

Entity Framework alapú adatbáziselérés 2

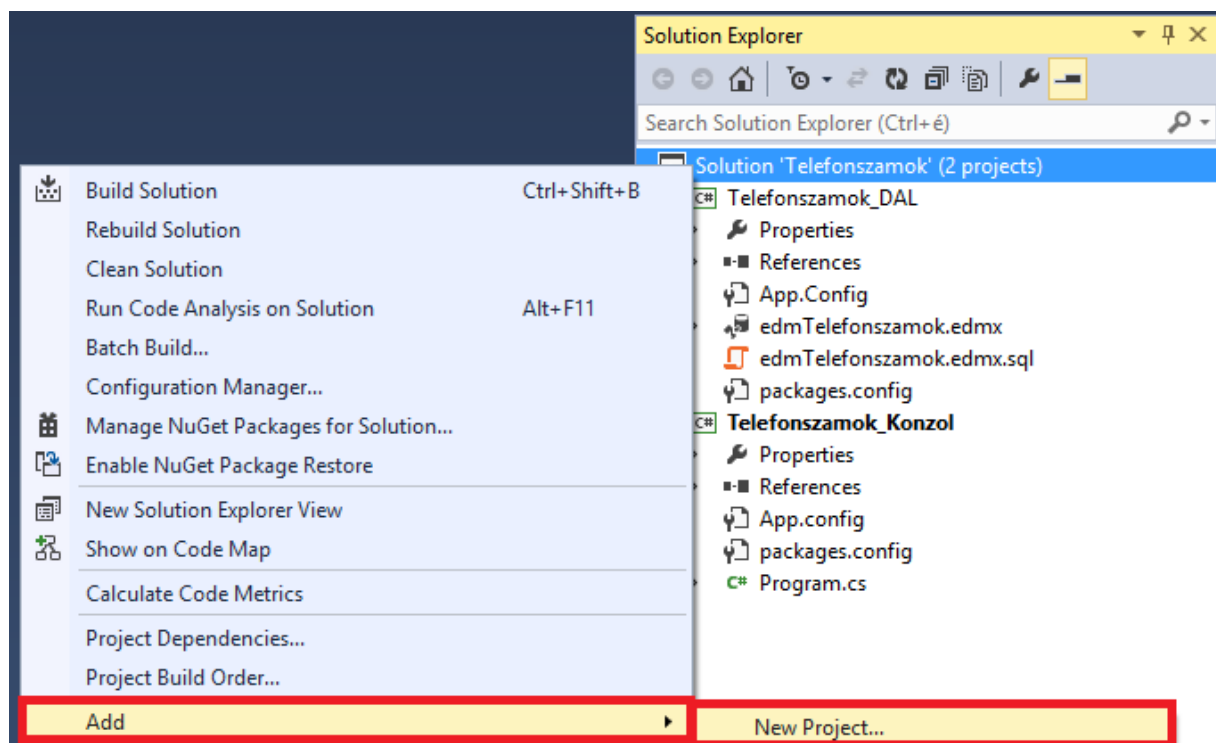
Dr. Johanyák Zsolt Csaba

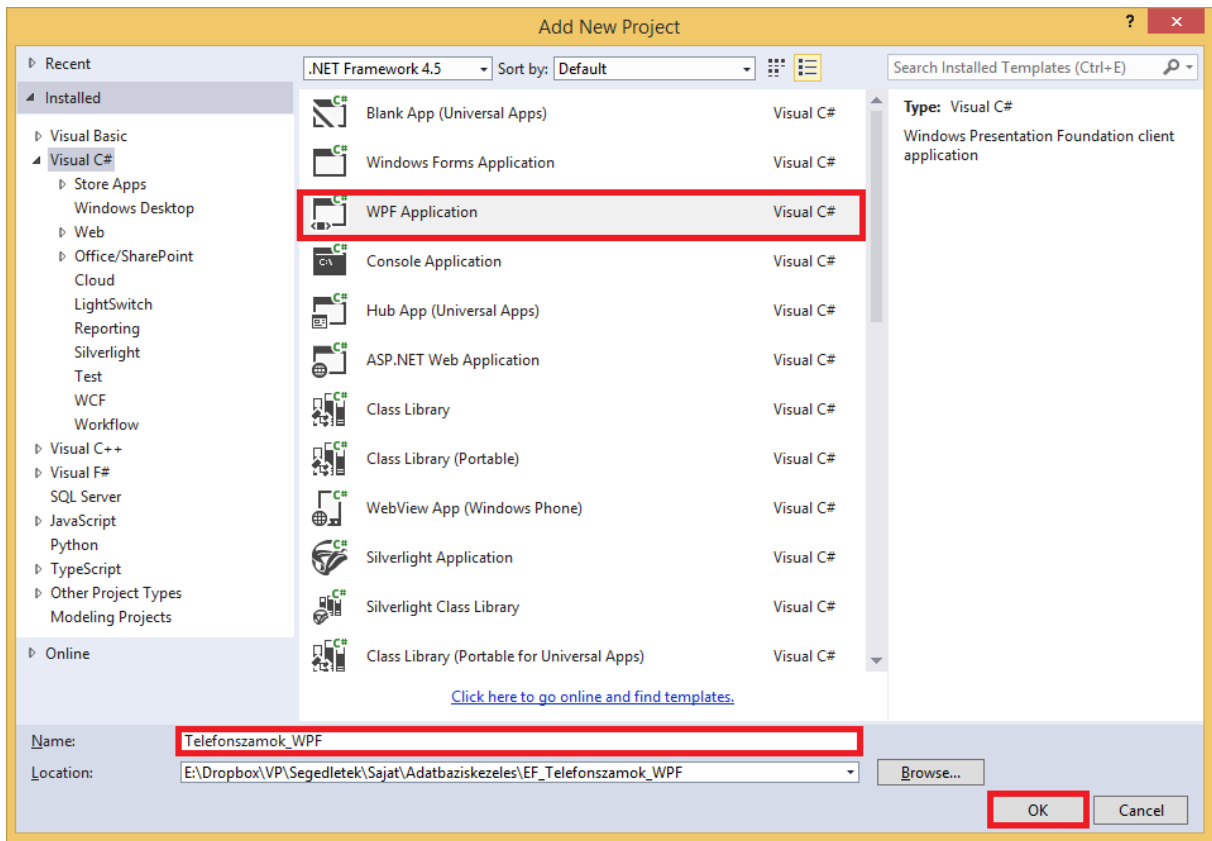
<http://johanyak.hu>

A gyakorlat célja az, hogy a korábban létrehozott Telefonszám kezelő alkalmazást kiegészítsük egy WPF típusú felülettel.

