2  10.22.22.2,10.22.22.22,ICMP,114
3  10.22.22.22,224.0.0.9,RIPv2,66
4  10.22.22.2,224.0.0.5,OSPF,94
5  10.22.22.22,224.0.0.10,EIGRP,74
```

Obrázok 5.20 Výsledná dátová sada

Celkovo dátová sada obsahuje niekoľko násobné odchytenie daných paketov. Celková veľkosť vytvorenej dátovej sady je v našom prípade 25 000 paketov. Po vytvorení a úprave dátovej sady je možné ďalej pokračovať voľbou a prácou s modelmi strojového učenia. Využitím vývojového prostredia Visual Studio, je možné postupovať nasledovne:

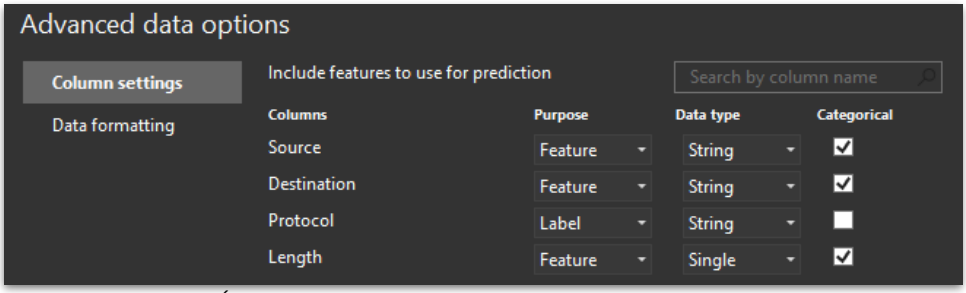
1. Pravým tlačidlom myši kliknúť na vytvorený projekt a zvoliť možnosť **Add → Machine Learning Model**, po čom sa zobrazí používateľské rozhranie ML.NET Model Builder, znázornené na obrázku 5.21:



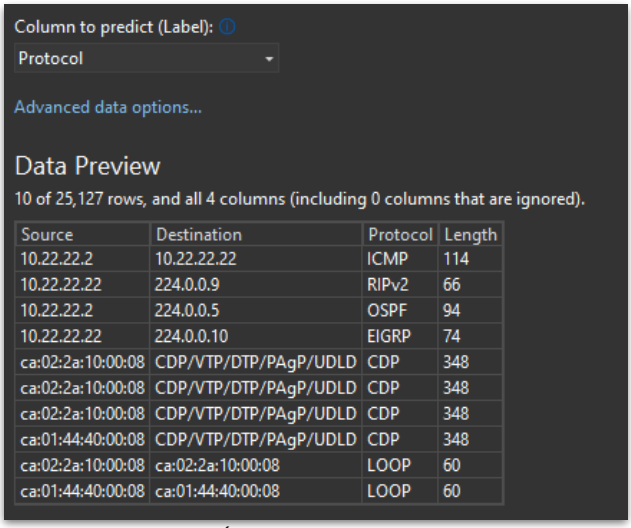
Obrázok 5.21 Návod na pridanie Modelu pre strojové učenie do projektu

2. Ďalej je ponúknutá voľba **Scenarios**, kde je potrebné zvoliť **Data Classification** a **Local**.

