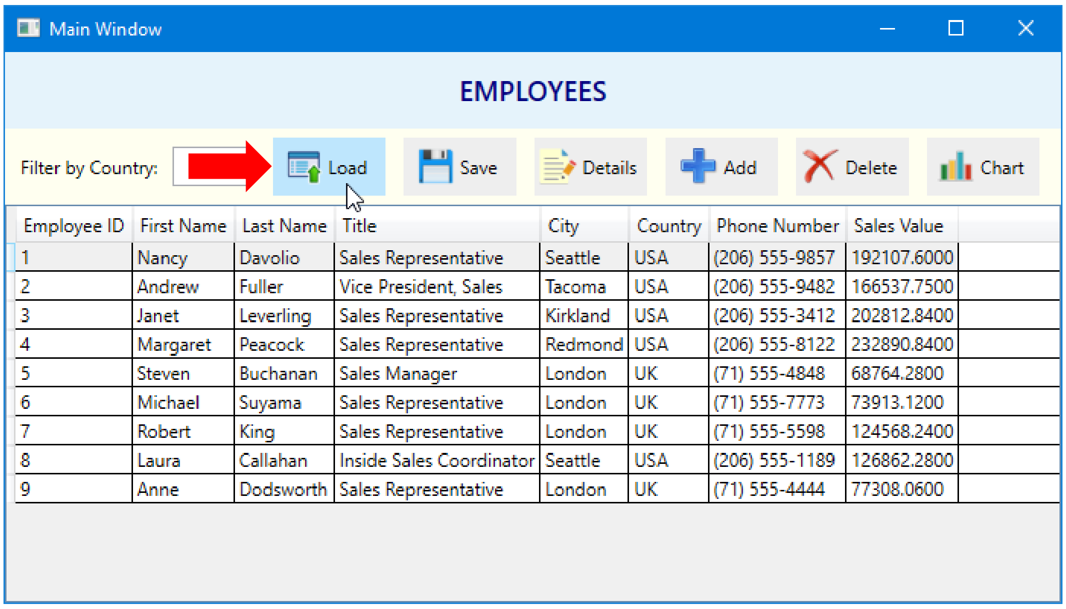
- WINDOWS APPLICATION PROGRAMMING -
PERSONAL PROJECT - part 2

In this step, new features will be added to the Windows application started in Part 1:

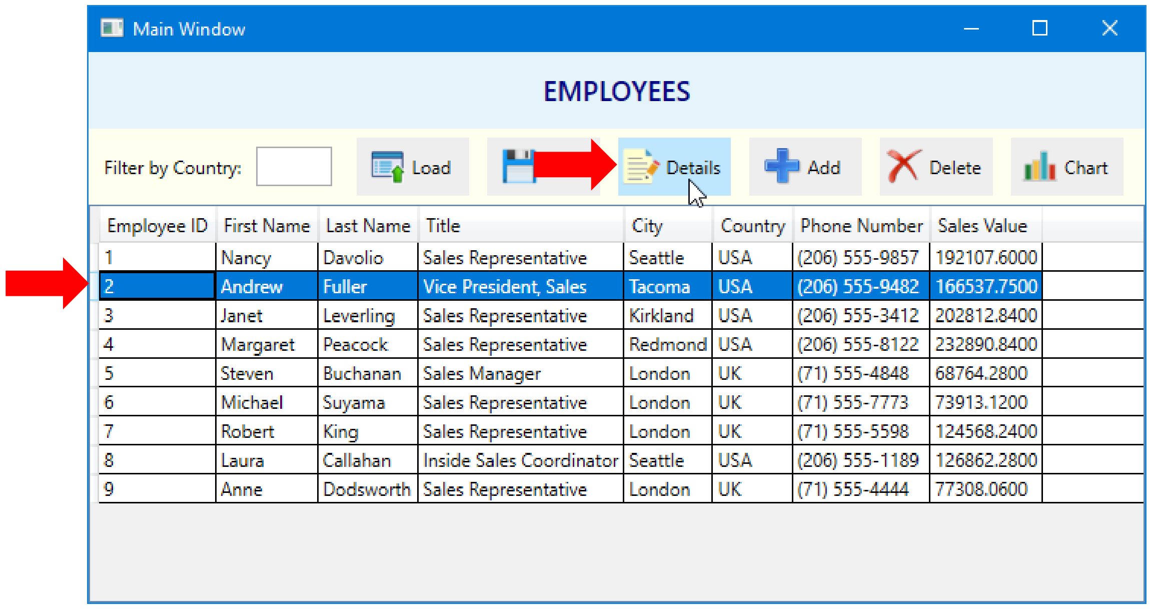
- Possibility to add a new object using a specialized window for details;
- Ability to change existing images by uploading external files;
- Ability to delete a row from the table, but only after user confirmation;
- Saving data to the database (including images);

How the application works, at the end, is shown in the following images:



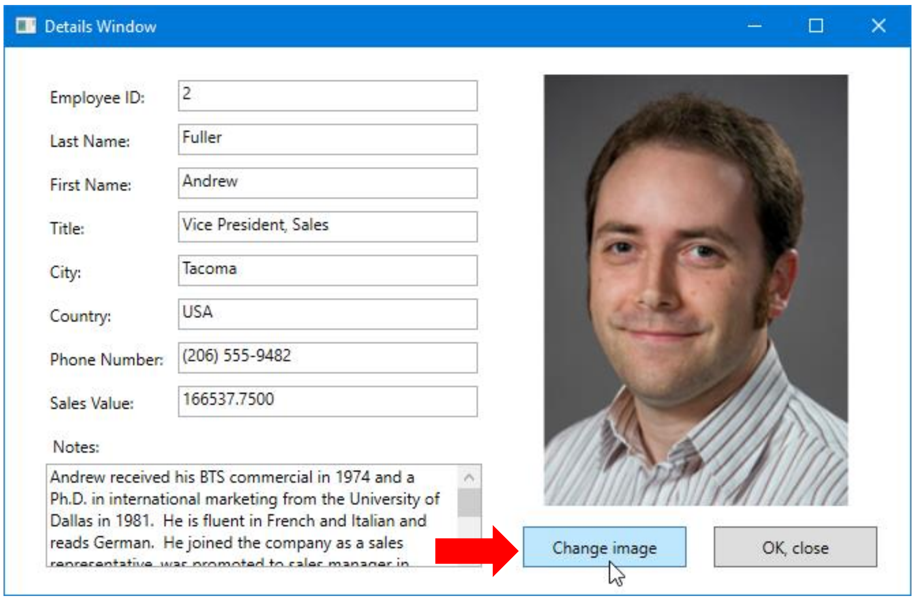
Employee ID	First Name	Last Name	Title	City	Country	Phone Number	Sales Value
1	Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857	192107.6000
2	Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482	166537.7500
3	Janet	Leverling	Sales Representative	Kirkland	USA	(206) 555-3412	202812.8400
4	Margaret	Peacock	Sales Representative	Redmond	USA	(206) 555-8122	232890.8400
5	Steven	Buchanan	Sales Manager	London	UK	(71) 555-4848	68764.2800
6	Michael	Suyama	Sales Representative	London	UK	(71) 555-7773	73913.1200
7	Robert	King	Sales Representative	London	UK	(71) 555-5598	124568.2400
8	Laura	Callahan	Inside Sales Coordinator	Seattle	USA	(206) 555-1189	126862.2800
9	Anne	Dodsworth	Sales Representative	London	UK	(71) 555-4444	77308.0600

Fig.1. The data uploaded from the database is displayed as a table



Employee ID	First Name	Last Name	Title	City	Country	Phone Number	Sales Value
1	Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857	192107.6000
2	Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482	166537.7500
3	Janet	Leverling	Sales Representative	Kirkland	USA	(206) 555-3412	202812.8400
4	Margaret	Peacock	Sales Representative	Redmond	USA	(206) 555-8122	232890.8400
5	Steven	Buchanan	Sales Manager	London	UK	(71) 555-4848	68764.2800
6	Michael	Suyama	Sales Representative	London	UK	(71) 555-7773	73913.1200
7	Robert	King	Sales Representative	London	UK	(71) 555-5598	124568.2400
8	Laura	Callahan	Inside Sales Coordinator	Seattle	USA	(206) 555-1189	126862.2800
9	Anne	Dodsworth	Sales Representative	London	UK	(71) 555-4444	77308.0600

Fig.2. Details are required for the selected item in the list

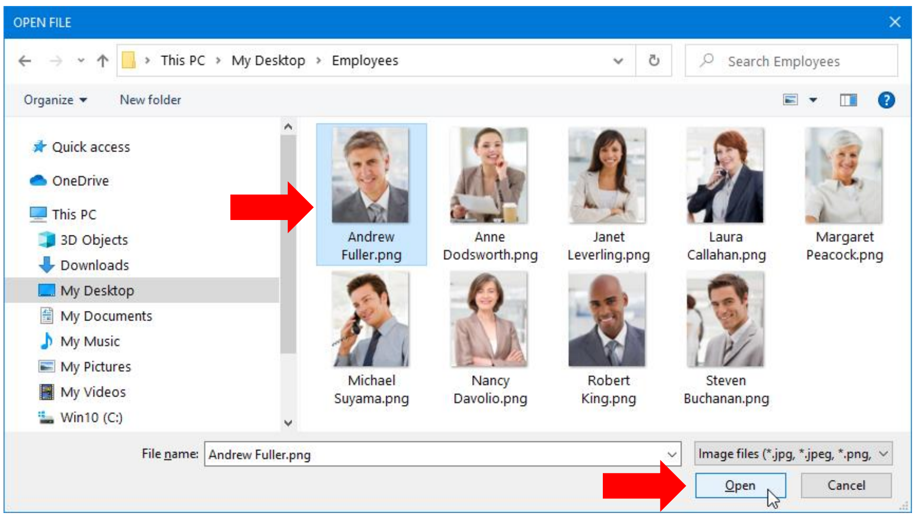

Details Window

Employee ID:	2
Last Name:	Fuller
First Name:	Andrew
Title:	Vice President, Sales
City:	Tacoma
Country:	USA
Phone Number:	(206) 555-9482
Sales Value:	166537.7500

Notes:

Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in

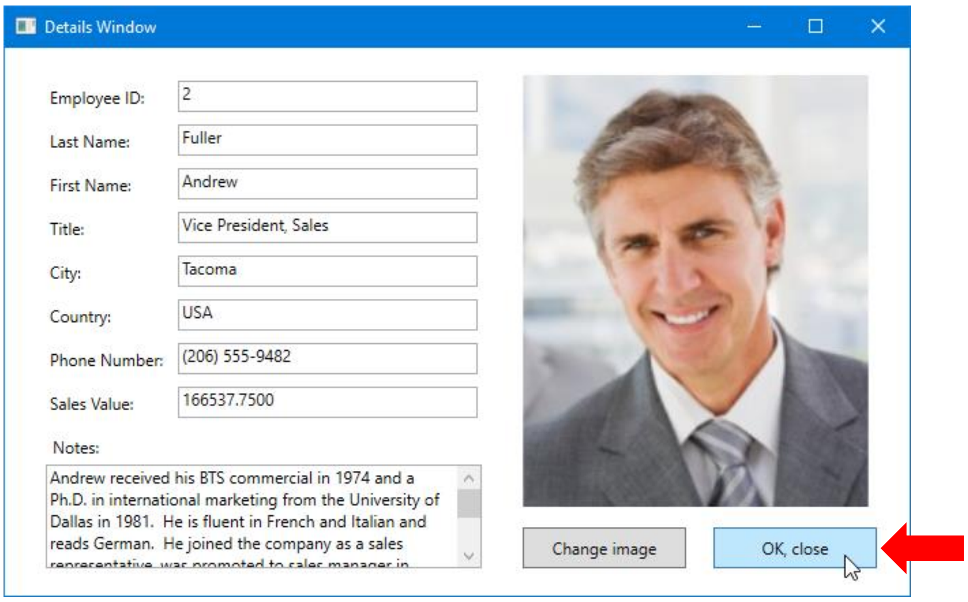
Fig.3. The details window contains the button to change the image



