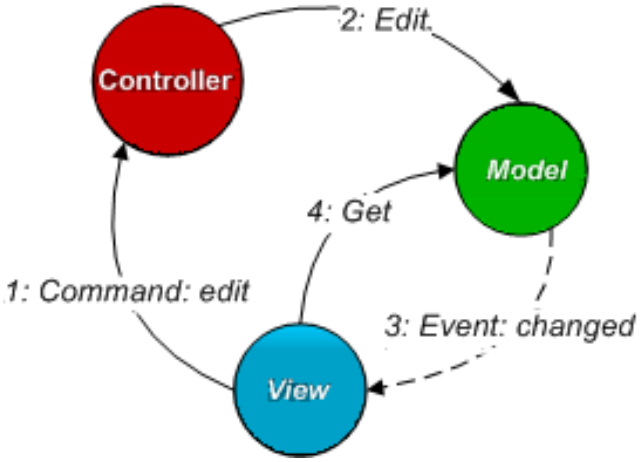
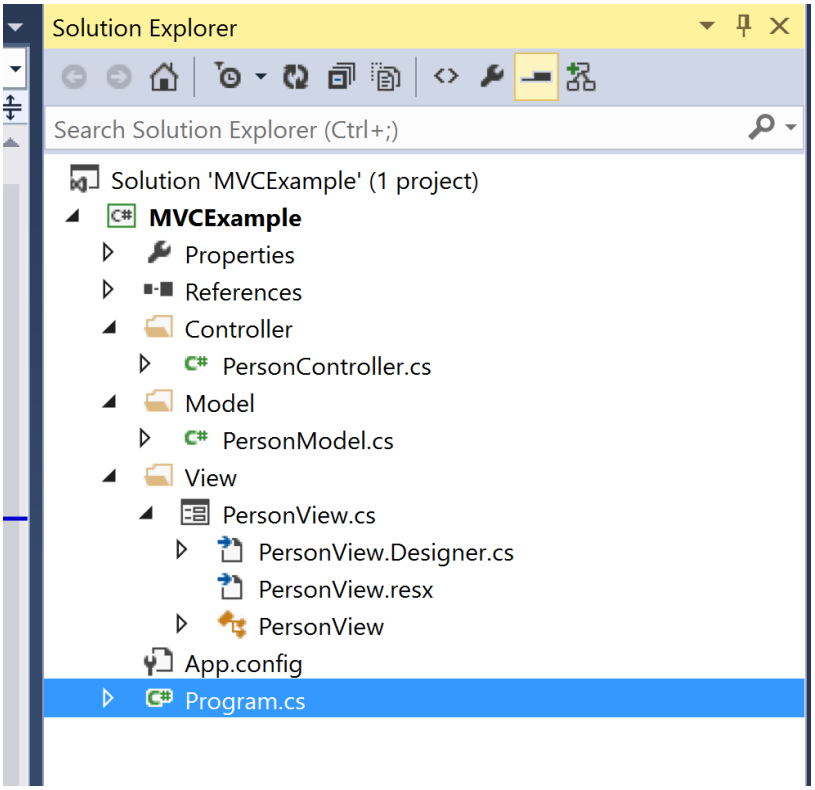
Model-View-Controller Pattern

Model: reprezinta datele/informatiile cu care lucreaza aplicatia;
View: reprezinta interfata grafica
Controller: face legatura intre Model si View, modifica View-ul in functie de actiunile userului;