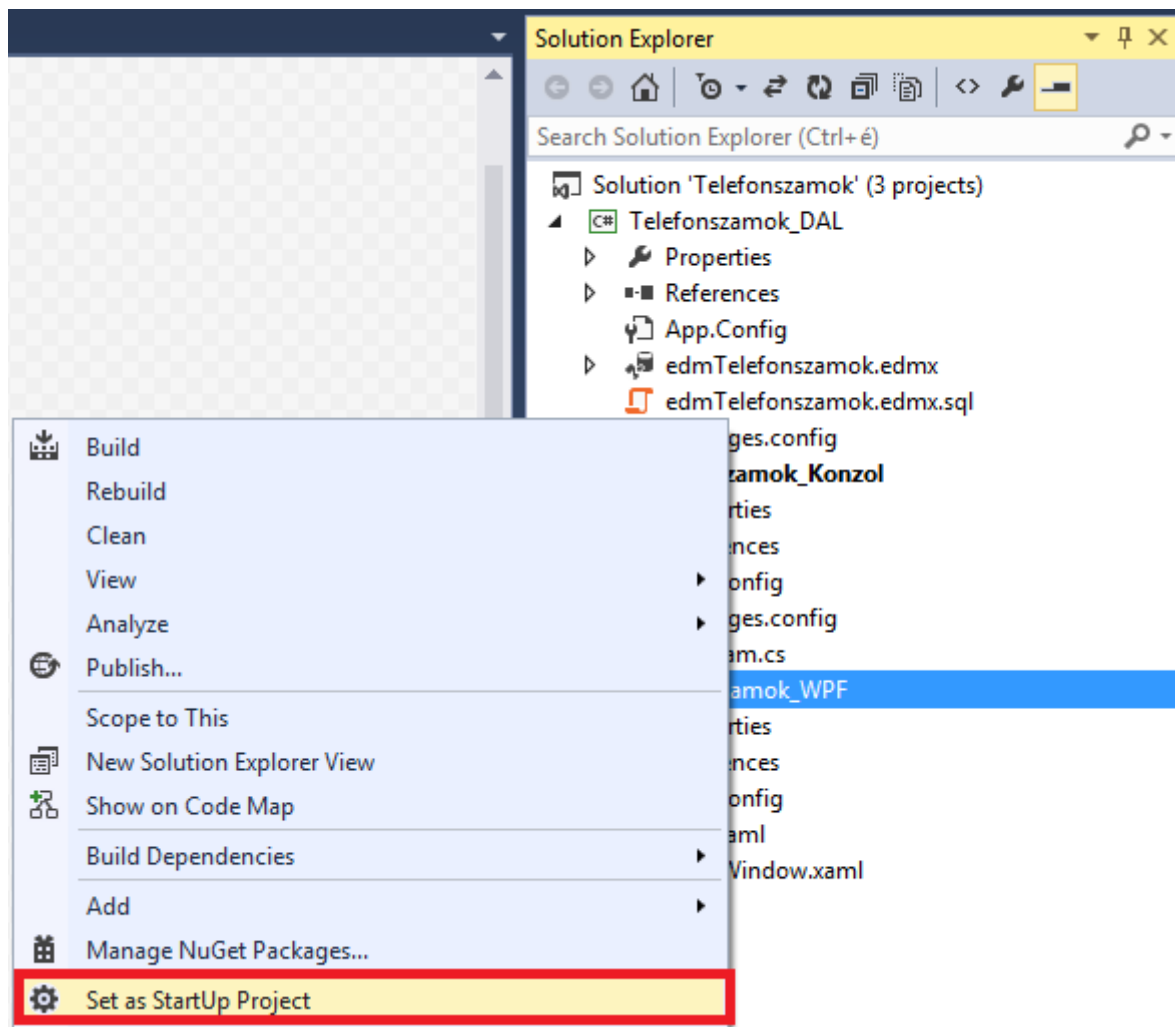
1. Projekt és alapbeállítások

Töltsük le és nyissuk meg a kiinduló megoldást (Solution-t), ami a korábban létrehozott adathozzáférési réteg (Telefonszamok_DAL) és konzol alkalmazás (Telefonszamok_Konzol) projekteket tartalmazza. Hozzunk létre egy új WPF projektet (Telefonszamok_WPF) a megoldáson belül.

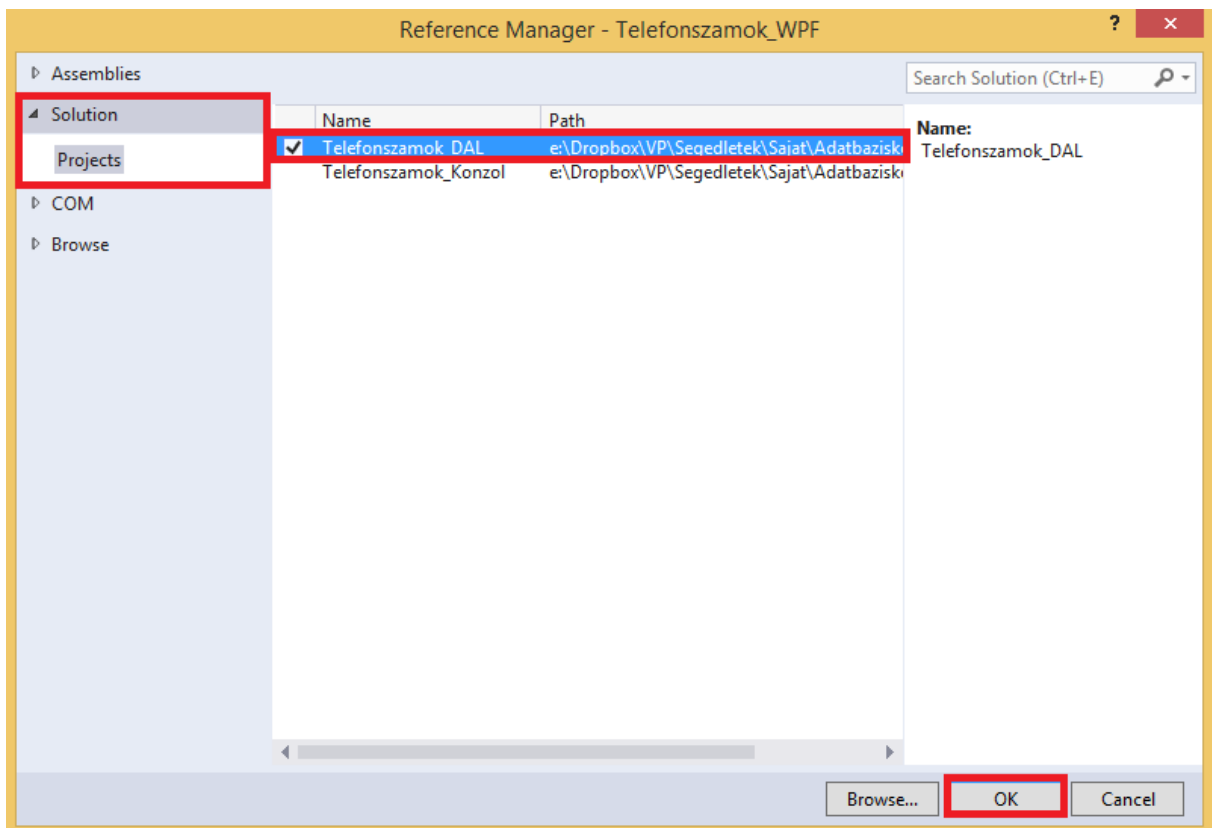
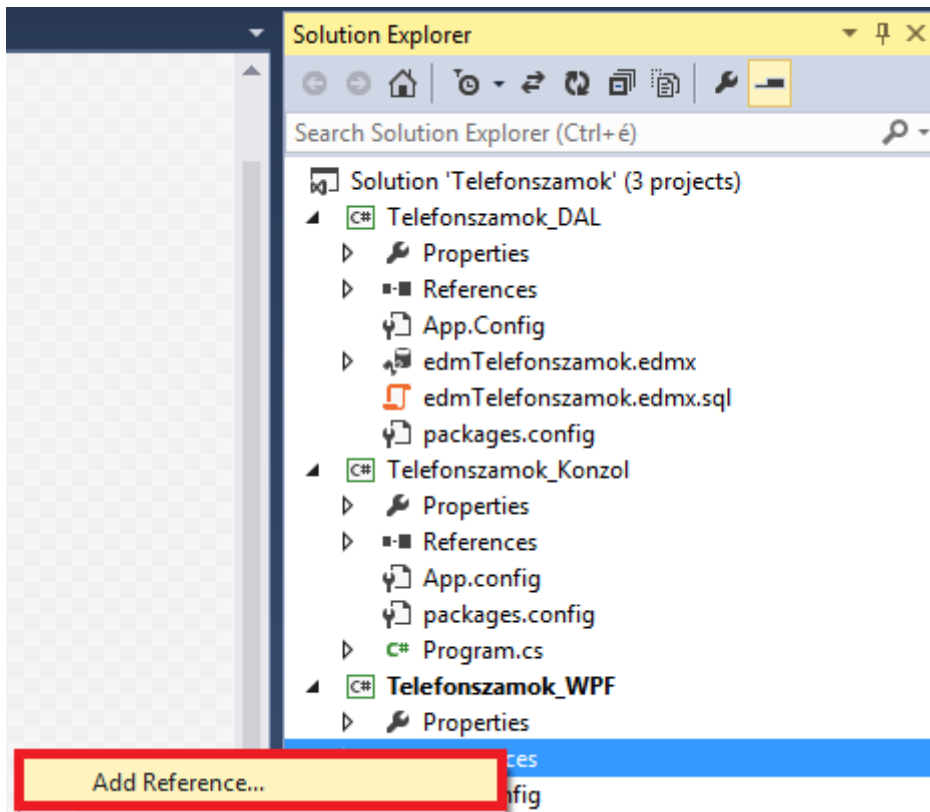