File name: Andrew Fuller.png

Image files (*.jpg, *.jpeg, *.png,

Fig.4. The user points to the new image file in a standard Open File dialog

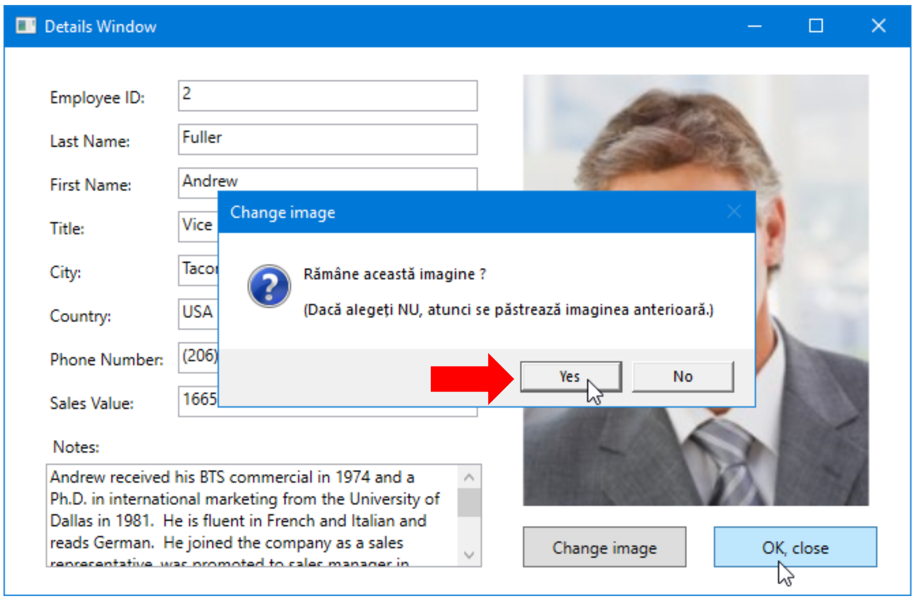

Details Window

Employee ID:	2
Last Name:	Fuller
First Name:	Andrew
Title:	Vice President, Sales
City:	Tacoma
Country:	USA
Phone Number:	(206) 555-9482
Sales Value:	166537.7500

Notes:

Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in

Fig.5. The changes made are saved if the window is closed with the OK button



Employee ID: 2
Last Name: Fuller
First Name: Andrew
Title: Vice
City: Taco
Country: USA
Phone Number: (206)
Sales Value: 1665

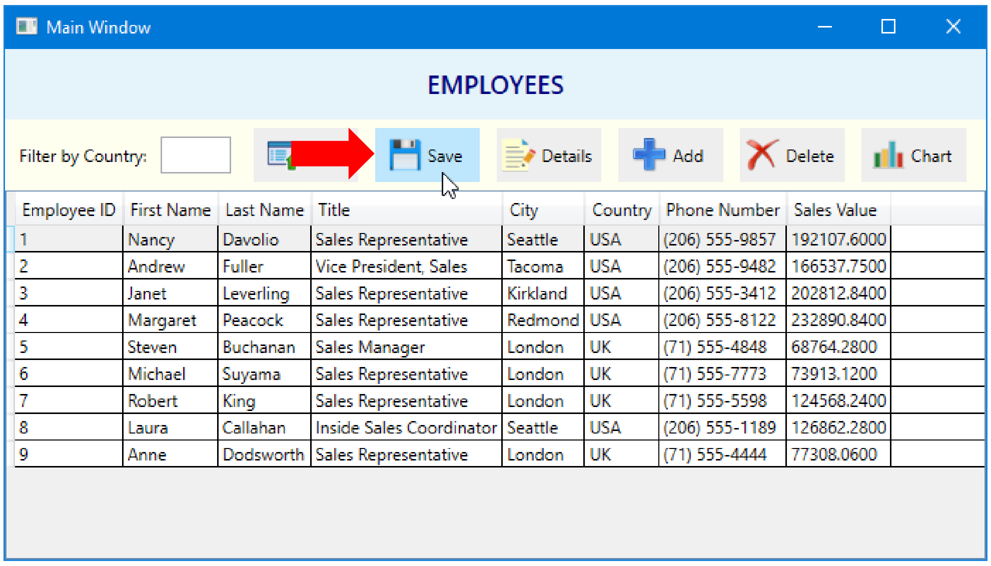
Notes:
Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in

Change image

Rămâne această imagine ?

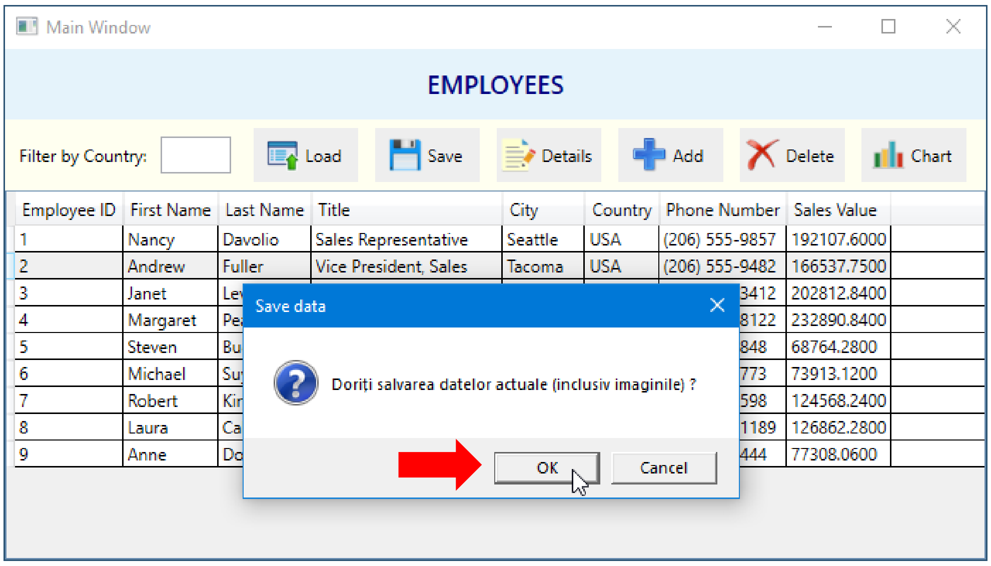
(Dacă alegeți NU, atunci se păstrează imaginea anterioară.)

Fig.6. If the image changes, confirmation is required when closing the window



Employee ID	First Name	Last Name	Title	City	Country	Phone Number	Sales Value
1	Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857	192107.6000
2	Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482	166537.7500
3	Janet	Leverling	Sales Representative	Kirkland	USA	(206) 555-3412	202812.8400
4	Margaret	Peacock	Sales Representative	Redmond	USA	(206) 555-8122	232890.8400
5	Steven	Buchanan	Sales Manager	London	UK	(71) 555-4848	68764.2800
6	Michael	Suyama	Sales Representative	London	UK	(71) 555-7773	73913.1200
7	Robert	King	Sales Representative	London	UK	(71) 555-5598	124568.2400
8	Laura	Callahan	Inside Sales Coordinator	Seattle	USA	(206) 555-1189	126862.2800
9	Anne	Dodsworth	Sales Representative	London	UK	(71) 555-4444	77308.0600

Fig.7. Pressing the Save Data button saves the data to the database



Employee ID	First Name	Last Name	Title	City	Country	Phone Number	Sales Value
1	Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857	192107.6000
2	Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482	166537.7500
3	Janet	Le				3412	202812.8400
4	Margaret	Pe				8122	232890.8400
5	Steven	Bu				848	68764.2800
6	Michael	Su				773	73913.1200
7	Robert	Ki				598	124568.2400
8	Laura	Ca				1189	126862.2800
9	Anne	Do				444	77308.0600

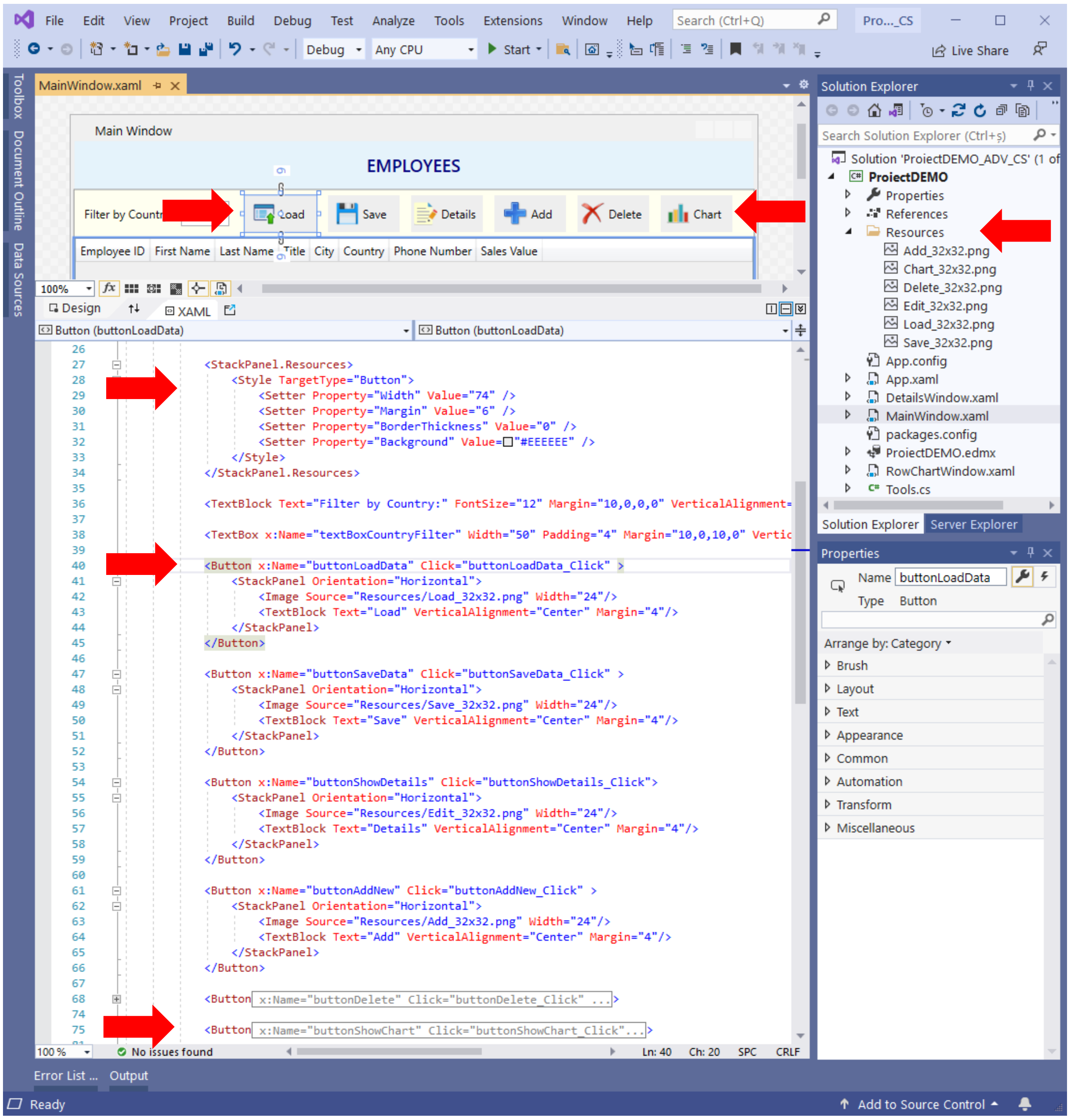
Doriți salvarea datelor actuale (inclusiv imaginile) ?