3. V ďalšom kroku je potrebné zvoliť vstupný súbor pre načítanie dát, z ktorého vyberieme ktoré stĺpce budú predstavovať charakteristiku vlastností (množina x), a ktorý stĺpec budeme predikovať (množina y). V našom prípade chceme predikovať stĺpec **Protocol** na základe dát v stĺpcoch **Source**, **Destination** a **Length**. To nastavíme v možnosti **Advanced data options** nasledovne podľa obrázkov 5.22 a 5.23:



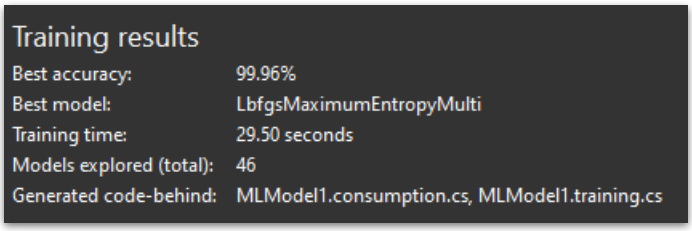
Obrázok 5.22 Voľba stĺpcov v súbore, ktoré tvoria charakteristiku vlastností



Obrázok 5.23 Voľba stĺpca pre predikciu a náhľad dát v súbore

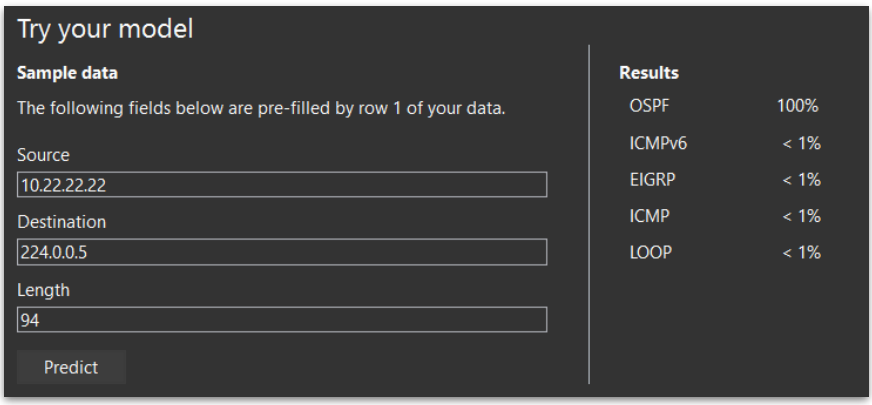
4. Ďalší krok predstavuje trénovanie modelu, kde stačí zvoliť dobu trvania trénovania. Čím viac času zadáme, tým bude predikcia presnejšia. Pre krátkosť času zvoľme napr. 30 sekúnd. V tejto chvíli je možné začať trénovanie stlačením tlačidla **Start training**.

5. Rámec ML.net za nás vykoná všetky kroky pre trénovanie a hľadanie najlepšieho modelu, pričom na konci trénovania vygeneruje report s údajmi ako je percentuálna presnosť trénovania a ktorý model bol zvolený za najlepší. V našom prípade bol najlepší model **LbfgsMaximumEntropyMulti** s presnosťou 99,96%, ako je možné vidieť na obrázku 5.24:



Obrázok 5.24 Výsledok trénovania modelu

6. Ďalším krokom je overenie funkčnosti modelu, kde Model Builder ponúkne formulár prislúchajúci k trénovacím dátam, kde môže používateľ vyplniť príslušné údaje a na základe nich predikovať výslednú hodnotu, čo znázorňuje obrázok 5.25:



Obrázok 5.25 Formulár pre testovanie modelu

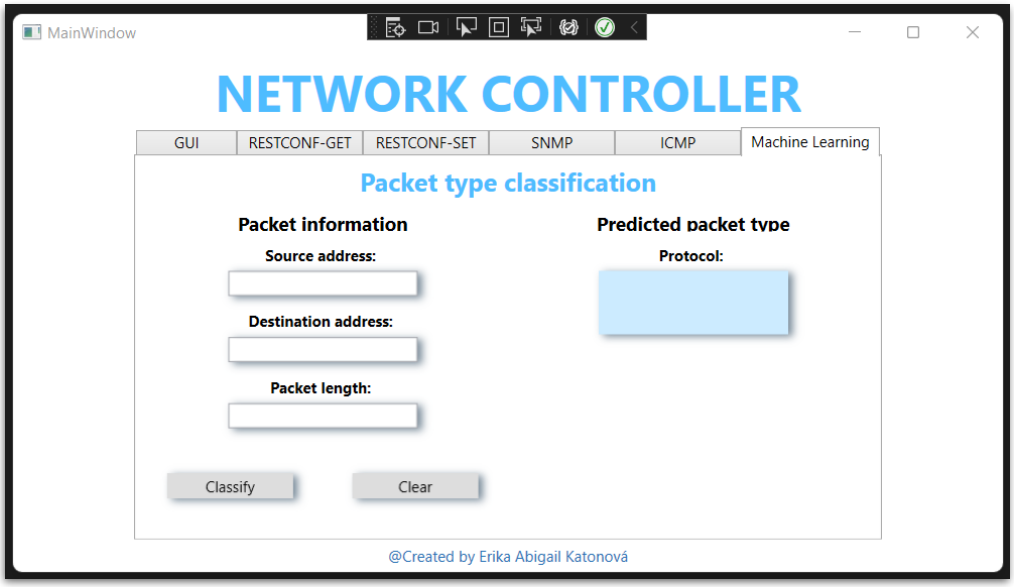
Výstup zobrazuje predikciu dát s percentuálnou pravdepodobnosťou pre jednotlivé možnosti výsledku predikcie. Po integrácii modelu do kontroléra je výstupom predikcie najlepšia hodnota, teda v našom prípade by bol výstupom protokol **OSPF**.