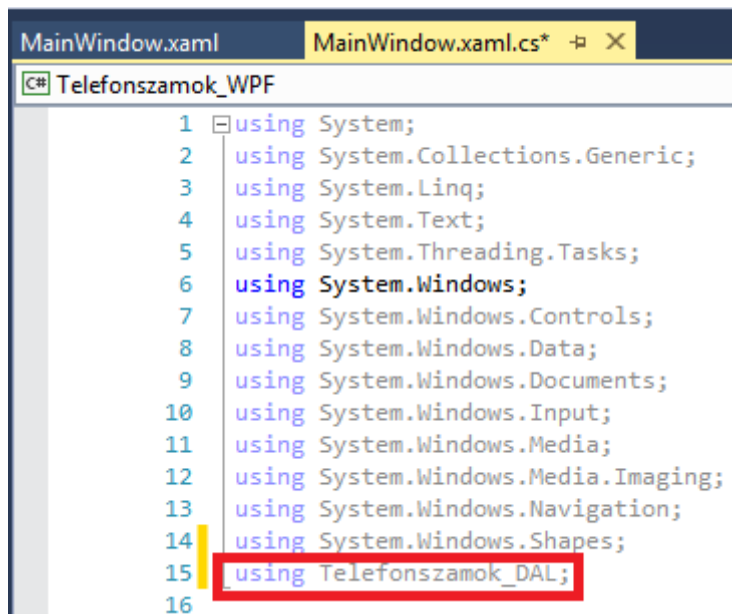
Állítsuk be indító projektté az újonnan létrehozott projektet.



A referenciák között állítsuk be a Telefonszamok_DAL projektet annak érdekének, hogy felhasználhassuk a későbbiekben az ott létrehozott entitás modellt.



Állítsuk be a DAL projekt névterét felhasznált névtérként az ablak C# kódjában (MainWindow.xaml.cs).



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows;
7 using System.Windows.Controls;
8 using System.Windows.Data;
9 using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15 using Telefonszamok_DAL;
16
```

Átmásoljuk az App.Config állományt a konzol projektből az aktuális projektbe, majd az ablak konstruktorában az alábbi két utasítás segítségével beállítjuk, hogy a Telefonszamok_DAL projekt könyvtárát tekintse a program adatkönyvtárnak, itt keresse az adatbázisállományt.

```
var DataDir =
    Directory.GetParent(Directory.GetParent(Directory.GetParent(
        Directory.GetCurrentDirectory()).FullName).FullName) +
    "\\Telefonszamok_DAL";
AppDomain.CurrentDomain.SetData("DataDirectory", DataDir);
```

Engedélyezzük a System.IO.Directory referencia felvételét a referenciák közé.

```
public MainWindow()
{
    InitializeComponent();
    var DataDir =
    Import 'System.IO.Directory' and all other references in file? (Alt+Enter) nt(
    Directory.GetCurrentDirectory()).FullName).FullName) +
    "\\Telefonszamok_DAL";
    AppDomain.CurrentDomain.SetData("DataDirectory", DataDir);
```

Létrehozunk egy adattagot az entitáskonténer számára, majd felvesszük a referenciák közé az Entity Framework-öt.

```

10
17 namespace Telefonszamok_WPF
18 {
19     /// <summary>
20     /// Interaction logic for MainWindow.xaml
21     /// </summary>
22     public partial class MainWindow : Window
23     {
24         private cnTelefonszámok cnTelefonszámok;
25     }
26 }

```

Add reference to 'EntityFramework'

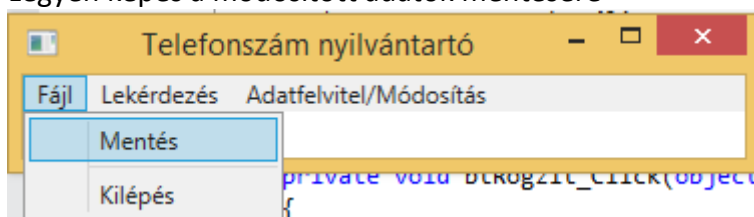
Létrehozuk a konténer objektumot a főablak konstruktorában.

```
cnTelefonszámok=new cnTelefonszámok();
```

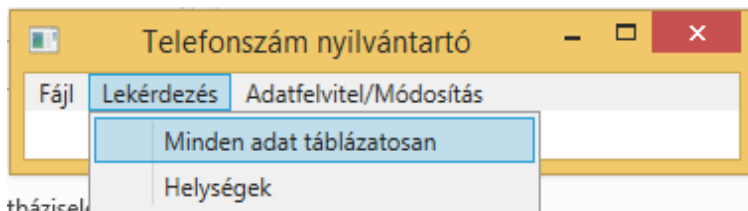
2. A felület elkészítése

Az alkalmazás felületét a megkövetelt funkcionalitás határozza meg. A következő funkcionalitást várjuk el.

1. Legyen képes a módosított adatok mentésére



2. A tárolt adatok legyenek lekérdezhetőek egyben egy minden információt tartalmazó táblázatban.



Vezetéknév	Utónév	Irsz	Helységnév	Lakcím	Telefonszámok
Senki	Alfonz	6000	Kecskemét	Kis út 1.	+36-11-555-5555, +36-12-444-4444
Gipsz	Jakab	6000	Kecskemét	Malom Köz 1.	
Erős	Áron	2038	Sóskút	Alma rét 3.	+36-11-111-1111
Olvasó	Jolán	2039	Pusztazámor	Hermelin sugárút 5.	+36-11-333-3333
Argon	Géza	2090	Remeteszőlós	Órdas köz 6.	+36-13-555-555, +36-13-556-555

A helységekre vonatkozó adatok legyenek lekérdezhetőek egy táblázatban.

Név	Irsz	
Kecskemét	6000	
Sóskút	2038	
Pusztazámor	2039	
Budaörs	2040	
Törökbálint	2045	
Remeteszőlős	2090	

3. A helységadatok legyenek módosíthatóak

A fenti igények megvalósításához StackPanel rétegmenedzsert használunk, amire egy menüt egy DataGrid és egy Grid komponenst fogunk elhelyezni. A Grid segítségével alakítjuk ki a helységadatok megjelenítéséhez szükséges űrlapot. Kezdetben sem a DataGrid sem a Grid nem látható. Az ablak XAML kódja az alábbi.