Fig.8. When saving data, confirmation is required

STEP 1. Changes to the XAML code to add the Save Data button

Change the design of the main window *MainWindow* so:

- the folder is added to the project *Resources* containing images for buttons;
- decrease the width of existing buttons to make room for new buttons (line code 27)
- new buttons are added, but their display mode is also changed so that an adjacent image and text appear on the button (see details in XAML code, starting with line 40)



```
26
27      <StackPanel.Resources>
28          <Style TargetType="Button">
29              <Setter Property="Width" Value="74" />
30              <Setter Property="Margin" Value="6" />
31              <Setter Property="BorderThickness" Value="0" />
32              <Setter Property="Background" Value=□"#EEEEEE" />
33          </Style>
34      </StackPanel.Resources>
35
36      <TextBlock Text="Filter by Country:" FontSize="12" Margin="10,0,0,0" VerticalAlignment=
37
38      <TextBox x:Name="textBoxCountryFilter" Width="50" Padding="4" Margin="10,0,10,0" Vertic
39
40      <Button x:Name="buttonLoadData" Click="buttonLoadData_Click" >
41          <StackPanel Orientation="Horizontal">
42              <Image Source="Resources/Load_32x32.png" Width="24"/>
43              <TextBlock Text="Load" VerticalAlignment="Center" Margin="4"/>
44          </StackPanel>
45      </Button>
46
47      <Button x:Name="buttonSaveData" Click="buttonSaveData_Click" >
48          <StackPanel Orientation="Horizontal">
49              <Image Source="Resources/Save_32x32.png" Width="24"/>
50              <TextBlock Text="Save" VerticalAlignment="Center" Margin="4"/>
51          </StackPanel>
52      </Button>
53
54      <Button x:Name="buttonShowDetails" Click="buttonShowDetails_Click">
55          <StackPanel Orientation="Horizontal">
56              <Image Source="Resources/Edit_32x32.png" Width="24"/>
57              <TextBlock Text="Details" VerticalAlignment="Center" Margin="4"/>
58          </StackPanel>
59      </Button>
60
61      <Button x:Name="buttonAddNew" Click="buttonAddNew_Click" >
62          <StackPanel Orientation="Horizontal">
63              <Image Source="Resources/Add_32x32.png" Width="24"/>
64              <TextBlock Text="Add" VerticalAlignment="Center" Margin="4"/>
65          </StackPanel>
66      </Button>
67
68      <Button x:Name="buttonDelete" Click="buttonDelete_Click" ...>
74
75      <Button x:Name="buttonShowChart" Click="buttonShowChart_Click"...>
```

Fig.9. Adding new buttons in the main window

STEP 2. Fill in the C # code to save the data in the database

In the code window, in the EVENTS region, the method that treats is observed **the event Click the Save Data button** (see **FIG. 10**From lower). The private method is called here**Saved**.

Method **Saved** is located in the PRIVATE METHODS region (see lines 28 and 312 in **Figure 10** From lower). Method code**Saved** is detailed in **Figure 11** in the following.

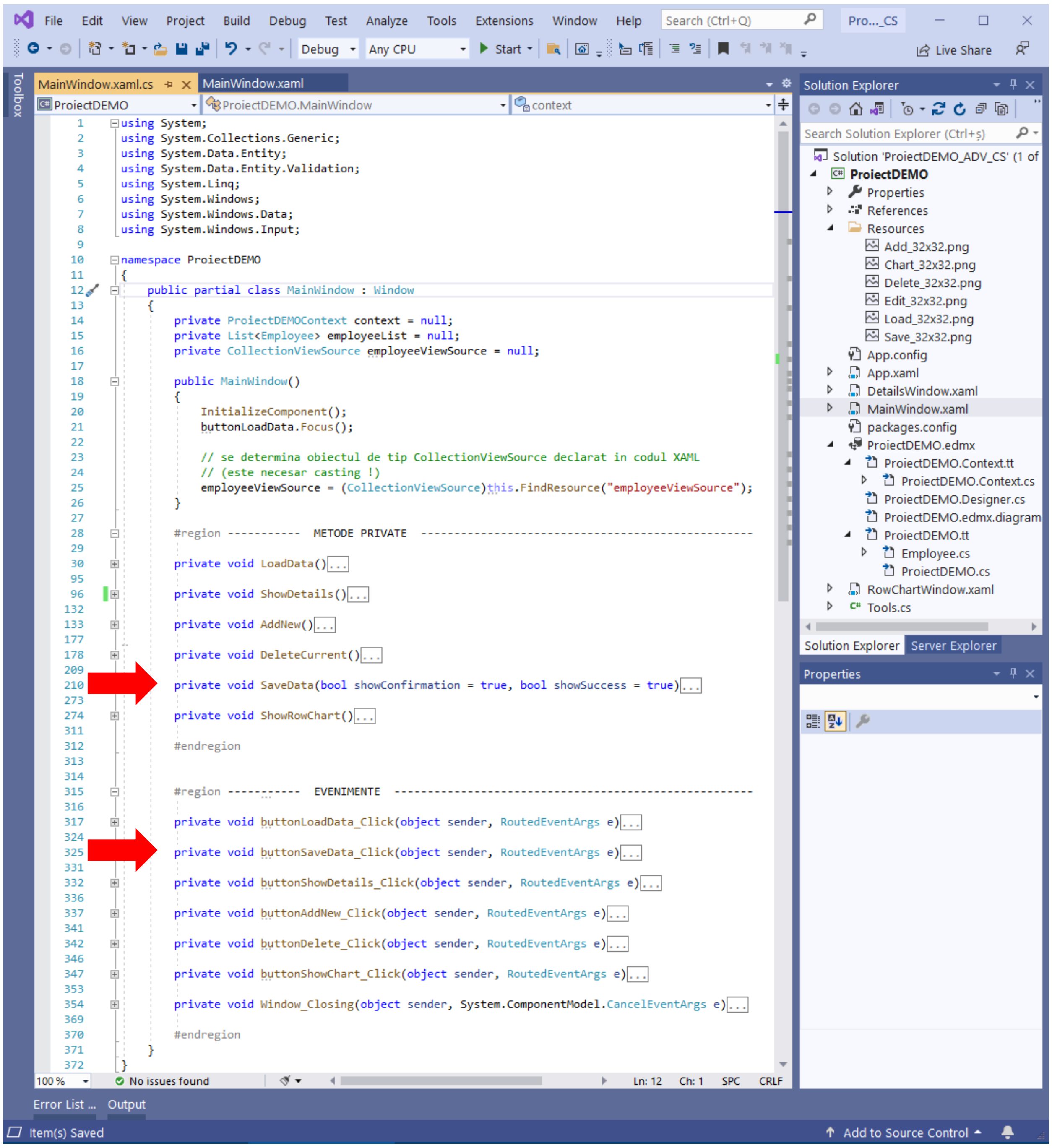
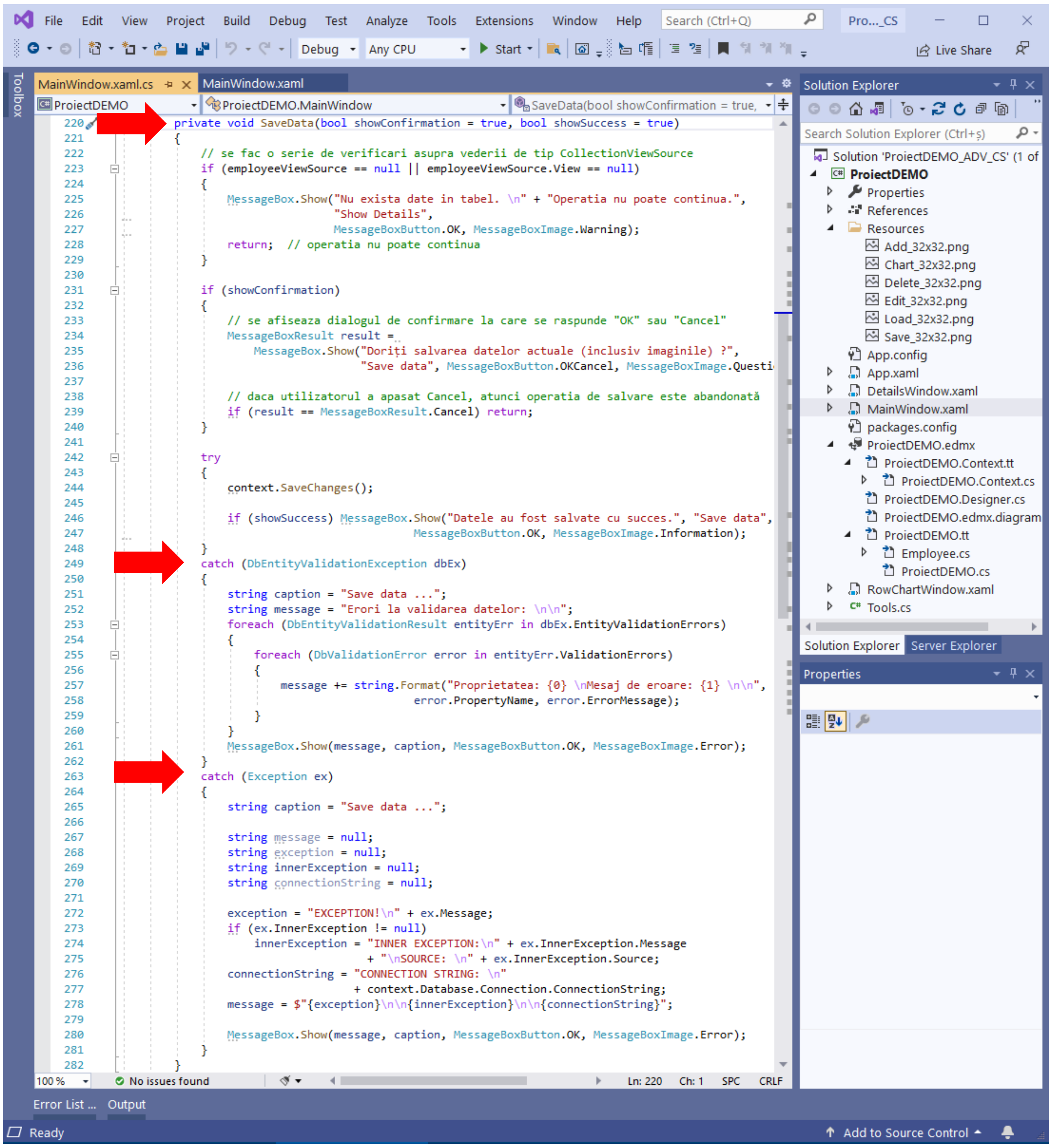
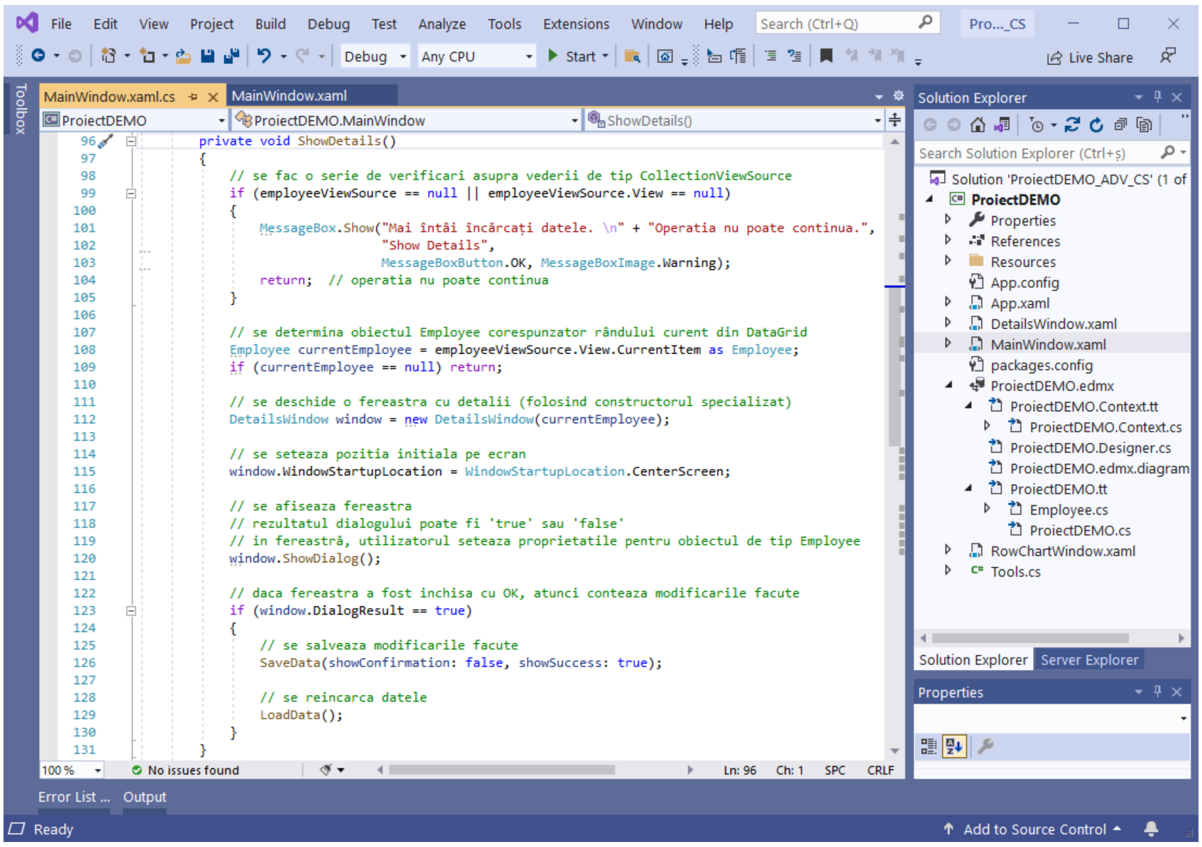

```
  1  using System;
  2  using System.Collections.Generic;
  3  using System.Data.Entity;
  4  using System.Data.Entity.Validation;
  5  using System.Linq;
  6  using System.Windows;
  7  using System.Windows.Data;
  8  using System.Windows.Input;
  9
 10  namespace ProiectDEMO
 11  {
 12      public partial class MainWindow : Window
 13      {
 14          private ProiectDEMOContext context = null;
 15          private List<Employee> employeeList = null;
 16          private CollectionViewSource employeeViewSource = null;
 17
 18          public MainWindow()
 19          {
 20              InitializeComponent();
 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()[...]
 95
 96          private void ShowDetails()[...]
132
133          private void AddNew()[...]
177
178          private void DeleteCurrent()[...]
209
210          private void SaveData(bool showConfirmation = true, bool showSuccess = true)[...]
273
274          private void ShowRowChart()[...]
311
312          #endregion
313
314
315          #region ----------  EVENIMENTE  -------------------------------------------------
316
317          private void buttonLoadData_Click(object sender, RoutedEventArgs e)[...]
324
325          private void buttonSaveData_Click(object sender, RoutedEventArgs e)[...]
331
332          private void buttonShowDetails_Click(object sender, RoutedEventArgs e)[...]
336
337          private void buttonAddNew_Click(object sender, RoutedEventArgs e)[...]
341
342          private void buttonDelete_Click(object sender, RoutedEventArgs e)[...]
346
347          private void buttonShowChart_Click(object sender, RoutedEventArgs e)[...]
353
354          private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)[...]
369
370          #endregion
371      }
372  }
```