7. Posledným krokom je automatické generovanie zdrojového kódu a jeho nalinkovanie k nášmu projektu. Stačí kliknúť na tlačidlo **Code** a následne **Add Projects**.

8. Po vytvorení modelu a jeho pripojení ku projektu Model Builder vygeneruje návod, akým spôsobom volať vytvorený model v zdrojovom kóde.

Ukážka praktického využitia v kóde:

Majme kontrolér, ktorého časť pre strojové učenie je zobrazená na obrázku 5.26:



Obrázok 5.26 Vzorové používateľské rozhranie po splnení šiestej úlohy

Rozhranie obsahuje polia pre zadanie charakteristiky paketu a tlačidlá pre spustenie klasifikácie a vymazanie vstupu od používateľa. Rozhranie zároveň obsahuje pole pre výpis výslednej klasifikácie. Zdrojový kód príslušného rozhrania je znázornený v nasledujúcich výpisoch:

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition/>
</Grid.RowDefinitions>
<Label Content="Packet type classification" Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="#4ebcff" FontSize="20" FontWeight="Bold"/>

<Grid Grid.Row="1">
   <Grid.RowDefinitions>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="3*"/>
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition Width="5*"/>
      <ColumnDefinition Width="5*"/>
      <ColumnDefinition Width="10*"/>
   </Grid.ColumnDefinitions>
   <Label Content="Packet information" Grid.Row="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="15" FontWeight="Bold"/>

   <Label Content="Source address: " Grid.Row="1" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontWeight="Bold"/>

   <TextBox Name="U6_ML_sa" Grid.Row="2" Grid.ColumnSpan="2" Height="20" Width="150"
VerticalAlignment="top" HorizontalAlignment="Center">
      <TextBox.Effect>
         <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
      </TextBox.Effect>
   </TextBox>

   <Label Content="Destination address: " Grid.Row="3" Grid.ColumnSpan="2"
HorizontalAlignment="Center" VerticalAlignment="Bottom" FontWeight="Bold"/>
```

```
    <TextBox Name="U6_ML_da" Grid.Row="4" Grid.ColumnSpan="2" Height="20" Width="150"
VerticalAlignment="top" HorizontalAlignment="Center">
        <TextBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </TextBox.Effect>
    </TextBox>

    <Label Content="Packet length: " Grid.Row="5" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontWeight="Bold"/>

    <TextBox Name="U6_ML_l" Grid.Row="6" Grid.ColumnSpan="2" Height="20" Width="150"
VerticalAlignment="top" HorizontalAlignment="Center">
        <TextBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </TextBox.Effect>
    </TextBox>

    <Button Content="Classify" Grid.Row="7" Grid.Column="0" Height="20" Width="100"
BorderThickness="0" VerticalAlignment="Center" HorizontalAlignment="Center"
Click="U6_Button_Classify">
        <Button.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </Button.Effect>
    </Button>

    <Button Content="Clear" Grid.Row="7" Grid.Column="1" Height="20" Width="100" BorderThickness="0"