```
<Window x:Class="Telefonszamok_WPF.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Telefonszám nyilvántartó" Height="350" Width="525">
    <StackPanel>
        <!-- Menü -->
        <Menu>
            <MenuItem x:Name="miFájl" Header="Fájl" >
                <MenuItem x:Name="miMentés" Header="Mentés" Click="miMentés_Click"/>
                <Separator/>
                <MenuItem x:Name="miKilépés" Header="Kilépés" Click="miKilépés_Click"/>
            </MenuItem>
            <MenuItem x:Name="miLekérdezés" Header="Lekérdezés">
                <MenuItem x:Name="miMindenAdat" Header="Minden adat táblázatosan"
                    Click="miMindenAdat_Click" />
            </MenuItem>
        </Menu>
    </StackPanel>
</Window>
```

```

        <MenuItem x:Name="miHelységek" Header="Helységek"
            Click="miHelységek_Click" />
    </MenuItem>
    <MenuItem x:Name="miAdatfelvitelMódosítás" Header="Adatfelvitel/Módosítás">
        <MenuItem x:Name="miHelységekAM" Header="Helységek"
            Click="miHelységekAM_Click"/>
    </MenuItem>
</Menu>
<!-- Adatrács -->
<DataGrid x:Name="dgAdatrács" ItemsSource="{Binding}" Visibility="Hidden"/>
<!-- Helységadatok megjelenítése és módosítása -->
<Grid x:Name="grHelység" Visibility="Hidden" Margin="0,10,0,0"
    DataContext="{Binding}">
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Label Content="Keresés irányítószám alapján" Grid.Row="0" Grid.Column="0"/>
    <ComboBox x:Name="cbIrsz" ItemsSource="{Binding}" DisplayMemberPath="Irsz"
        Grid.Row="0" Grid.Column="1" Margin="5"
        SelectionChanged="cbIrsz_SelectionChanged"
        IsSynchronizedWithCurrentItem="True"/>
    <Label Content="Keresés helységnév alapján" Grid.Row="1" Grid.Column="0" />
    <ComboBox x:Name="cbHelységnév" ItemsSource="{Binding}"
        DisplayMemberPath="Név"
        Grid.Row="1" Grid.Column="1" Margin="5"
        SelectionChanged="cbIrsz_SelectionChanged"
        IsSynchronizedWithCurrentItem="True"/>
    <Label Content="Irányítószám" Grid.Row="2" Grid.Column="0"/>
    <TextBox x:Name="tbIrsz" Grid.Row="2" Grid.Column="1" Margin="5"/>
    <Label Content="Helységnév" Grid.Row="3" Grid.Column="0"/>
    <TextBox x:Name="tbHelységnév" Grid.Row="3" Grid.Column="1" Margin="5"/>
    <StackPanel Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2"
        Orientation="Horizontal" HorizontalAlignment="Center">
        <Button x:Name="btRögzít" Content="Módosított adatszám rögzítése"
            Click="btRögzít_Click" Margin="50,50,10,10"/>
        <Button x:Name="btÚjHelység" Content="Új helység"
            Margin="50,50,10,10" Click="btÚjHelység_Click"/>
        <Button x:Name="btVissza" Content="Vissza"
            Click="btVissza_Click" Margin="50,50,10,10" />
    </StackPanel>
</Grid>
</StackPanel>
</Window>

```

Minden olyan menüponthoz, amelyből nem nyílik almenü, egy eseménykezelőt készítünk a Properties ablak események (Events) funkciójával.

A Mentés menüpont eseménykezelőjében elmentjük az adatbázisba az eddig végrehajtott módosításokat.

```

private void miMentés_Click(object sender, RoutedEventArgs e)
{

```



```

        cnTelefonszámok.SaveChanges();
    }

```

A Kilépés menüpont eseménykezelőjében kilépünk a programból.

```

private void miKilépés_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}

```

3. Egyszerű lekérdezés

A helységadatok megjelenítésénél egy táblázatot szeretnénk látni, amelyben a helységek nevei és irányítószámaik láthatóak. Az eseménykezelőben először láthatóvá tesszük az adatrácsot, majd végrehajtjuk a lekérdezést és végül a lekérdezés eredményét adatkötéssel kapcsoljuk a rácshoz. A lekérdezés eredményét egy listává alakítjuk, mert csak ekkor fog ténylegesen végrehajtódni a LINQ lekérdezés.

```

private void miHelységek_Click(object sender, RoutedEventArgs e)
{
    grHelység.Visibility=Visibility.Hidden;
    dgAdatrács.Visibility=Visibility.Visible;
    var er = (from x in cnTelefonszámok.enHelységek
              select new {x.Név, x.Irsz}).ToList();
    dgAdatrács.ItemsSource = er;
}

