

- WINDOWS APPLICATION PROGRAMMING -

PERSONAL PROJECT - part 1

Run the DEMO app to see **appearance** and **Behavior** final application windows.

STEP 1. Create the project in Visual Studio

Visual Studio is started and the command is given to create a new type project **WPF App (.NET Framework)**, in the C # language.

Project Name depends on the student's first and last name. For example, student Pop George will create the project **PopGeorge2021 Project**, and the Visual Studio solution will be named **ProjectPopGeorge2021_CS**. In the text below, the name of the project will be written **DEMO project**.

STEP 2. Create the database, tables and insert the data into the tables

Using SQL Server Management Studio (SSMS) creates a new database.

Database name depends on the student's first and last name. For example, student Pop George will create the database named **PopGeorge2021 Project**. (In this text, the base will be named **DEMO project** !) Then, the SQL script given by the teacher is executed to create the data tables.

When submitting the project, [the database must also be submitted](#) as an SQL file exported from SSMS.

STEP 3. Install the NuGet EntityFramework package in the project This operation is done as in the previous paper on **Entity Framework**.

STEP 4. Create a database connection

This operation is done as in the previous paper on **Entity Framework**, but using the database created in STEP 2.

STEP 5. Create the data model using the previously created database

This operation is done as in the previous paper on **Entity Framework**, but using the database created in STEP 2. In the data model **Entity Framework** only the tables required for the current application must be available (here, only the table is required **Employees**).

STEP 6. Create the main window that will display the data as a table

A control must be used in the data display window **DataGrid**. How to work with this control was presented in a previous paper, **UsingDataGrid**.

Depending on the table in the database, enter the XAML code to display the item **DataGrid**.

STEP 7. Fill in the code to use the previously created data model

A private method is written for uploading data specifically for this purpose. Usually, it's called **LoadData**. Method code **LoadData** is analogous to the code in the previous paper **UsingDataGrid**.

This method is used when the event occurs **Click** on the button **Load Data**.

STEP 8. Filter the displayed data as a table

The DEMO application shows how to filter the displayed data. Proceed as in the previous work **UsingDataGrid**. See XAML and C # code there. In class **MainWindow** the variables are defined:

```
private ProjectDEMOContext context = null; private List<Employee> EmployeeList = null; private CollectionViewSource  
employeeViewSource = null;
```

the student is not ProjectDEMO (see Step 2)

LINQ statement in the method **LoadData** must be adapted (see next image).

```
private void LoadData()
{
    try
    {
        // se extrage sirul de caractere introdus de utilizator in TextBox
        string countryFilter = textBoxCountryFilter.Text.Trim();

        // se creeaza un nou context de date
        if (context == null) context = new ProjectDEMOContext();

        // Load este o metodă de extensie definită în spațiul de nume System.Data.Entity.
        // Această metodă enumeră rezultatele interogării,
        // similar cu ToList dar fără a crea o listă.
        // Când este utilizată cu LINQ to Entities, această metodă
        // creează obiectele entitate și le adaugă în context.
        // VEZI: https://docs.microsoft.com/en-us/ef/ef6/fundamentals/databinding/wpf

        context.Employees.Load();

        // După ce datele sunt încărcate, se apelează proprietatea "Local"
        // pentru a utiliza un DbSet<T> ca sursă pentru DataBinding.

        var query = from emp in context.Employees.Local
                     where emp.Country.StartsWith(countryFilter, StringComparison.InvariantCultureIgnoreCase)
                     select emp;

        // se extrage lista de angajati (vizibila in tot codul clasei)
        EmployeeList = query.ToList();

        // lista de angajati este sursa pentru CollectionViewSource
        employeeViewSource.Source = EmployeeList;

        // lista de angajati ajunge in contextul de date al ferestrei
        this.DataContext = employeeViewSource;
    }
    catch (Exception ex)
    {
        string caption = "Loading data ...";

        string message = null;
        string exception = null;
        string innerException = null;
        string connectionString = null;

        exception = "EXCEPTION!\n" + ex.Message;
        if (ex.InnerException != null)
            innerException = "INNER EXCEPTION:\n" + ex.InnerException.Message
                             + "\nSOURCE: \n" + ex.InnerException.Source;
        connectionString = "CONNECTION STRING: \n"
                             + context.Database.Connection.ConnectionString;

        message = $"{exception}\n\n{innerException}\n\n{connectionString}";

        MessageBox.Show(message, caption, MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Fig.1. LoadData method code (MainWindow class)

STEP 9. Display the window with details about the selected item in the table

Run the DEMO application to see how the current employee details are displayed in the table. Proceed as in the previous work **UsingDataGrid** (part 2). But there are also new elements.

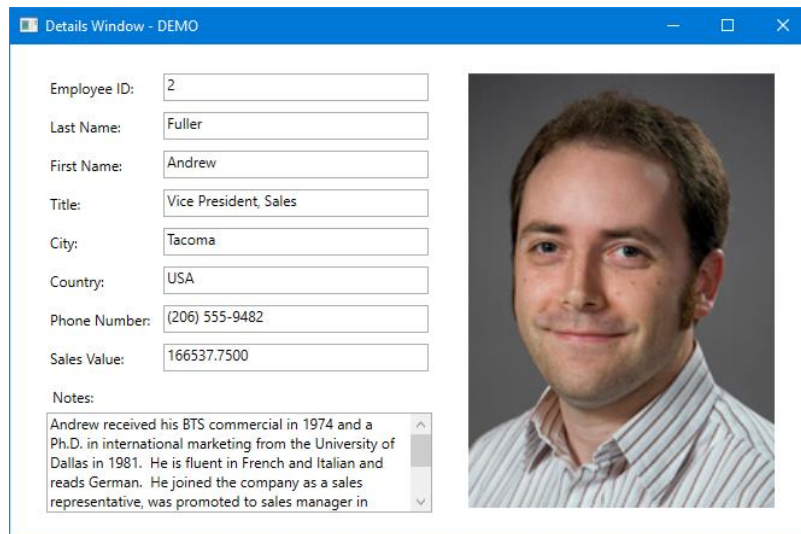


Fig.2. An employee details window

In the database, the table of employees has a column **Notes**. Here are important details (important details) about each employee. A control must be provided to display this text in the window **TextBox** with multiple lines of text and with **Scrollbar** vertical (appears automatically if necessary).

STEP 10. Display the chart with the total sales made by employees

Run the DEMO application to see how the graphical data window (total sales per employee) is displayed. These graphs are displayed in a separate window.

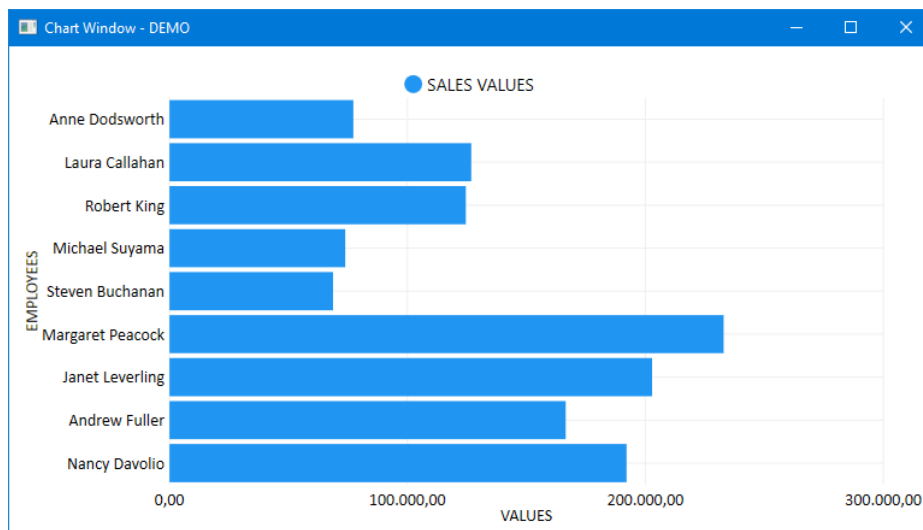


Fig.3. Sales chart window

Proceed as in the previous work **UsingCharts**. But there the values were kind **double**, here are the type **decimal**. A slight change should be made in the code of the method preparing the data:

```
SeriesCollection.Add (new RowSeries {  
    Title = title,  
    Values = new ChartValues <decimal> (values)  
});
```



USEFUL DETAILS

```

1  using System;
2  using System.Collections.Generic;
3  using System.Data.Entity;
4  using System.Linq;
5  using System.Windows;
6  using System.Windows.Data;
7  using System.Windows.Input;
8
9  namespace ProjectDEMO
10 {
11     public partial class MainWindow : Window
12     {
13         private ProjectDEMOContext context = null;
14         private List<Employee> EmployeeList = null;
15         private CollectionViewSource employeeViewSource = null;
16
17         public MainWindow()
18         {
19             InitializeComponent();
20
21             buttonLoadData.Focus();
22
23             // se determina obiectul de tip CollectionViewSource declarat in codul XAML
24             // (este necesar casting !)
25             employeeViewSource = (CollectionViewSource)this.FindResource("employeeViewSource");
26         }
27
28         #region ----- METODE PRIVATE -----
29
30         private void LoadData()...
31
32         private void ShowDetails()...
33
34         private void ShowRowChart()...
35
36         #endregion
37
38         #region ----- EVENIMENTE -----
39
40         private void buttonLoadData_Click(object sender, RoutedEventArgs e)...
```

Fig.4. MainWindow class code structure

```

<StackPanel Orientation="Horizontal" Background="Ivory" Margin="0,50,0,0" Height="50" VerticalAlignment="Top" >
    <StackPanel.Resources>
        <Style TargetType="Button">
            <Setter Property="Width" Value="150" />
            <Setter Property="Margin" Value="10" />
            <Setter Property="BorderThickness" Value="0" />
            <Setter Property="Background" Value="#EEEEEE" />
        </Style>
    </StackPanel.Resources>

    <TextBlock Text="Filter by Country:" FontSize="12" Margin="10,0,0,0" VerticalAlignment="Center" />

    <TextBox x:Name="textBoxCountryFilter" Width="50" Padding="4" Margin="10,0,10,0" VerticalAlignment="Center"/>

    <Button x:Name="buttonLoadData" Content="Load Data" IsDefault="True" Click="buttonLoadData_Click" />

    <Button x:Name="buttonShowDetails" Content="Show Details" Click="buttonShowDetails_Click"/>

    <Button x:Name="buttonShowChart" Content="Show Chart" Click="buttonShowChart_Click"/>
</StackPanel>
```

Fig.5. The StackPanel with application buttons may also contain a button style