VerticalAlignment="Center" HorizontalAlignment="Center" Click="U6_Button_Clear">
        <Button.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </Button.Effect>
    </Button>

    <Label Content="Predicted packet type" Grid.Row="0" Grid.Column="2" HorizontalAlignment="Center"
VerticalAlignment="Center" FontSize="15" FontWeight="Bold"/>

    <Label Content="Protocol: " Grid.Row="1" Grid.Column="2" HorizontalAlignment="Center"
VerticalAlignment="Bottom" FontWeight="Bold"/>

    <ListBox Name="U6_ML_predicted" Grid.Row="2" Grid.RowSpan="4" Grid.Column="2" Height="50"
Width="150" VerticalAlignment="Top" Background="#ccebff" BorderThickness="0">
        <ListBox.Effect>
            <DropShadowEffect Direction="-25" Color="#FFB6C2CB" ShadowDepth="4" BlurRadius="10"/>
        </ListBox.Effect>
    </ListBox>
</Grid>
```

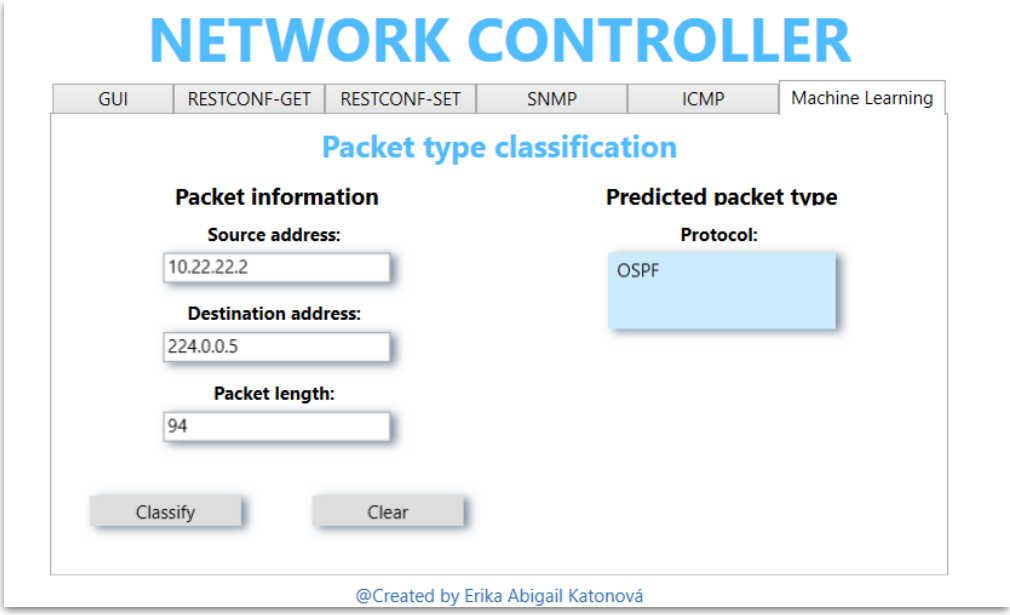
Teraz je možné vytvoriť funkciu, ktorá sa spustí po stlačení tlačidla **Classify**. Najskôr je ale potrebné použiť vytvorený modul využitím:

```
using Network_ProgramabilityML.Model;
```

Telo výslednej funkcie by mohlo vyzerať nasledovne:

```
private void U6_Button_Classify(object sender, RoutedEventArgs e) {
    //získanie zadaných údajov
    string src_add = U6_ML_sa.Text;
    string dst_add = U6_ML_da.Text;
    string p_length = U6_ML_l.Text;

    //pridanie vstupných dát pre predikciu
    var input = new ModelInput();
    input.Source = src_add;
    input.Destination = dst_add;
    input.Length = float.Parse(p_length);

    //načítanie modelu a výpis predikcie do aplikácie
    ModelOutput result = ConsumeModel.Predict(input);
    U6_ML_predicted.Items.Add(result.Prediction);
}
```

Premenné `src_add`, `dst_add` a `p_length` obsahujú hodnoty vyplnené vo formulári. Tieto hodnoty následne vložíme do premennej `input`, ktorá predstavuje vstup pre model strojového učenia. Netreba zabúdať na to, že je potrebné vyplniť, v našom prípade všetky tri atribúty, ktoré boli použité ako vstup pri tvorbe modelu. Nakoniec stačí danú premennú `input` vložiť do funkcie `Predict()`, ktorej výsledok sa uloží do premennej `result`. V našom prípade je výsledkom predikcie na základe daných dát hodnota OSPF, ktorú je možné napríklad vypísať späť do aplikácie. Tento krok v uvedenom zdrojovom kóde predstavuje posledný riadok, kde do objektu elementu `U6_ML_predicted`, ktorý predstavuje zobrazovacie pole v aplikácii, vložíme hodnotu predikcie. Výsledný stav aplikácie po stlačení tlačidla **Classify** by mohol vyzerať nasledovne (obrázok 5.27):



Obrázok 5.27 Výstup aplikácie po naprogramovaní šiestej úlohy

Poslednou nevyriešenou úlohou je vytvorenie funkcie pre tlačidlo **Clear**, ktoré len vyčistí polia vyplnené používateľom a výsledok predikcie. Telo funkcie vyzerá nasledovne:

```
private void U6_Button_Clear(object sender, RoutedEventArgs e) {
    U6_ML_sa.Clear();
    U6_ML_da.Clear();
    U6_ML_l.Clear();
    U6_ML_predicted.Items.Clear();
}
```