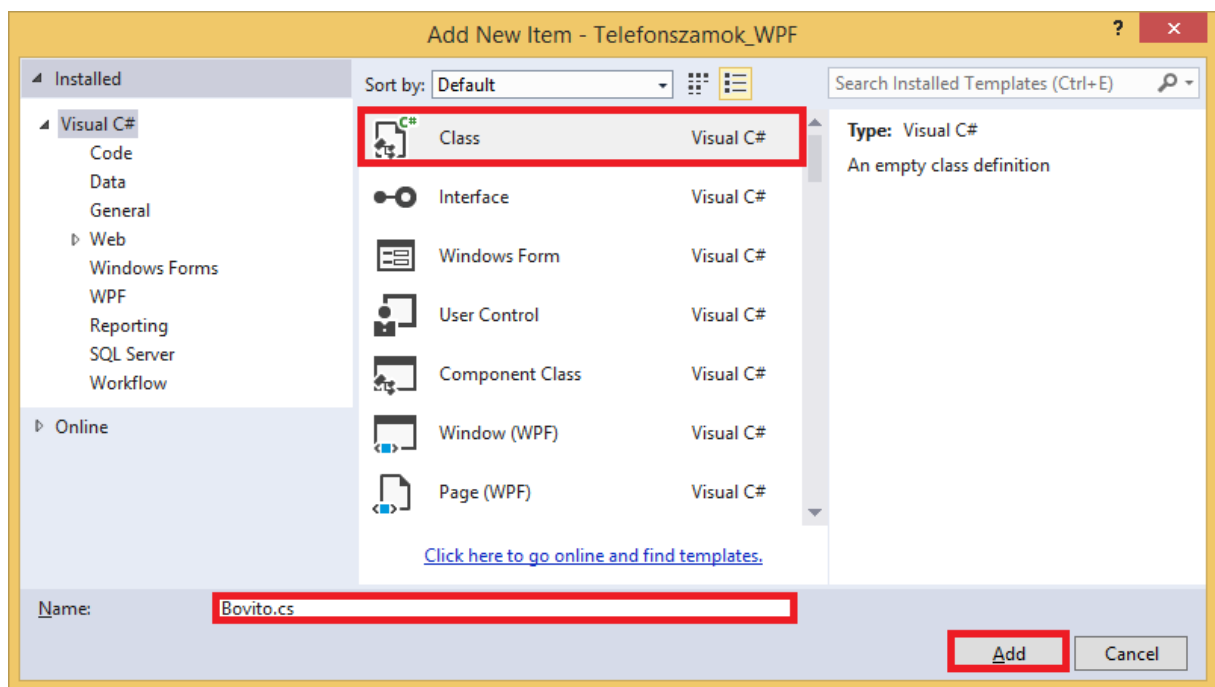
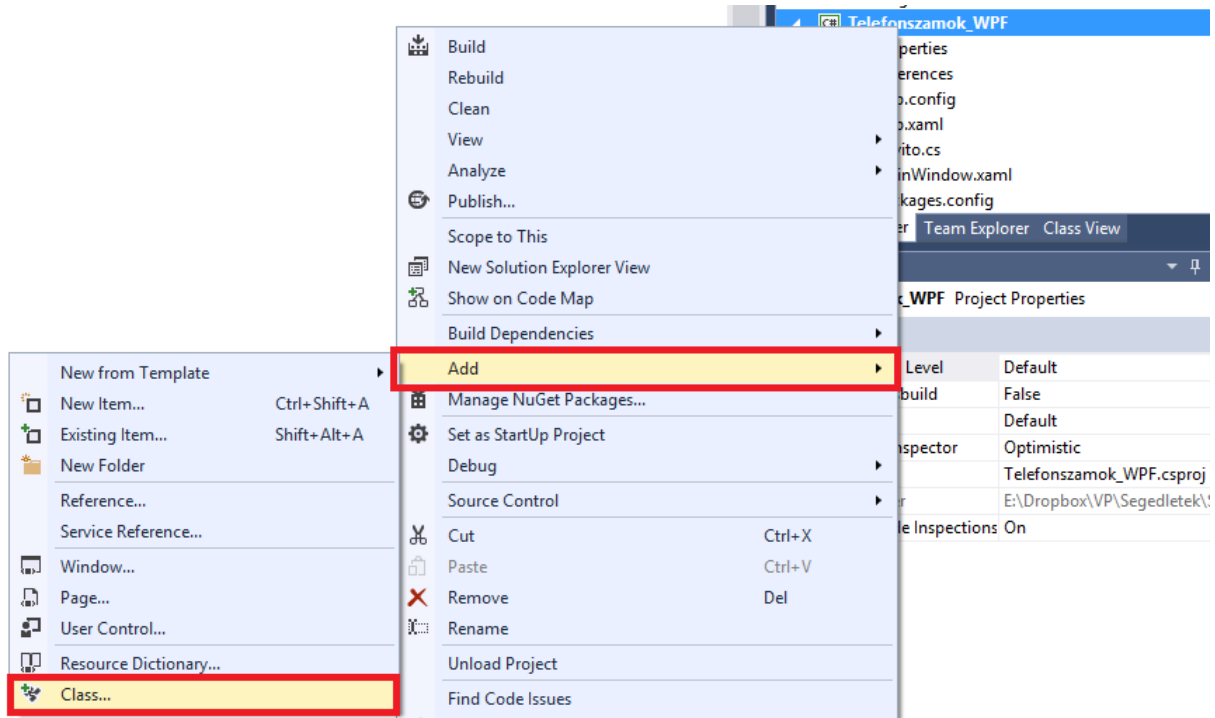
```

Fájl Lekérdezések	
Név	Irsz
Kecskemét	6000
Sóskút	2038
Pusztazámor	2039
Budaörs	2040
Törökbálint	2045
Remeteszőlős	2090

Próbáljuk ki a kódot úgy is, hogy elhagyjuk a ToList() metódushívást.

4. Komplex lekérdezés

Az összes adat megjelenítésénél táblázatos formában szeretnénk látni minden tárolt adatot. Ennek érdekében egy olyan gyűjteményt kell előállítanunk, aminek egy eleme egy személy összes adatát tartalmazza. Mivel egy személyhez több telefonszám is tartozik, ezért azt szeretnénk elérni, hogy ezek a számok egyetlen sztringben, egymástól vesszővel elválasztva jelenjenek meg. Ezt a részfeladatot úgy oldjuk meg, hogy készítünk egy bővítő metódust az enSzemély entitás osztályhoz a CustomExtensions névtérben. A bővítő metódust egy Bővítő nevű statikus osztályban helyezzük el.



A Telefonszámok bővítő metódusban megkapjuk egy enSzemély entitás referenciáját, és végigiterálunk a kapcsolódó telefonszám információkat tartalmazó entításokon. Az utolsó kivételével mindegyik után teszünk egy vesszőt. A metódus visszaadja a telefonszámok listáját tartalmazó sztringet.

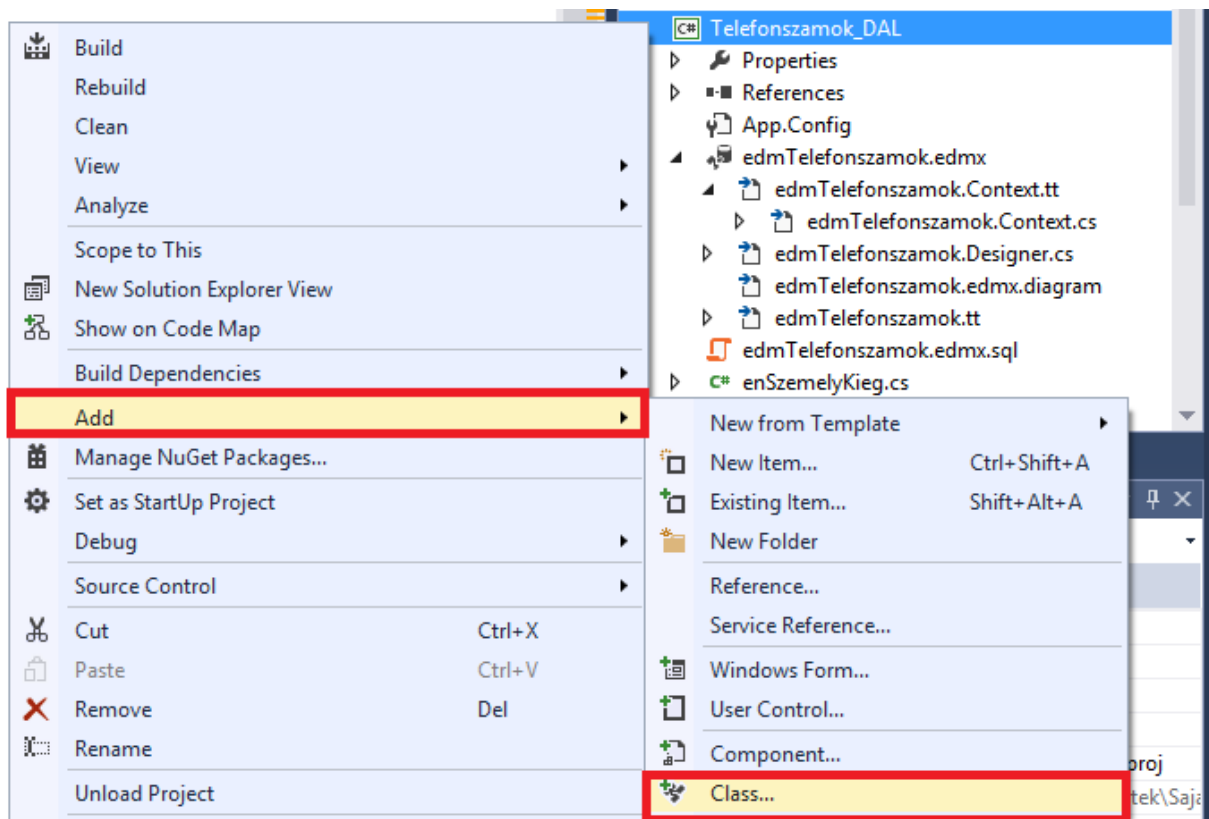
```
using Telefonaszamok_DAL;
namespace CustomExtensions
{
    public static class Bővítő
```