Fig.10. MainWindow class code structure (constructor, private methods, events)



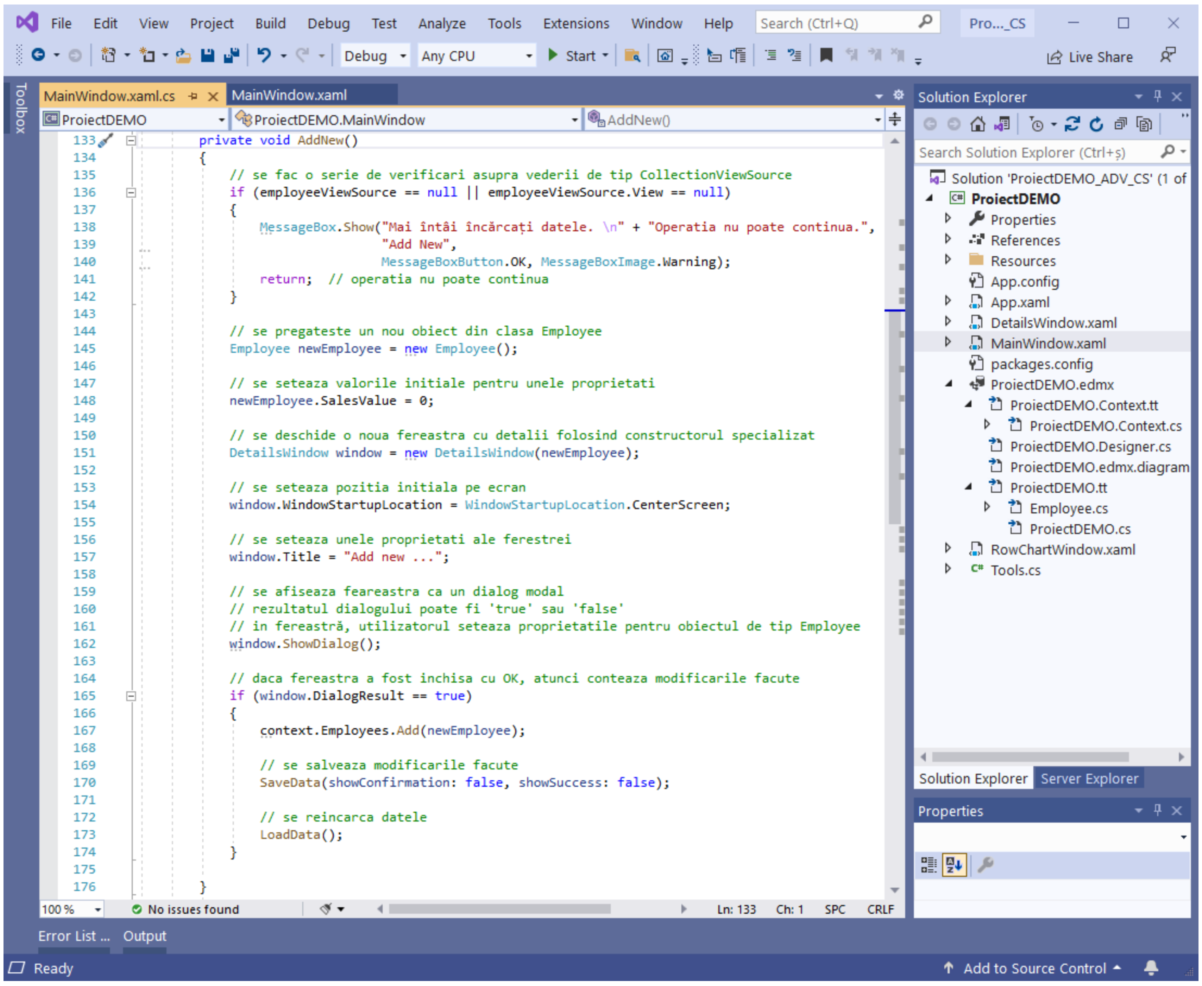
```csharp
private void SaveData(bool showConfirmation = true, bool showSuccess = true)
{
    // se fac o serie de verificari asupra vederii de tip CollectionViewSource
    if (employeeViewSource == null || employeeViewSource.View == null)
    {
        MessageBox.Show("Nu exista date in tabel. \n" + "Operatia nu poate continua.",
                        "Show Details",
                        MessageBoxButton.OK, MessageBoxImage.Warning);
        return;  // operatia nu poate continua
    }

    if (showConfirmation)
    {
        // se afiseaza dialogul de confirmare la care se raspunde "OK" sau "Cancel"
        MessageBoxResult result =
            MessageBox.Show("Doriți salvarea datelor actuale (inclusiv imaginile) ?",
                            "Save data", MessageBoxButton.OKCancel, MessageBoxImage.Questi

        // daca utilizatorul a apasat Cancel, atunci operatia de salvare este abandonată
        if (result == MessageBoxResult.Cancel) return;
    }

    try
    {
        context.SaveChanges();

        if (showSuccess) MessageBox.Show("Datele au fost salvate cu succes.", "Save data",
                                          MessageBoxButton.OK, MessageBoxImage.Information);
    }
    catch (DbEntityValidationException dbEx)
    {
        string caption = "Save data ...";
        string message = "Erori la validarea datelor: \n\n";
        foreach (DbEntityValidationResult entityErr in dbEx.EntityValidationErrors)
        {
            foreach (DbValidationError error in entityErr.ValidationErrors)
            {
                message += string.Format("Proprietatea: {0} \nMesaj de eroare: {1} \n\n",
                                          error.PropertyName, error.ErrorMessage);
            }
        }
        MessageBox.Show(message, caption, MessageBoxButton.OK, MessageBoxImage.Error);
    }
    catch (Exception ex)
    {
        string caption = "Save 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.11. Private method code *Saved*, in detail



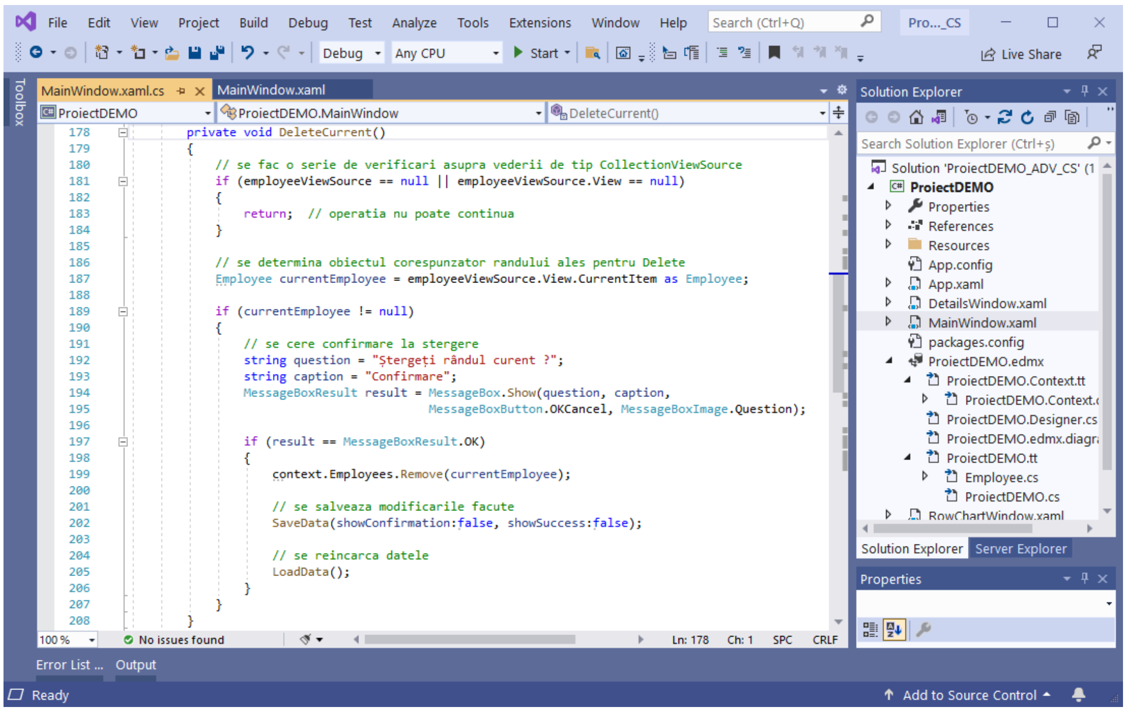
```
private void ShowDetails()
{
    // se fac o serie de verificari asupra vederii de tip CollectionViewSource
    if (employeeViewSource == null || employeeViewSource.View == null)
    {
        MessageBox.Show("Mai întâi încărcați datele. \n" + "Operatia nu poate continua.",
                        "Show Details",
                        MessageBoxButton.OK, MessageBoxImage.Warning);
        return;  // operatia nu poate continua
    }

    // se determina obiectul Employee corespunzator rândului curent din DataGrid
    Employee currentEmployee = employeeViewSource.View.CurrentItem as Employee;
    if (currentEmployee == null) return;

    // se deschide o fereastra cu detalii (folosind constructorul specializat)
    DetailsWindow window = new DetailsWindow(currentEmployee);

    // se seteaza pozitia initiala pe ecran
    window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

    // se afiseaza fereastra
    // rezultatul dialogului poate fi 'true' sau 'false'
    // in fereastră, utilizatorul seteaza proprietatile pentru obiectul de tip Employee
    window.ShowDialog();