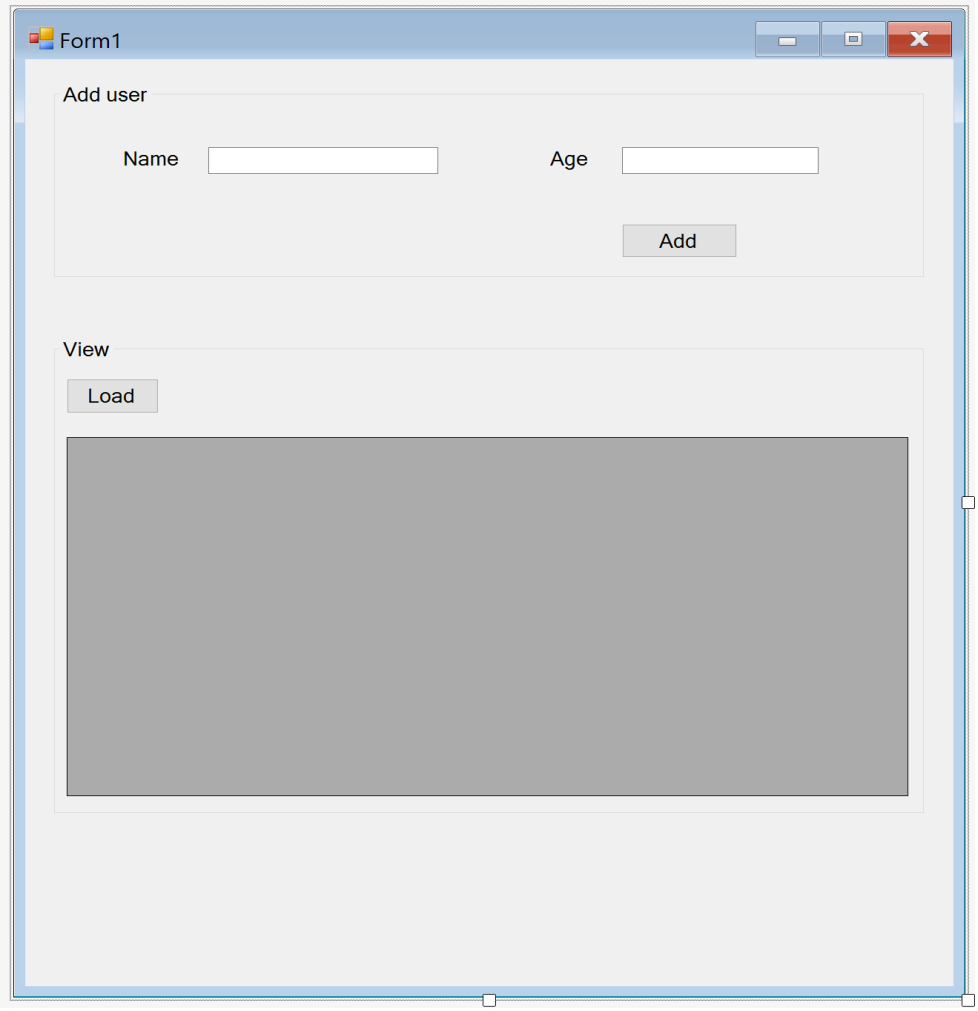
Exemplu MVC pe un proiect simplu

Proiectul are urmatoarea structura:



In interfata grafica userul are posibilitatea sa adauge o persoana in baza da date si sa vada

toate persoanele din baza de date.



1. **Add**

La apasarea butonului “Add” se creeaza un obiect de tipul PersonModel, care e trimis apoi catre baza de date. In acest moment View-ul (fereastra) declanseaza evenimentul AddPerson la care Controller-ul este abonat, si mai departe stie ce sa faca.
Evenimentul contine un parametru de tipul PersonModel, si anume obiectul creat in view pe click-ul butonului “Add”. Controller-ul trimite apoi acel obiect catre baza de date.

Functia din View arata asa:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if(AddPerson != null)
    {
        PersonModel person = new PersonModel();
        person.Name = txtName.Text;
        person.Age = Convert.ToInt32(txtAge.Text);

        AddPerson(this, person);
    }
}
```

Linia incadrada in chenarul rosu reprezinta declansarea evenimentului. Controllerul se aboneaza la acest eveniment in momentul crearii:

```
IPersonView _view;

1 reference
public PersonController(IPersonView view)
{
    _view = view;
    _view.AddPerson += _view_AddPerson;
    _view.GetPeople += _view_GetPeople;
}
1 reference
void _view_AddPerson(object sender, Model.PersonModel e)
{
    new PersonDAL().AddPerson(e);
}
```

Clasa PersonDAL este folosita pentru a comunica direct cu baza de date(DAL = Data Access Layer), deoarece niciuna dintre cele 3 componente (Model,View, Controller) nu are aceasta responsabilitate.

PersonDAL are urmatoarea structura:

```
public class PersonDAL
{
    SqlConnection connection = new SqlConnection("user id=sa;" +
                                "password=[redacted];server=[redacted]" +
                                "Trusted_Connection=yes;" +
                                "database=MVCExample; " +
                                "connection timeout=30");

    string addPersonQuery = "Insert into Person values (@name, @age)";
    string getPeopleQuery = "Select * from Person";

    1 reference
    public void AddPerson(PersonModel person)
    {
        try
        {
            if(connection.State != System.Data.ConnectionState.Closed)
            {
                connection.Close();
            }
            connection.Open();
            SqlCommand command = new SqlCommand(addPersonQuery, connection);
            command.Parameters.Add("@name", person.Name);
            command.Parameters.Add("@age", person.Age);
            command.ExecuteNonQuery();
            connection.Close();
        }
        catch (Exception ex)
        {

            throw;
        }
    }
```

2. **Vizualizare**

Pentru a doua optiune din interfata, vizualizarea persoanelor, pasii sunt aceiasi, singura diferenta este ca acum controllerul trebuie sa intoarca si date catre view, persoanele pe care le-a citit din baza de date. Functia din DAL intoarce datele necesare, iar controller-ul le trimite catre view:

```
1 reference
void _view_GetPeople(object sender, EventArgs e)
{
    var people = new PersonDAL().GetPeople();
    _view.DisplayPeople(people);
}
```

Patternul MVC reprezinta structurarea proiectului astfel incat functionalitatile si responsabilitatile sa nu se amestece, facand astfel mai usoare dezvoltarea proiectului si modificarile necesare pe parcursul dezvoltarii.

Clasele din aplicatie:

PersonDAL.cs

```
public class PersonDAL
   {
      SqlConnection connection = new SqlConnection("user id=sa;" +
                                 "password=licenta;server=Madalin\\SQLEXPRESS;" +
                                 "Trusted_Connection=yes;" +
                                 "database=MVCExample; " +
                                 "connection timeout=30");

      string addPersonQuery = "Insert into Person values (@name, @age)";
      string getPeopleQuery = "Select * from Person";

      public void AddPerson(PersonModel person)
      {
         try
         {
            if(connection.State != System.Data.ConnectionState.Closed)
            {
               connection.Close();
            }
            connection.Open();
            SqlCommand command = new SqlCommand(addPersonQuery, connection);
            command.Parameters.Add("@name", person.Name);
            command.Parameters.Add("@age", person.Age);
            command.ExecuteNonQuery();
            connection.Close();
         }
         catch (Exception ex)
         {

            throw;
         }
      }

      public DataTable GetPeople()
      {
         if (connection.State != System.Data.ConnectionState.Closed)
         {
            connection.Close();
```

```
        }
        connection.Open();

        SqlCommand command = new SqlCommand(getPeopleQuery, connection);
        SqlDataAdapter sa = new SqlDataAdapter(command);
        DataTable dt = new DataTable();
        sa.Fill(dt);

        connection.Close();

        return dt;
    }
  }
```

PersonController.cs

```
public class PersonController
{
    IPersonView _view;

    1 reference
    public PersonController(IPersonView view)
    {
        _view = view;
        _view.AddPerson += _view_AddPerson;
        _view.GetPeople += _view_GetPeople;
    }
    1 reference
    void _view_AddPerson(object sender, Model.PersonModel e)
    {
        new PersonDAL().AddPerson(e);
    }

    1 reference
    void _view_GetPeople(object sender, EventArgs e)
    {
        var people = new PersonDAL().GetPeople();
        _view.DisplayPeople(people);
    }
}
```

PersonView.cs

```
public partial class PersonView : Form, IPersonView
{
    public event EventHandler<PersonModel> AddPerson;
    public event EventHandler GetPeople;
    1 reference
    public PersonView()
    {
        InitializeComponent();
    }

    1 reference
    private void btnAdd_Click(object sender, EventArgs e)
    {
        if(AddPerson != null)
        {
            PersonModel person = new PersonModel();
            person.Name = txtName.Text;
            person.Age = Convert.ToInt32(txtAge.Text);

            AddPerson(this, person);
        }
    }

    1 reference
    private void btnLoad_Click(object sender, EventArgs e)
    {
        if(GetPeople != null)
        {
            GetPeople(this, new EventArgs());
        }
    }

    2 references
    public void DisplayPeople(DataTable people)
    {
        dataGridView1.DataSource = people;
    }
```

PersonModel.cs

```
public class PersonModel
{
    0 references
    public int ID { get; set; }
    2 references
    public string Name { get; set; }
    2 references
    public int Age { get; set; }
}
```

Controller-ul nu comunica direct cu fereastra, ci cu o interfata implementata de clasa ferestrei, in cazul nostru IPersonView.

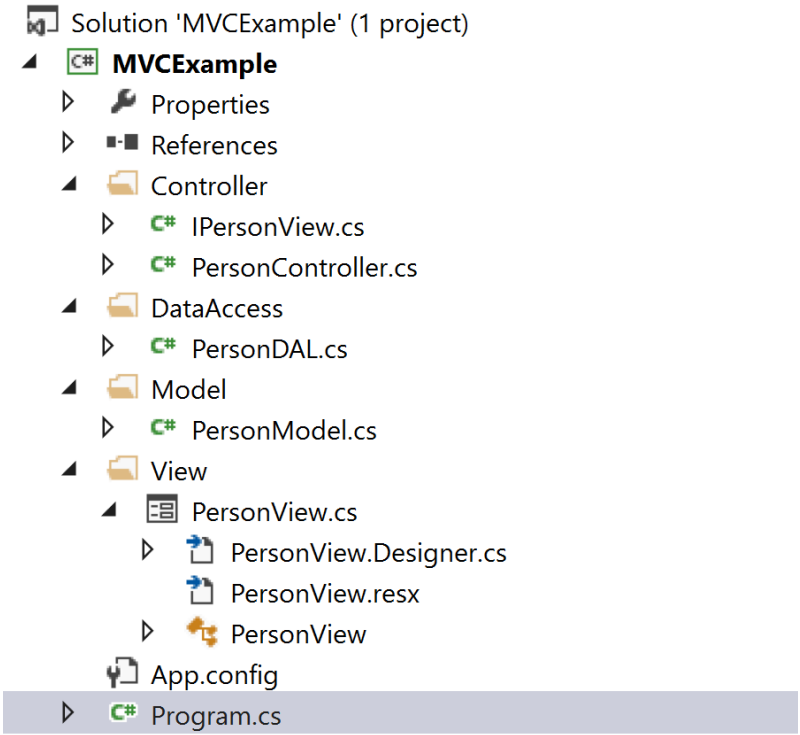
Aceasta are urmatoarele componente:

```
public interface IPersonView
{
    event EventHandler<PersonModel> AddPerson;
    event EventHandler GetPeople;
    2 references
    void DisplayPeople(DataTable people);
}
```

Legatura se face prin instantierea ferestrei la inceput, apoi trimiterea ei ca parametru catre controller(pe constructor). Astfel clasa Program.cs se modifica astfel:

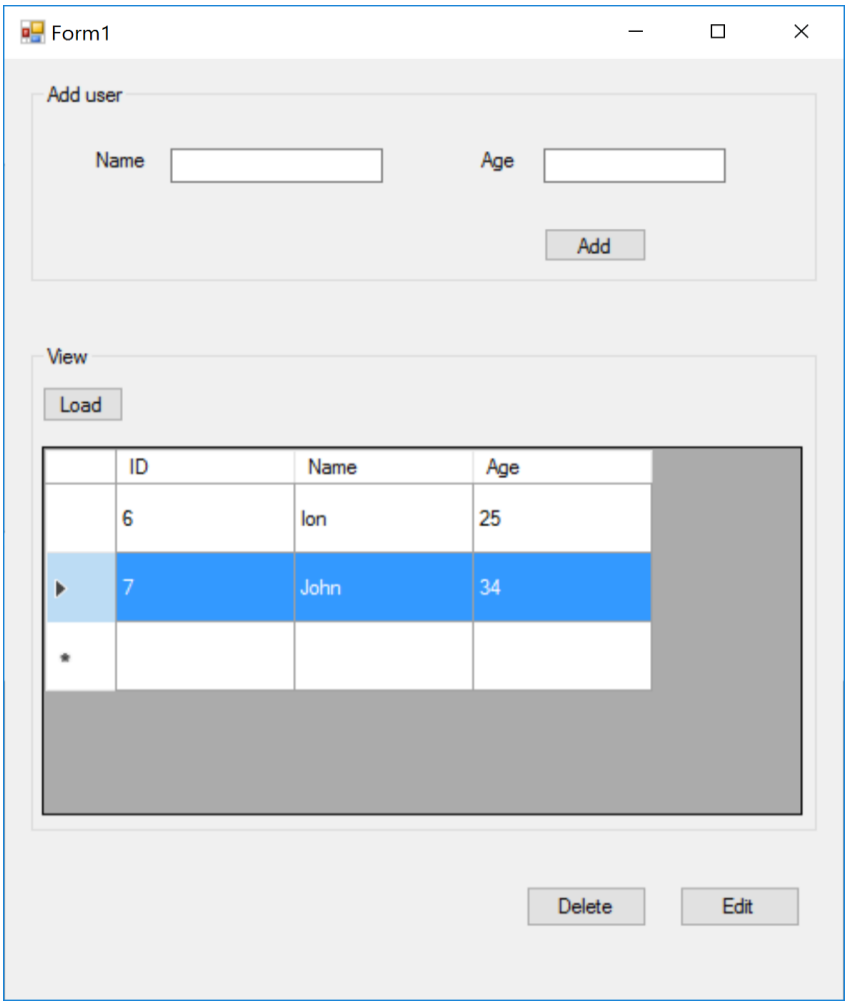
```
static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        PersonView view = new PersonView();
        PersonController controller = new PersonController(view);
        Application.Run(view);
    }
}
```

La final solutia trebuie sa aiba urmatoarea structura:


Solution 'MVCExample' (1 project)
MVCExample
Properties
References
Controller
IPersonView.cs
PersonController.cs
DataAccess
PersonDAL.cs
Model
PersonModel.cs
View
PersonView.cs
PersonView.Designer.cs
PersonView.resx
PersonView
App.config
Program.cs

Teme

1. Sa se adauge un buton care sa stearga persoana selectata din tabel.
2. In aplicatia existenta sa se adauge un buton “Edit”, care sa deschida o fereastra de editare pentru persoana selectata din tabel.S



Nota: Fiecare fereastra are propriul controller, nu se foloseste acelasi controller pentru mai multe ferestre