```

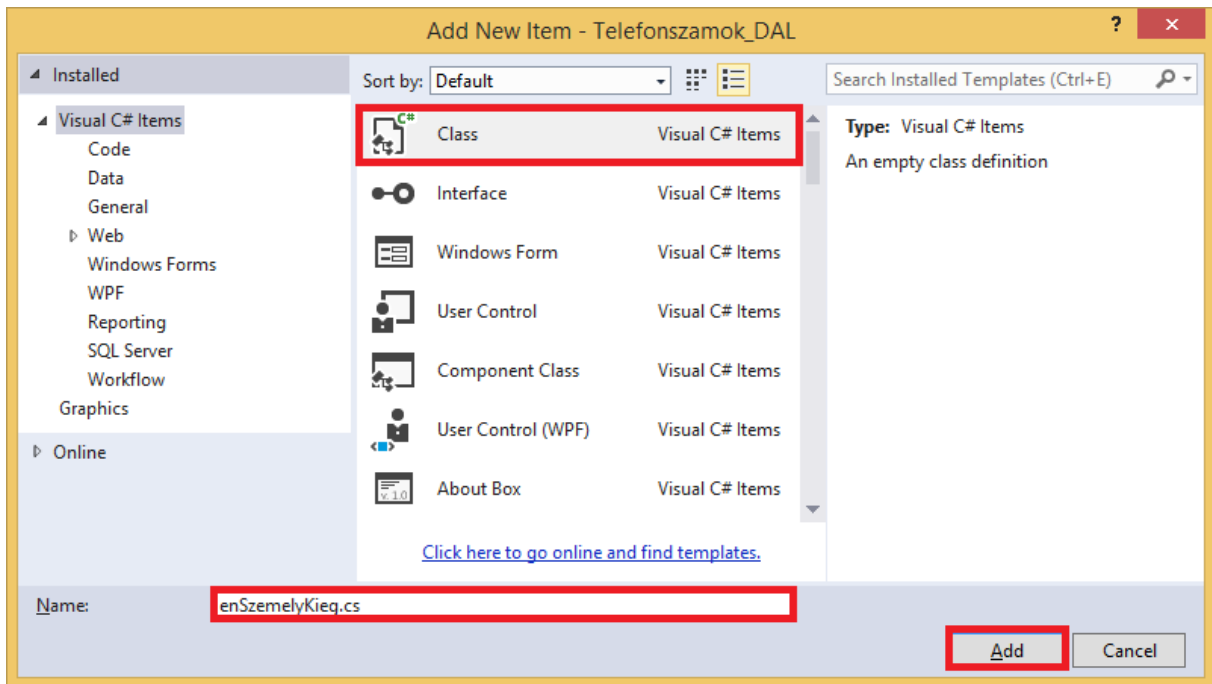
{
    public static string Telefonszámok(this enSzemély enSzemély)
    {
        var s = "";
        foreach (var x in enSzemély.enTelefonszámok)
        {
            s = s + x.Szám;
            if (x != enSzemély.enTelefonszámok.Last())
                s = s + ", ";
        }
        return s;
    }
}

```

A telefonszámlista előállításának egy másik útja az is lehetne, hogy a Telefonszámok_DAL projektben létrehozunk a személyeket leíró enSzemély entitás osztályhoz egy kiegészítést (partial class), amiben egy új tulajdonságot definiálunk Telefonszámok néven. Ehhez a Solution Explorerben egy új osztály adunk a Telefonszámok_DAL projekthez



A létrehozáskor az enSzemelyKieg nevet adjuk az új osztálynak, azonban a C# kódban ezt átírjuk.



A tulajdonság csak get elérővel rendelkezik, és a lekérdezéshez tartozó kód gyakorlatilag azonos az előzőekben ismertetettel.

```
namespace Telefonszamok_DAL
{
    public partial class enSzemély
    {
        public string Telefonszámok
        {
            get
            {
                var s = "";
                foreach (var x in enTelefonszámok)
                {
                    s = s + x.Szám;
                    if(x!= enTelefonszámok.Last())
                        s=s+ ", ";
                }
                return s;
            }
        }
    }
}
```

Ennek a megoldásnak az az előnye, hogy a későbbiekben a Telefonszámok tulajdonságra ugyanúgy tudunk hivatkozni, mintha az az entitás egy tárolt tulajdonsága lenne.

Visszatérve a MainWindow osztály miMindenAdat_Click metódusához a következő feladatunk az, hogy előállítsuk azt a gyűjteményt, amit adatkötéssel az adatrácshoz kívánunk rendelni. Első ötletként az alábbi kódrészlet kínálkozik elegáns megoldásként. Egy LINQ lekérdezés, ami kihasználja az előzőekben alternatív megoldásként bemutatott Telefonszámok tulajdonságot.

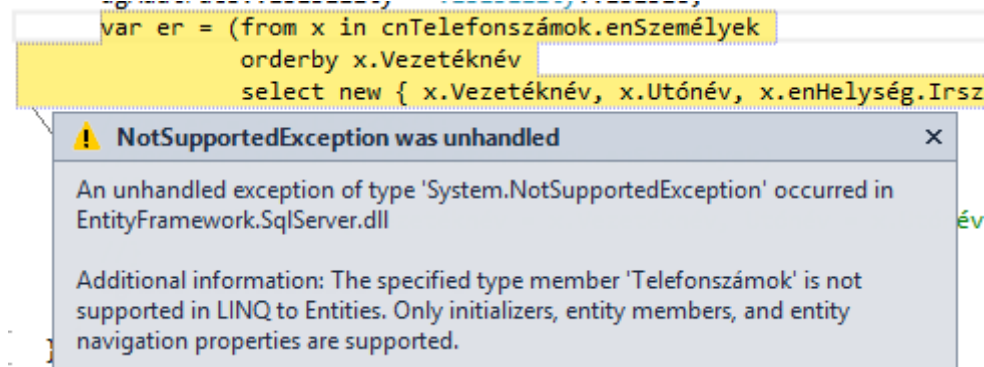
```
var er = (from x in cnTelefonszámok.enSzemélyek
```