    // daca fereastra a fost inchisa cu OK, atunci conteaza modificarile facute
    if (window.DialogResult == true)
    {
        // se salveaza modificarile facute
        SaveData(showConfirmation: false, showSuccess: true);

        // se reincarca datele
        LoadData();
    }
}
```

Fig.12. Private method code*ShowDetails*, in detail



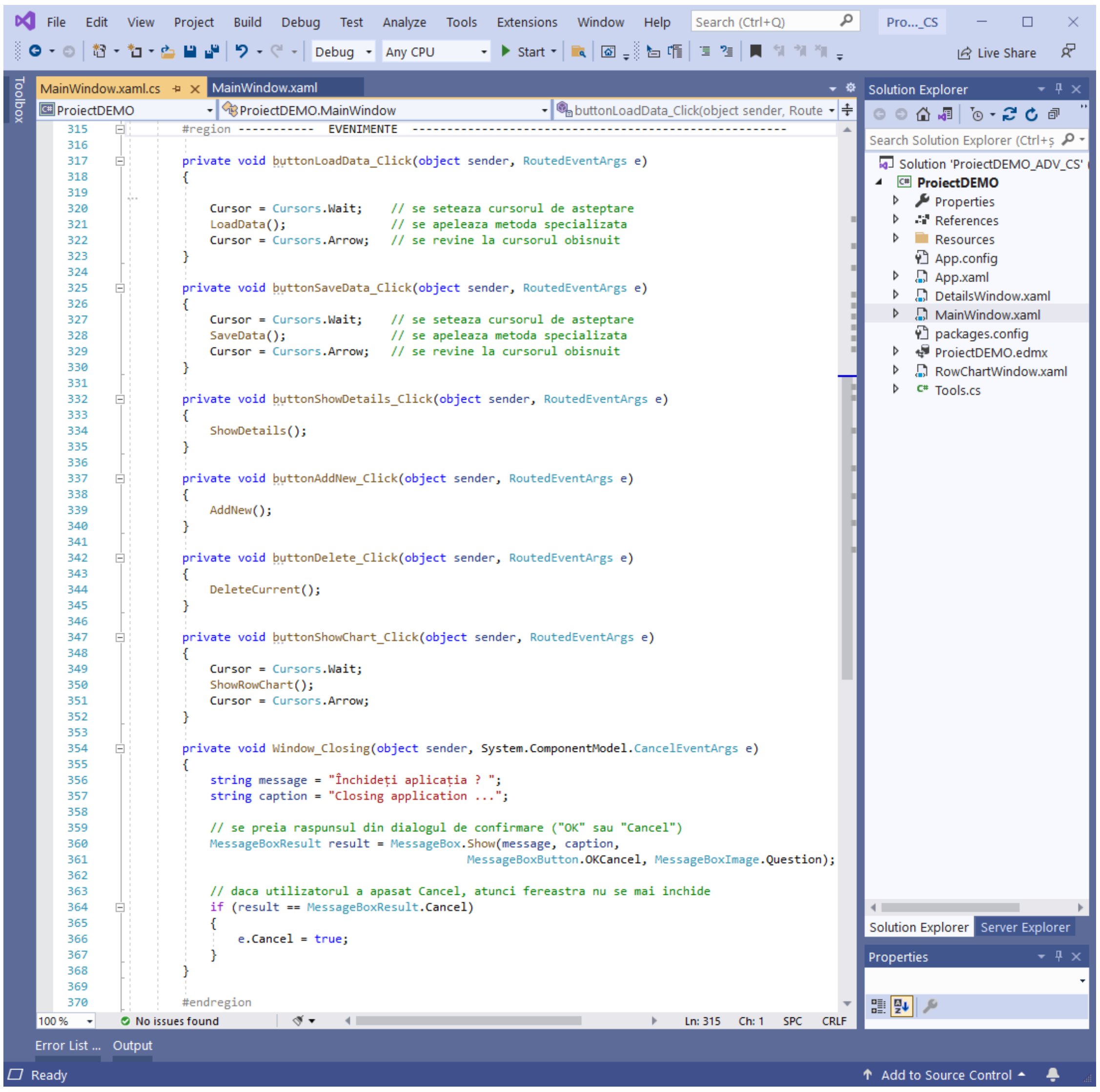
```
private void AddNew()
{
    // se fac o serie de verificari asupra vederii de tip CollectionViewSource
    if (employeeViewSource == null || employeeViewSource.View == null)
    {
        MessageBox.Show("Mai întâi încărcați datele. \n" + "Operatia nu poate continua.",
                        "Add New",
                        MessageBoxButton.OK, MessageBoxImage.Warning);
        return;  // operatia nu poate continua
    }

    // se pregateste un nou obiect din clasa Employee
    Employee newEmployee = new Employee();

    // se seteaza valorile initiale pentru unele proprietati
    newEmployee.SalesValue = 0;

    // se deschide o noua fereastra cu detalii folosind constructorul specializat
    DetailsWindow window = new DetailsWindow(newEmployee);

    // se seteaza pozitia initiala pe ecran
    window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

    // se seteaza unele proprietati ale ferestrei
    window.Title = "Add new ...";

    // se afiseaza feareastra ca un dialog modal
    // rezultatul dialogului poate fi 'true' sau 'false'
    // in fereastră, utilizatorul seteaza proprietatile pentru obiectul de tip Employee
    window.ShowDialog();

    // daca fereastra a fost inchisa cu OK, atunci conteaza modificarile facute
    if (window.DialogResult == true)
    {
        context.Employees.Add(newEmployee);

        // se salveaza modificarile facute
        SaveData(showConfirmation: false, showSuccess: false);

        // se reincarca datele
        LoadData();
    }
}
```

Fig.13. Private method code*AddNew*, in detail



```
private void DeleteCurrent()
{
    // se fac o serie de verificari asupra vederii de tip CollectionViewSource
    if (employeeViewSource == null || employeeViewSource.View == null)
    {
        return;  // operatia nu poate continua
    }

    // se determina obiectul corespunzator randului ales pentru Delete
    Employee currentEmployee = employeeViewSource.View.CurrentItem as Employee;

    if (currentEmployee != null)
    {
        // se cere confirmare la stergere
        string question = "Ștergeți rândul curent ?";
        string caption = "Confirmare";
        MessageBoxResult result = MessageBox.Show(question, caption,
                                  MessageBoxButton.OKCancel, MessageBoxImage.Question);

        if (result == MessageBoxResult.OK)
        {
            context.Employees.Remove(currentEmployee);

            // se salveaza modificarile facute
            SaveData(showConfirmation:false, showSuccess:false);

            // se reincarca datele
            LoadData();
        }
    }
}
```

Fig.14. Private method code *DeleteCurrent*, in detail

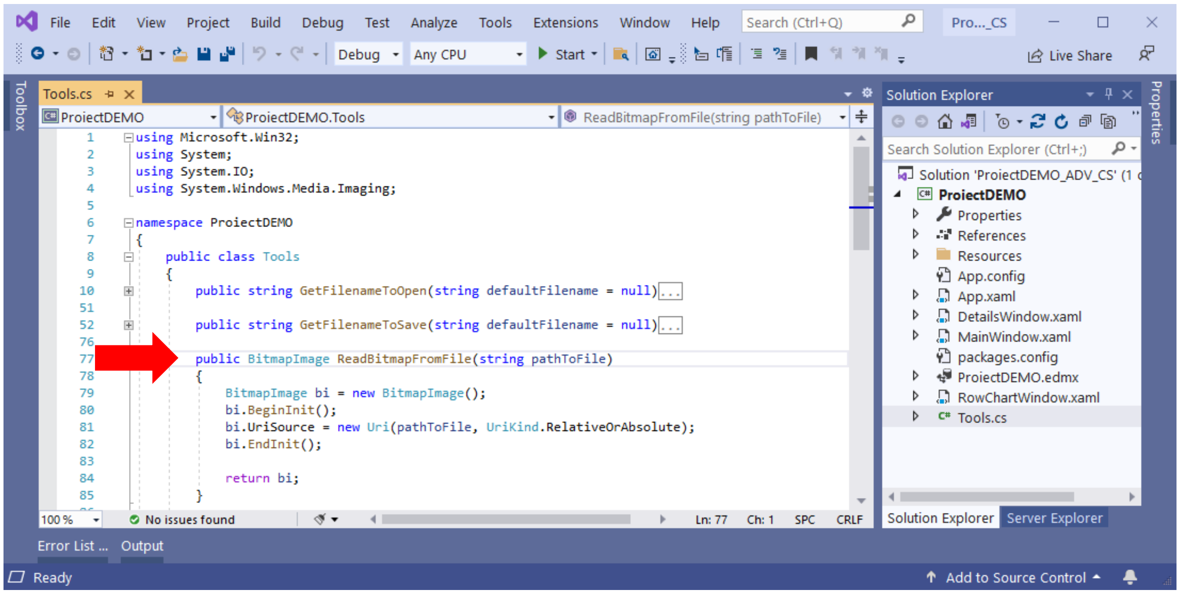


```
#region -----------  EVENIMENTE  ---------------------------------------------------------

private void buttonLoadData_Click(object sender, RoutedEventArgs e)
{

    Cursor = Cursors.Wait;    // se seteaza cursorul de asteptare
    LoadData();               // se apeleaza metoda specializata
    Cursor = Cursors.Arrow;   // se revine la cursorul obisnuit
}

private void buttonSaveData_Click(object sender, RoutedEventArgs e)
{
    Cursor = Cursors.Wait;    // se seteaza cursorul de asteptare
    SaveData();               // se apeleaza metoda specializata
    Cursor = Cursors.Arrow;   // se revine la cursorul obisnuit
}

private void buttonShowDetails_Click(object sender, RoutedEventArgs e)
{
    ShowDetails();
}

private void buttonAddNew_Click(object sender, RoutedEventArgs e)
{
    AddNew();
}

private void buttonDelete_Click(object sender, RoutedEventArgs e)
{
    DeleteCurrent();
}

private void buttonShowChart_Click(object sender, RoutedEventArgs e)
{
    Cursor = Cursors.Wait;
    ShowRowChart();
    Cursor = Cursors.Arrow;
}

private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    string message = "Închideți aplicația ? ";
    string caption = "Closing application ...";

    // se preia raspunsul din dialogul de confirmare ("OK" sau "Cancel")
    MessageBoxResult result = MessageBox.Show(message, caption,
                                  MessageBoxButton.OKCancel, MessageBoxImage.Question);

    // daca utilizatorul a apasat Cancel, atunci fereastra nu se mai inchide
    if (result == MessageBoxResult.Cancel)
    {
        e.Cancel = true;
    }
}

#endregion
```

Fig.15. Method code *event-handler* from *MainWindow*

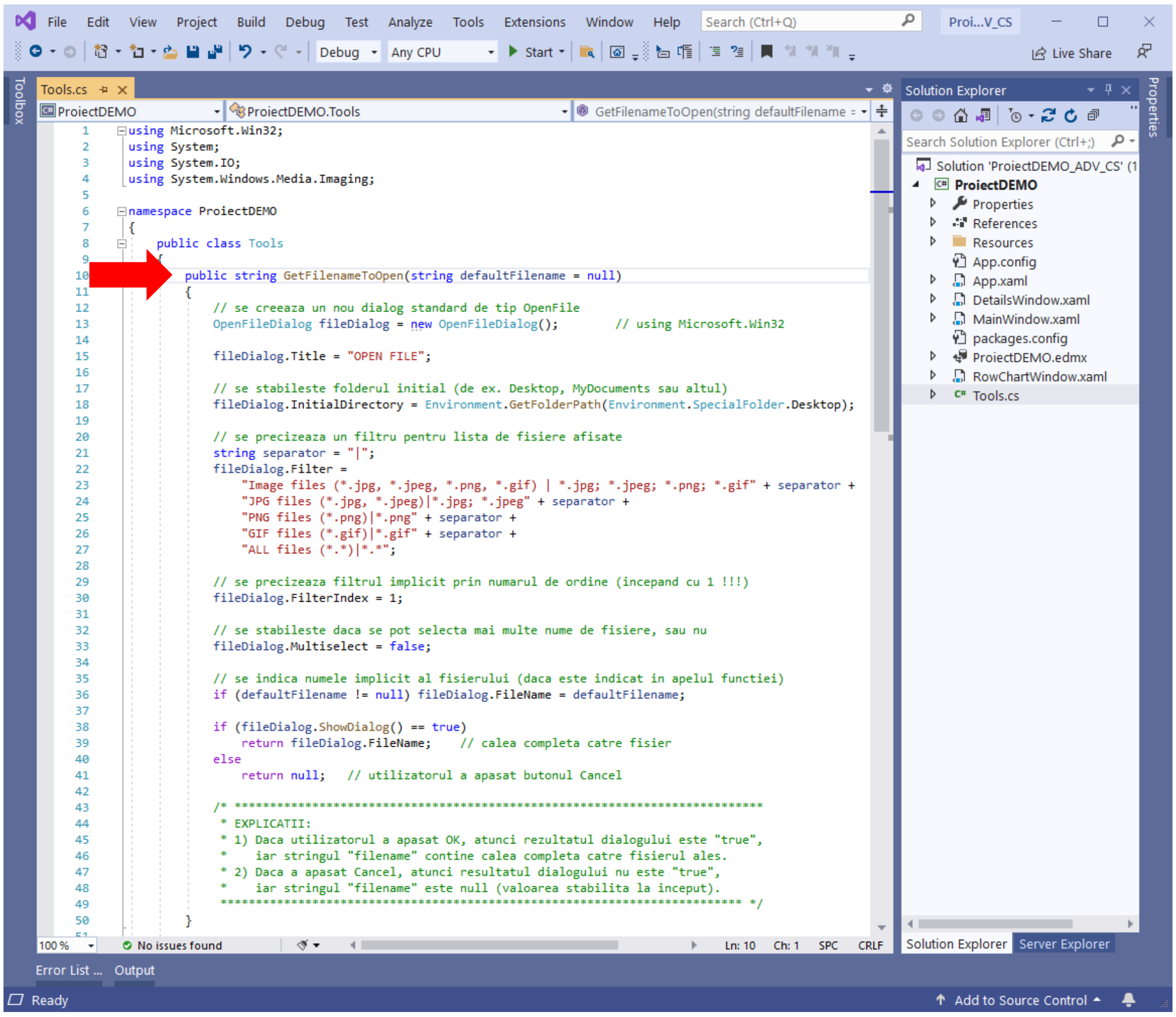
STEP 3. Create a class with helpful methods and functions

A new class named is added to the Visual Studio project **Tools**. It contains auxiliary functions that will be needed later:*GetFilenameToOpen, GetFilenameToSave, ReadBitmapFromFile*.



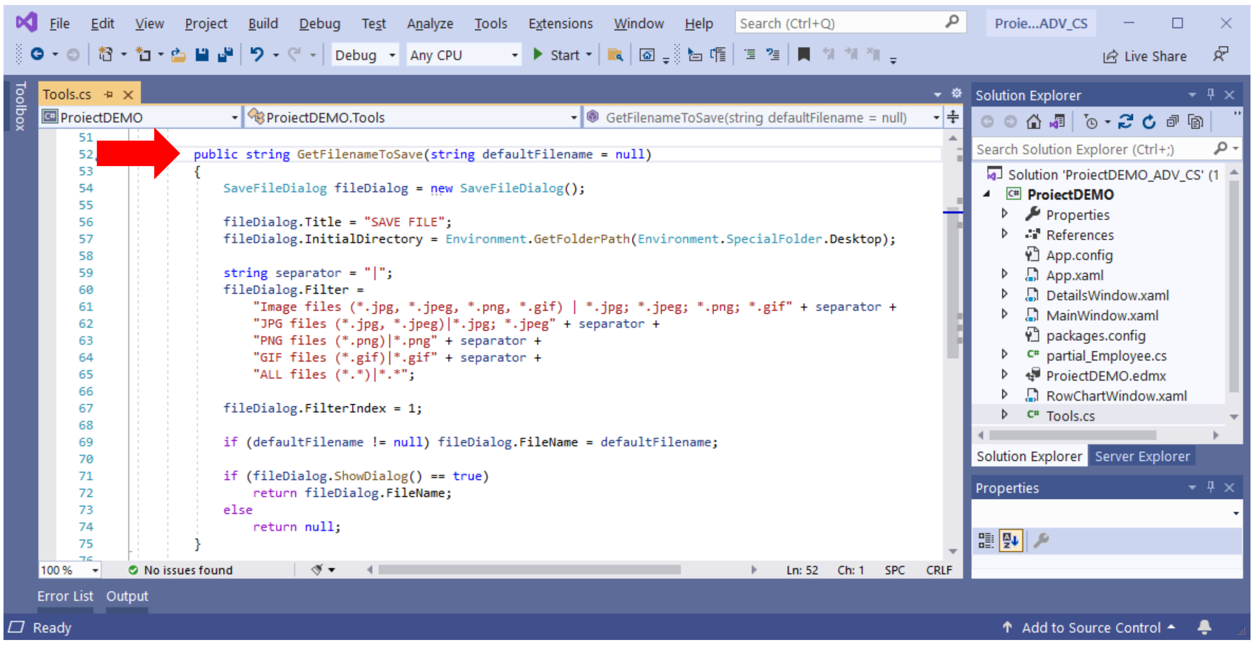
```
using Microsoft.Win32;
using System;
using System.IO;
using System.Windows.Media.Imaging;

namespace ProiectDEMO
{
    public class Tools
    {
        public string GetFilenameToOpen(string defaultFilename = null)...

        public string GetFilenameToSave(string defaultFilename = null)...

        public BitmapImage ReadBitmapFromFile(string pathToFile)
        {
            BitmapImage bi = new BitmapImage();
            bi.BeginInit();
            bi.UriSource = new Uri(pathToFile, UriKind.RelativeOrAbsolute);
            bi.EndInit();

            return bi;
        }
```

Fig.16. Class code structure *Tools*and the method *ReadBitmapFromFile*



```
using Microsoft.Win32;
using System;
using System.IO;
using System.Windows.Media.Imaging;

namespace ProiectDEMO
{
    public class Tools
    {
        public string GetFilenameToOpen(string defaultFilename = null)
        {
            // se creeaza un nou dialog standard de tip OpenFile
            OpenFileDialog fileDialog = new OpenFileDialog();        // using Microsoft.Win32

            fileDialog.Title = "OPEN FILE";

            // se stabileste folderul initial (de ex. Desktop, MyDocuments sau altul)
            fileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

            // se precizeaza un filtru pentru lista de fisiere afisate
            string separator = "|";
            fileDialog.Filter =
                "Image files (*.jpg, *.jpeg, *.png, *.gif) | *.jpg; *.jpeg; *.png; *.gif" + separator +
                "JPG files (*.jpg, *.jpeg)|*.jpg; *.jpeg" + separator +
                "PNG files (*.png)|*.png" + separator +
                "GIF files (*.gif)|*.gif" + separator +
                "ALL files (*.*)|*.*";

            // se precizeaza filtrul implicit prin numarul de ordine (incepand cu 1 !!!)
            fileDialog.FilterIndex = 1;

            // se stabileste daca se pot selecta mai multe nume de fisiere, sau nu
            fileDialog.Multiselect = false;

            // se indica numele implicit al fisierului (daca este indicat in apelul functiei)
            if (defaultFilename != null) fileDialog.FileName = defaultFilename;

            if (fileDialog.ShowDialog() == true)
                return fileDialog.FileName;    // calea completa catre fisier
            else
                return null;   // utilizatorul a apasat butonul Cancel

            /* ***************************************************************************
             * EXPLICATII:
             * 1) Daca utilizatorul a apasat OK, atunci rezultatul dialogului este "true",
             *    iar stringul "filename" contine calea completa catre fisierul ales.
             * 2) Daca a apasat Cancel, atunci resultatul dialogului nu este "true",
             *    iar stringul "filename" este null (valoarea stabilita la inceput).
             ************************************************************************** */
        }
```

Fig.17. Method code *GetFilenameToOpen*(class *Tools*)



```
public string GetFilenameToSave(string defaultFilename = null)
{
    SaveFileDialog fileDialog = new SaveFileDialog();

    fileDialog.Title = "SAVE FILE";
    fileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

    string separator = "|";
    fileDialog.Filter =
        "Image files (*.jpg, *.jpeg, *.png, *.gif) | *.jpg; *.jpeg; *.png; *.gif" + separator +
        "JPG files (*.jpg, *.jpeg)|*.jpg; *.jpeg" + separator +
        "PNG files (*.png)|*.png" + separator +
        "GIF files (*.gif)|*.gif" + separator +
        "ALL files (*.*)|*.*";

    fileDialog.FilterIndex = 1;

    if (defaultFilename != null) fileDialog.FileName = defaultFilename;