```

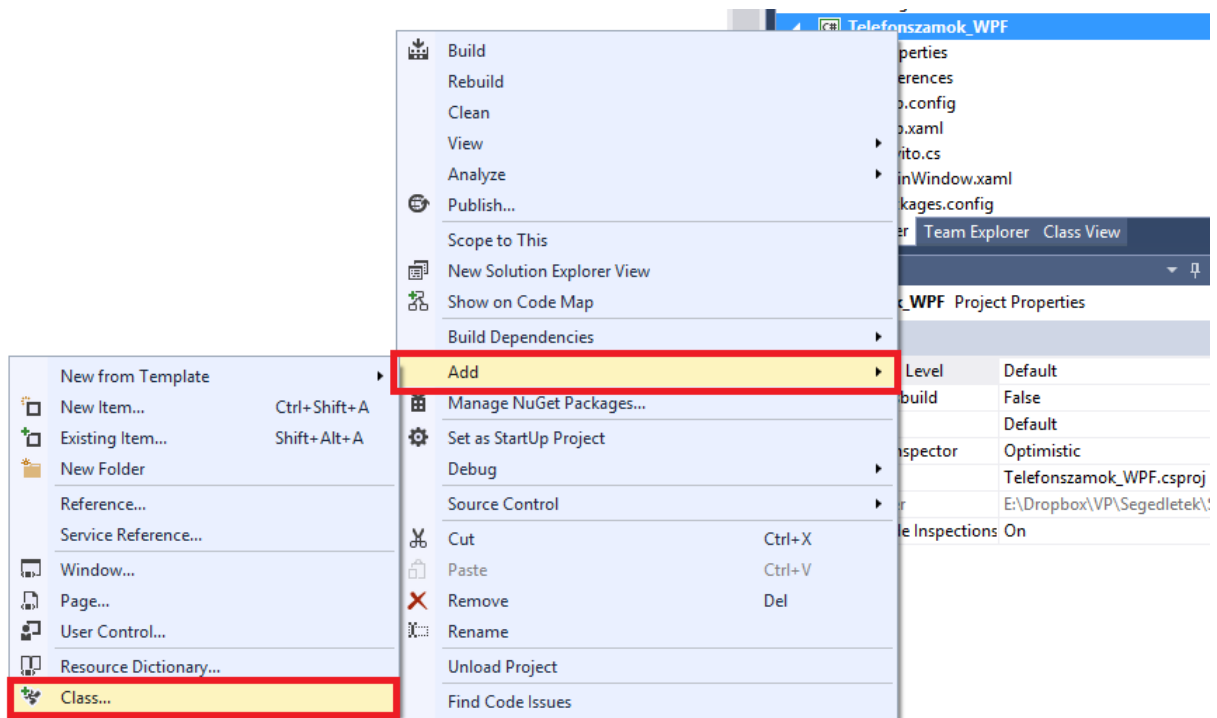
orderby x.Vezetéknév
select new
{
    x.Vezetéknév, x.Utónév, x.enHelység.Irsz,
    x.enHelység.Név, x.Lakcím, x.Telefonszámok }).ToList();

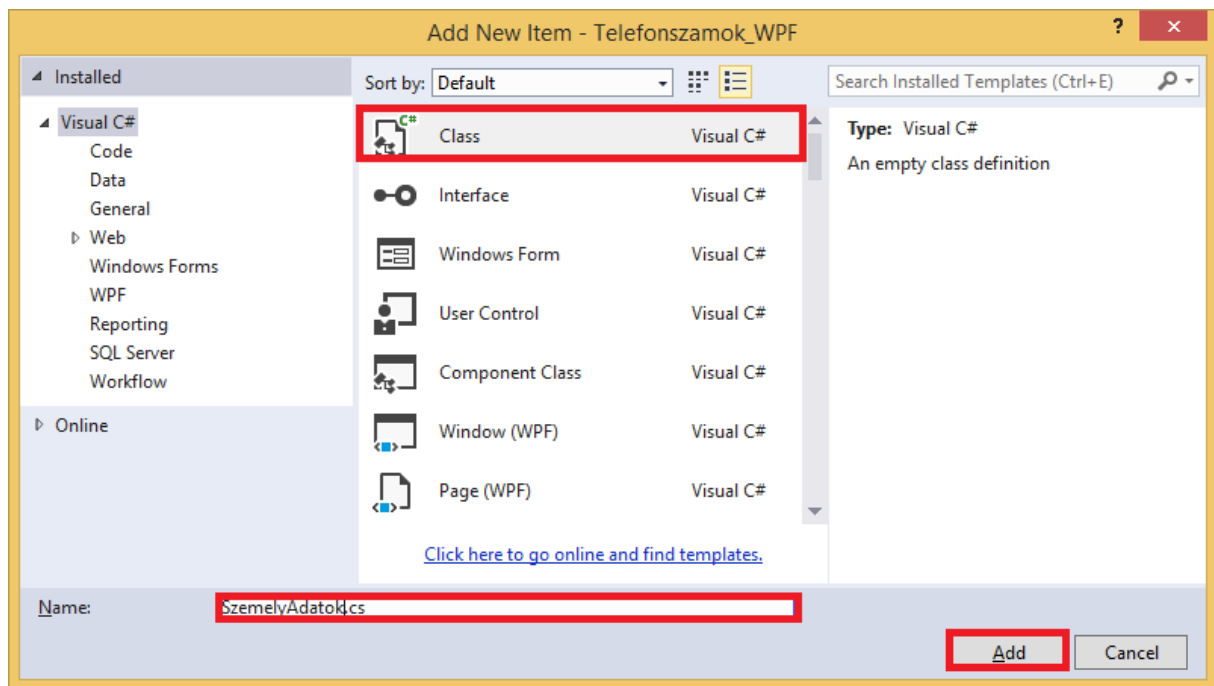
```

Futtatáskor azonban azt kell tapasztalnunk, hogy az Entity Framework 6.1 változatához kapcsolódó LINQ to Entities a fentiekben ismertetett két megoldás egyikét se támogatja LINQ lekérdezésben.



Mivel LINQ-val nem tudjuk célunkat elérni, ezért egy foreach ciklussal haladunk végig a személy entitásokat tartalmazó gyűjteményen. LINQ nélkül az eredmény gyűjtemény előállításához először definiálnunk kell egy segédosztályt (SzemélyAdatok).





```
public class SzemelyAdatok
{
    public string Vezetéknév { get; set; }
    public string Utónév { get; set; }
    public Int16 Irsz { get; set; }
    public string Helységnév { get; set; }
    public string Lakcím { get; set; }
    public string Telefonszámok { get; set; }
}
```

A MainWindow osztály miMindenAdat_Click metódusában létrehozunk egy típusos lista objektumot, amelynek elemtípusa SzemelyAdatok, majd az entitás halmazon végighaladva, minden személyhez létrehozunk egy SzemelyAdatok típusú névtelen objektumot, amit hozzáadunk a listához.