    if (fileDialog.ShowDialog() == true)
        return fileDialog.FileName;
    else
        return null;
}
```

Fig.18. Method code *GetFilenameToSave*(class *Tools*)

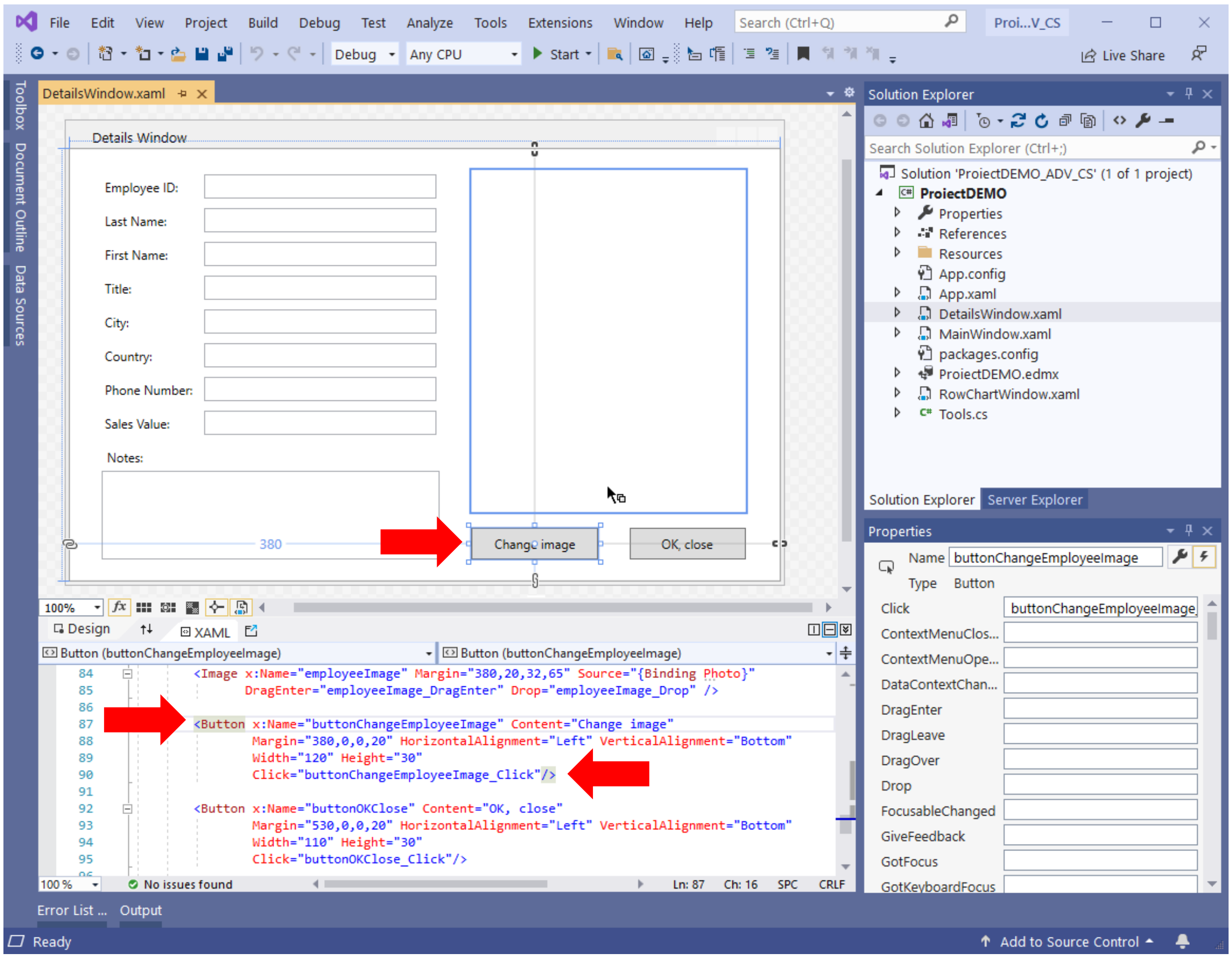
The role of these auxiliary functions is indicated by their name:

- ***GetFilenameToOpen***-returns the full path to the file to be uploaded (using a standard type dialog**Open File**);
- ***GetFilenameToSave***-returns the full path to the file to be saved (using a standard type dialog**Save File**);
- ***ReadBitmapFromFile***-returns an object from the class**BitmapImage**obtained by reading the contents of a file on disk (with the extension * .jpg, * .png, * .gif, etc.).

STEP 4. Change the appearance of the details display window

In the window**DetailsWindow** the type control is slightly resized*Image* to make room at the bottom for new buttons. Then two new buttons are added:**Change image** and **OK, close**.

The image below shows these details in the XAML code.

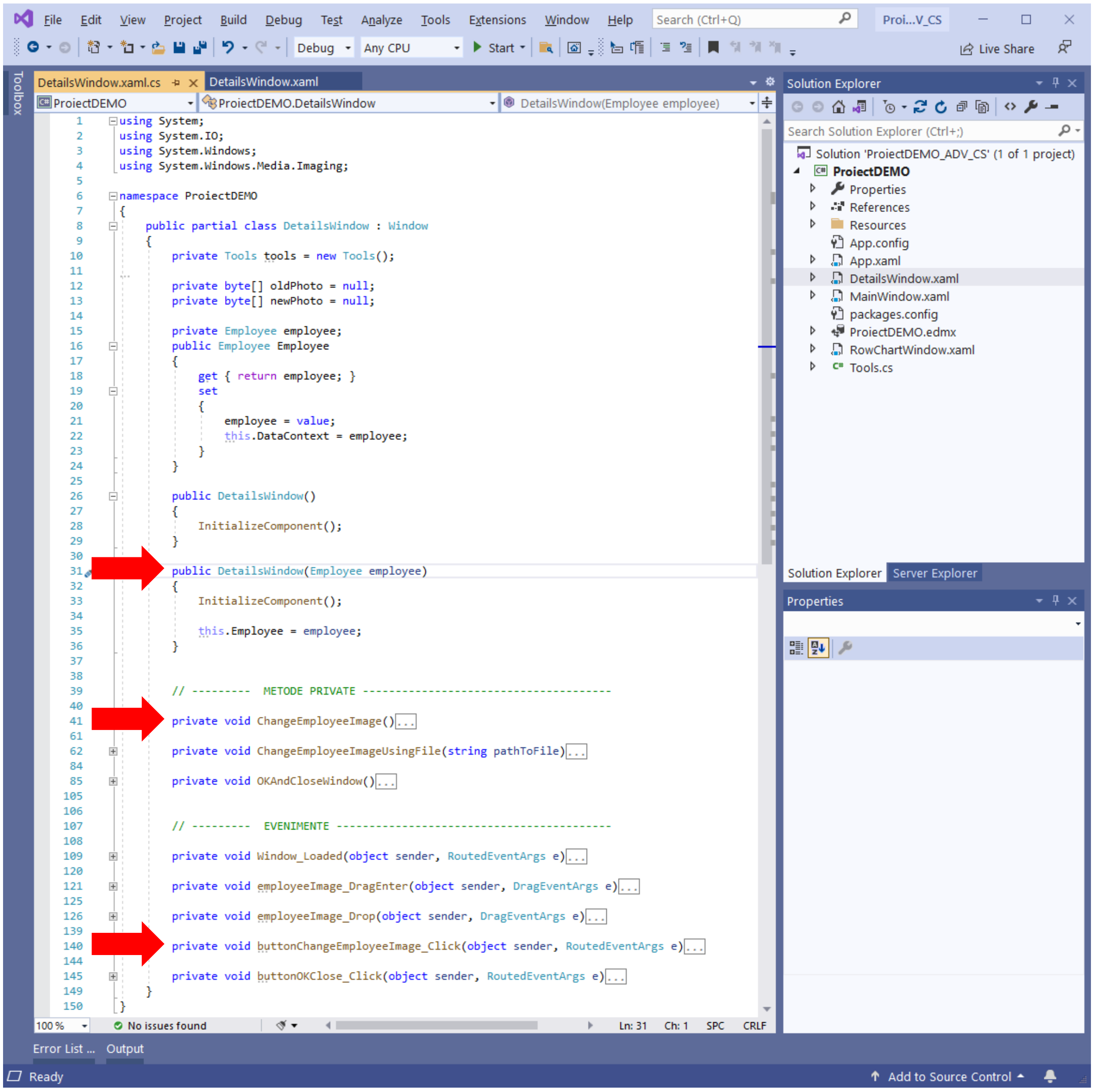


```
<Image x:Name="employeeImage" Margin="380,20,32,65" Source="{Binding Photo}"
       DragEnter="employeeImage_DragEnter" Drop="employeeImage_Drop" />

<Button x:Name="buttonChangeEmployeeImage" Content="Change image"
        Margin="380,0,0,20" HorizontalAlignment="Left" VerticalAlignment="Bottom"
        Width="120" Height="30"
        Click="buttonChangeEmployeeImage_Click"/>

<Button x:Name="buttonOKClose" Content="OK, close"
        Margin="530,0,0,20" HorizontalAlignment="Left" VerticalAlignment="Bottom"
        Width="110" Height="30"
        Click="buttonOKClose_Click"/>
```

Fig.19. Window changes*DetailsWindow* is reflected in the XAML code

Changes (additions) to window functionality **DetailsWindow** are presented below.

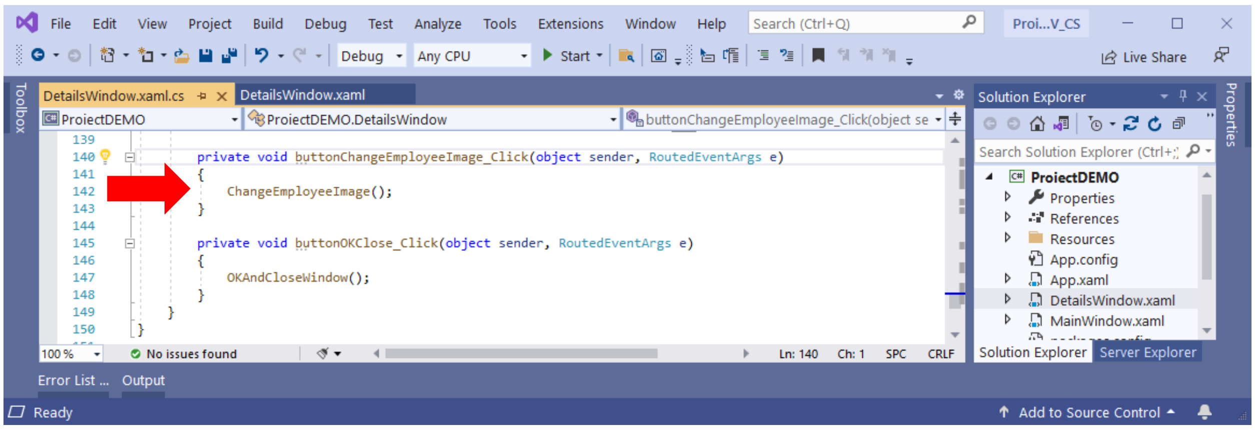
STEP 5. Modify (extend) the functionality of the detail display window



```
using System;
using System.IO;
using System.Windows;
using System.Windows.Media.Imaging;

namespace ProiectDEMO
{
    public partial class DetailsWindow : Window
    {
        private Tools tools = new Tools();

        private byte[] oldPhoto = null;
        private byte[] newPhoto = null;

        private Employee employee;
        public Employee Employee
        {
            get { return employee; }
            set
            {
                employee = value;
                this.DataContext = employee;
            }
        }

        public DetailsWindow()
        {
            InitializeComponent();
        }

        public DetailsWindow(Employee employee)
        {
            InitializeComponent();

            this.Employee = employee;
        }


        // --------  METODE PRIVATE -----------------------------------

        private void ChangeEmployeeImage()...

        private void ChangeEmployeeImageUsingFile(string pathToFile)...

        private void OKAndCloseWindow()...


        // --------  EVENIMENTE ---------------------------------------

        private void Window_Loaded(object sender, RoutedEventArgs e)...

        private void employeeImage_DragEnter(object sender, DragEventArgs e)...

        private void employeeImage_Drop(object sender, DragEventArgs e)...

        private void buttonChangeEmployeeImage_Click(object sender, RoutedEventArgs e)...

        private void buttonOKClose_Click(object sender, RoutedEventArgs e)...
    }
}
```

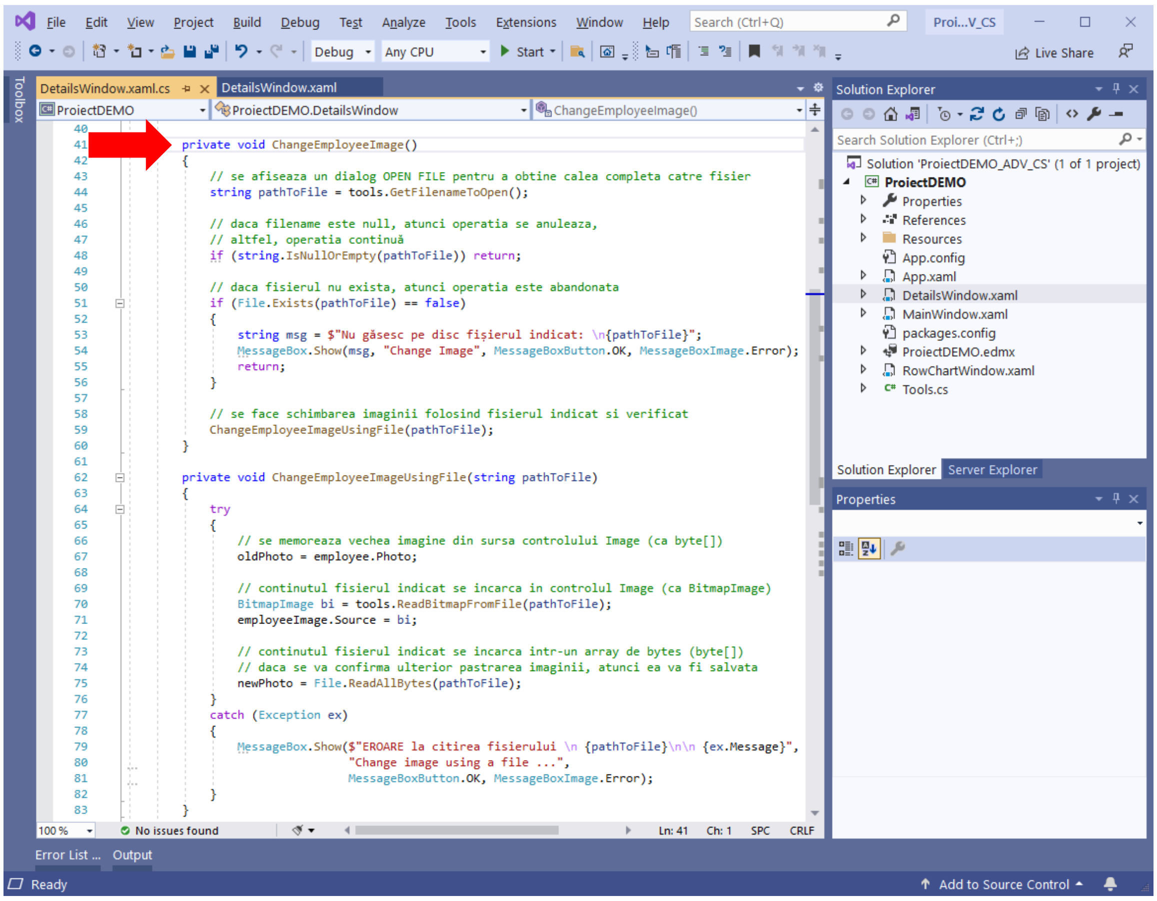
Fig.20. The C # code structure of the DetailsWindow class

The figure above shows the structure of the class code **DetailsWindow**.



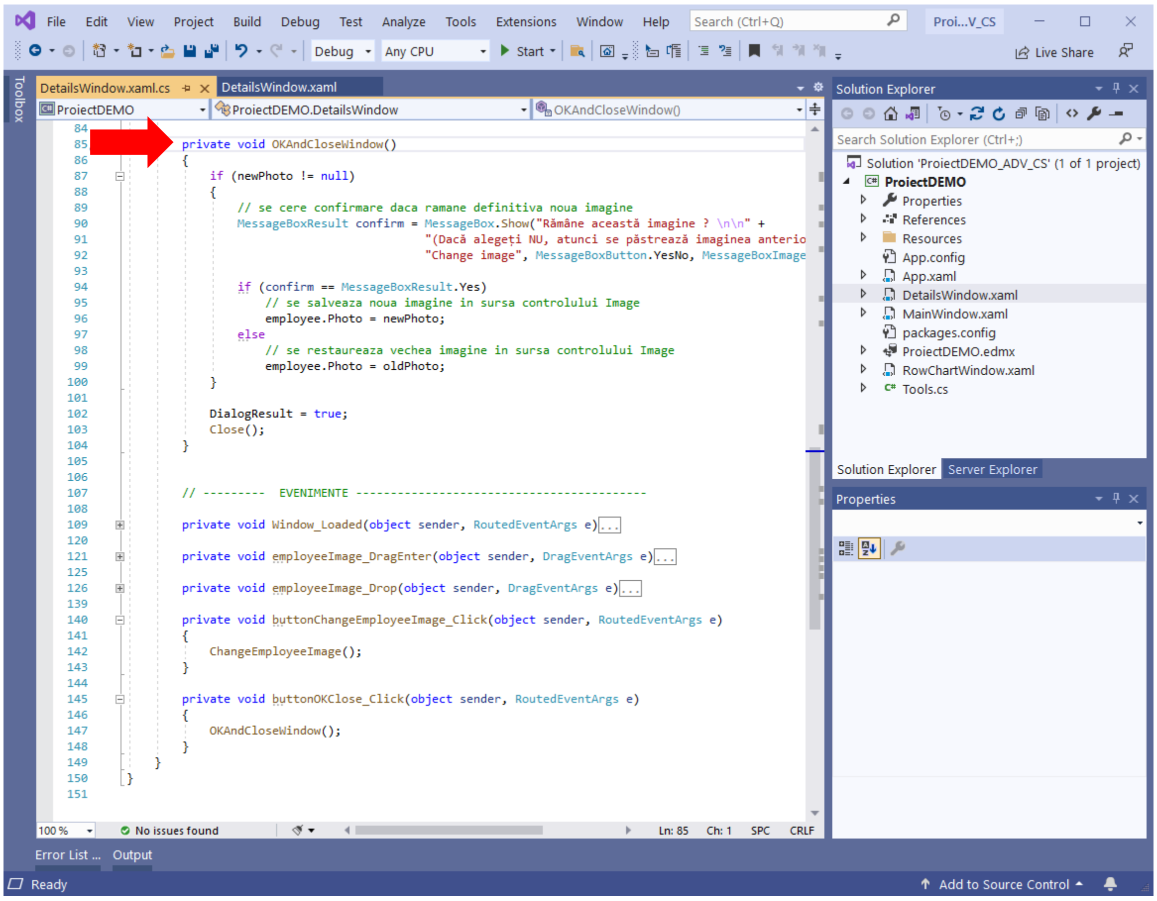
```
        private void buttonChangeEmployeeImage_Click(object sender, RoutedEventArgs e)
        {
            ChangeEmployeeImage();
        }

        private void buttonOKClose_Click(object sender, RoutedEventArgs e)
        {
            OKAndCloseWindow();
        }
    }
}
```

Fig.22. Event-handler methods in the DetailsWindow window



```
private void ChangeEmployeeImage()
{
    // se afiseaza un dialog OPEN FILE pentru a obtine calea completa catre fisier
    string pathToFile = tools.GetFilenameToOpen();

    // daca filename este null, atunci operatia se anuleaza,
    // altfel, operatia continuă
    if (string.IsNullOrEmpty(pathToFile)) return;

    // daca fisierul nu exista, atunci operatia este abandonata
    if (File.Exists(pathToFile) == false)
    {
        string msg = $"Nu găsesc pe disc fișierul indicat: \n{pathToFile}";
        MessageBox.Show(msg, "Change Image", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    // se face schimbarea imaginii folosind fisierul indicat si verificat
    ChangeEmployeeImageUsingFile(pathToFile);
}

private void ChangeEmployeeImageUsingFile(string pathToFile)
{
    try
    {
        // se memoreaza vechea imagine din sursa controlului Image (ca byte[])
        oldPhoto = employee.Photo;

        // continutul fisierul indicat se incarca in controlul Image (ca BitmapImage)
        BitmapImage bi = tools.ReadBitmapFromFile(pathToFile);
        employeeImage.Source = bi;

        // continutul fisierul indicat se incarca intr-un array de bytes (byte[])
        // daca se va confirma ulterior pastrarea imaginii, atunci ea va fi salvata
        newPhoto = File.ReadAllBytes(pathToFile);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"EROARE la citirea fisierului \n {pathToFile}\n\n {ex.Message}",
                        "Change image using a file ...",
                        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

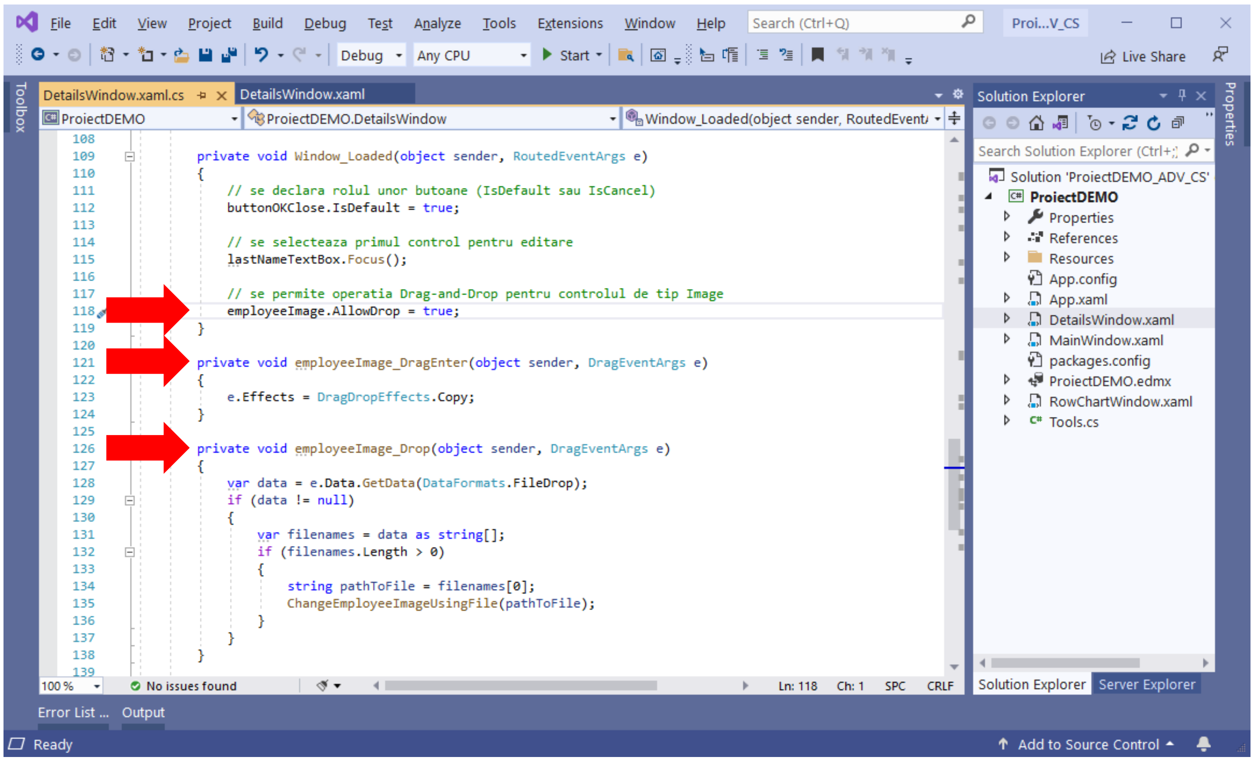
Fig.23. Private method code *ChangeEmployeeImage* (DetailsWindow)



```
        private void OKAndCloseWindow()
        {
            if (newPhoto != null)
            {
                // se cere confirmare daca ramane definitiva noua imagine
                MessageBoxResult confirm = MessageBox.Show("Rămâne această imagine ? \n\n" +
                                           "(Dacă alegeți NU, atunci se păstrează imaginea anterio
                                           "Change image", MessageBoxButton.YesNo, MessageBoxImage

                if (confirm == MessageBoxResult.Yes)
                    // se salveaza noua imagine in sursa controlului Image
                    employee.Photo = newPhoto;
                else
                    // se restaureaza vechea imagine in sursa controlului Image
                    employee.Photo = oldPhoto;
            }

            DialogResult = true;
            Close();
        }


        // ---------  EVENIMENTE ----------------------------------------

        private void Window_Loaded(object sender, RoutedEventArgs e)...

        private void employeeImage_DragEnter(object sender, DragEventArgs e)...

        private void employeeImage_Drop(object sender, DragEventArgs e)...

        private void buttonChangeEmployeeImage_Click(object sender, RoutedEventArgs e)
        {
            ChangeEmployeeImage();
        }

        private void buttonOKClose_Click(object sender, RoutedEventArgs e)
        {
            OKAndCloseWindow();
        }
    }
}
```

Fig.24. Private method code *OKCloseWindow* (DetailsWindow)

STEP 6. Functionality *drag-and-drop* for images in the DetailsWindow window



```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    // se declara rolul unor butoane (IsDefault sau IsCancel)
    buttonOKClose.IsDefault = true;

    // se selecteaza primul control pentru editare
    lastNameTextBox.Focus();

    // se permite operatia Drag-and-Drop pentru controlul de tip Image
    employeeImage.AllowDrop = true;
}

private void employeeImage_DragEnter(object sender, DragEventArgs e)
{
    e.Effects = DragDropEffects.Copy;
}

private void employeeImage_Drop(object sender, DragEventArgs e)
{
    var data = e.Data.GetData(DataFormats.FileDrop);
    if (data != null)
    {
        var filenames = data as string[];
        if (filenames.Length > 0)
        {
            string pathToFile = filenames[0];
            ChangeEmployeeImageUsingFile(pathToFile);
        }
    }
}
```

Fig.25. Methods for implementing functionality "*drag-and-drop*"for images

In the figure above, the instruction in the method *Window_Loaded* indicated by the colored arrow activates the functionality *drag-and-drop* for control *Image* indicated (here *employeeImage*). This setting cannot be made in the panel *Properties*, but only in code. The instruction can be written in constructors (in all, if there are several), but it is more practical to be written only once in the method ***Window_Loaded*** (which always runs automatically after the window builder, either the default one or the specialized one).

The other two colored arrows point to two methods *event-handler* for two type control events *Image* from the window (here the control *employeeImage*).

It is not enough to write the above code. The correct association between the event and the method handling the event must be made (see XAML code in Fig.19). The easiest way is to use the panel *Properties* and with a double-click on the respective event to perform two operations at once: <u>creating the method</u> *event-handler*, But and <u>the correct association of the method with the event</u> .