```
private void miMindenAdat_Click(object sender, RoutedEventArgs e)
{
    grHelység.Visibility=Visibility.Hidden;
    dgAdatrács.Visibility = Visibility.Visible;
    var er = new List<SzemelyAdatok>();
    foreach (var x in cnTelefonszámok.enSzemélyek)
    {
        er.Add(
            new SzemelyAdatok()
            {
                Vezetéknév = x.Vezetéknév, Utónév = x.Utónév,
                Helységnév = x.enHelység.Név, Irsz = x.enHelység.Irsz,
                Lakcím = x.Lakcím, Telefonszámok = x.Telefonszámok()
            });
    }
    dgAdatrács.ItemsSource = er;
}
```

A programot futtatva a Lekérdezések menü Minden adat táblázatosan menüpontját választva az alábbi ábrán látható táblázatot kapjuk.

Telefonszám nyilvántartó					
Fájl Lekérdezések					
Vezetéknév	Utónév	Irsz	Helységnév	Lakcím	Telefonszámok
Senki	Alfonz	6000	Kecskemét	Kis út 1.	+36-11-555-5555, +36-12-444-4444
Gipsz	Jakab	6000	Kecskemét	Malom Köz 1.	
Erős	Áron	2038	Sóskút	Alma rét 3.	+36-11-111-1111
Olvasó	Jolán	2039	Pusztazámor	Hermelin sugárút 5.	+36-11-333-3333
Argon	Géza	2090	Remeteszőlős	Ordas köz 6.	+36-13-555-555, +36-13-556-555

5. Helységadatok módosítása

A helységadat adatfelvitel/módosítás során a felhasználó két funkció közül választhat. Ezek az adatbázisban szereplő adatok módosítása és új adatok felvétele.

Módosítás esetén a két legördülő kombinált lista egyikével kiválasztja az aktuális települést, ezek adatai megjelennek a TextBox komponensekben, majd a módosítást követően kattint a „Módosított adatpár rögzítése” feliratú gombon.

Telefonszám nyilvántartó	
Fájl Lekérdezés Adatfelvitel/Módosítás	
Keresés irányítószám alapján	6000
Keresés helységnév alapján	Kecskemét
Irányítószám	6000
Helységnév	Kecskemét
<input type="button" value="Módosított adatpár rögzítése"/> <input type="button" value="Új helység"/> <input type="button" value="Vissza"/>	

A kívánt adatok megjelenítéséhez a láthatóság beállítását követően az adatok listáját adatkötéssel a Grid-hez rendeljük, majd kiválasztjuk az első helységet.

```
private void miHelységekAM_Click(object sender, RoutedEventArgs e)
{
    dgAdatrács.Visibility = Visibility.Collapsed;
    grHelység.Visibility = Visibility.Visible;
    grHelység.DataContext = cnTelefonszámok.enHelységek.ToList();
    cbIrsz.SelectedIndex = 0;
}
```

A két ComboBox együtt fog változni a „IsSynchronizedWithCurrentItem=True” attribútum használatának köszönhetően. Ezért elegendő hozzájuk egy közös eseménykezelő

(cbIrsz_SelectionChanged) készítése. Amikor bármelyikben változik a kiválasztott elem, akkor a két szövegmező tartalmát frissítjük.

```
private void cbIrsz_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    var enAktuális = ((ComboBox) sender).SelectedItem as enHelység;
    cbHelységnev.SelectedItem = enAktuális;
    tbIrsz.Text = enAktuális.Irsz.ToString();
    tbHelységnev.Text = enAktuális.Név;
}
```

A Rögzít gombon történő kattintáskor a gyűjteményben módosítjuk az adatokat.

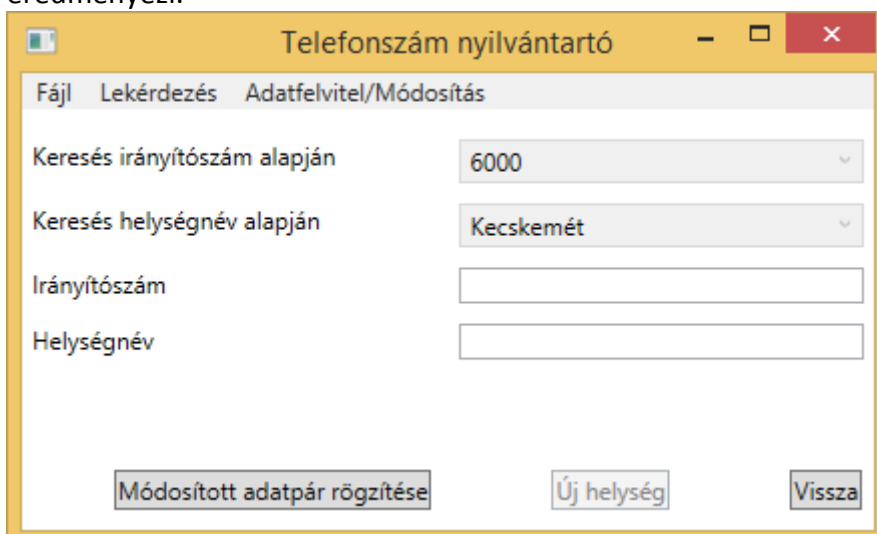
```
private void btRögzít_Click(object sender, RoutedEventArgs e)
{
    var enAktuális = cbIrsz.SelectedItem as enHelység;
    //
    // Ide ellenőrzés kellene az adatok rögzítése előtt.
    // ...
    //
    enAktuális.Irsz = Int16.Parse(tbIrsz.Text);
    enAktuális.Név = tbHelységnev.Text;
    grHelység.Visibility = Visibility.Hidden;
}
```

A fenti metódus nem tartalmazza a szövegmezőbe írt adatok ellenőrzését, ez az olvasó önálló feladata. A gyűjteménybe történő felvétel még nem jelenti az adatbázisban történő tárolást. Ez utóbbi csak a Fájl menü Mentés pontjának választása után történik meg.

A Vissza gombon történő kattintás hatására eltüntetjük az űrlapot.

```
private void btVissza_Click(object sender, RoutedEventArgs e)
{
    grHelység.Visibility=Visibility.Hidden;
}
```

Amennyiben a felhasználó új helységadatokat akar felvinni az adatbázisba, akkor az „Új helység” gombon kattint, ami a két ComboBox letiltását és a szövegmezők ürítését eredményezi.




```
private void btÚjHelység_Click(object sender, RoutedEventArgs e)
{
    Beállít(false);
    tbIrsz.Text = "";
    tbHelységnev.Text = "";
}
```

```
private void Beállít(bool b)
{
    btÚjHelység.IsEnabled = b;
    cbIrsz.IsEnabled = b;
    cbHelységnev.IsEnabled = b;
}
```

A fentiekben ismertetett `btRögzít_Click` metódust kis módosítással alkalmassá tesszük az új adatok rögzítésére is. Az új változatban létrehozunk egy új `enHelység` objektumot, majd felvesszük azt a helységek gyűjteményébe.

```
private void btRögzít_Click(object sender, RoutedEventArgs e)
{
    var enAktuális = cbIrsz.SelectedItem as enHelység;
    //
    // Ide ellenőrzés kellene az adatok rögzítése előtt.
    // ...
    //
    if (!btÚjHelység.IsEnabled)
    {
        enAktuális = new enHelység();
        cnTelefonszámok.enHelységek.Add(enAktuális)
    }
    enAktuális.Irsz = Int16.Parse(tbIrsz.Text);
    enAktuális.Név = tbHelységnev.Text;
    grHelység.Visibility = Visibility.Hidden;
}
```

Hasonlóan a módosítás esetéhez, tartós tárolás itt is csak a Mentés menüpont segítségével érhető el.

Tartalomjegyzék

Entity Framework alapú adatbáziselérés 2	1
1. Projekt és alapbeállítások.....	1
2. A felület elkészítése.....	6
3. Egyszerű lekérdezés	9
4. Komplex lekérdezés.....	9
5. Helységadatok	15