ЗА ДЪРЖАВЕН ИЗПИТ

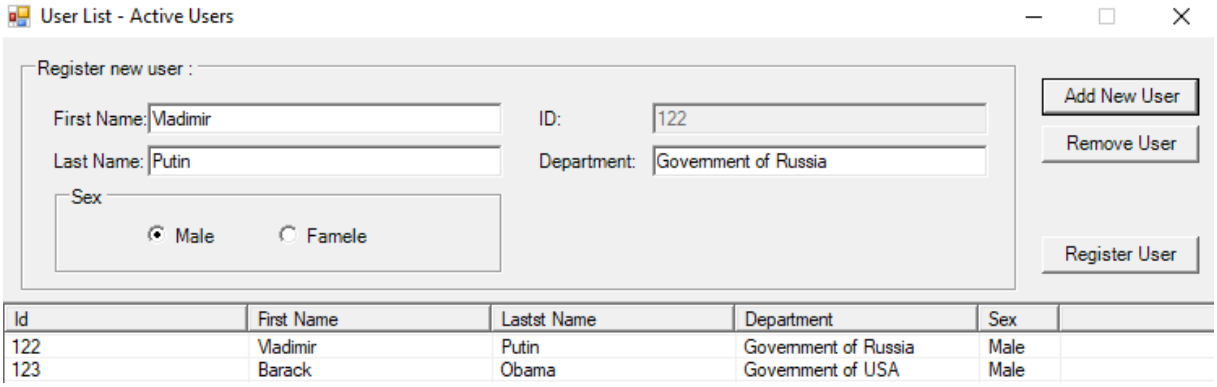
– ЧАСТ ПО ПРАКТИКА НА ПРОФЕСИЯТА

МЕНИДЖМЪНТ НА ПОТРЕБИТЕЛСКА ИНФОРМАЦИЯ

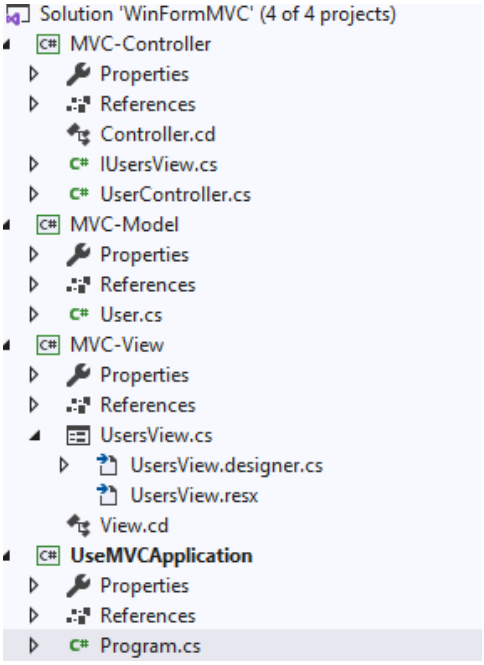
(Разработване на многослойно приложение.)

Задание:

- Да се разработи графично приложение.
- Да се използва архитектура MVC.
- Да се показва списък с контакти, с възможност да добавяте, променяте и изтривате съществуващи контакти. Всички клиенти имат лична карта, собствено име, фамилия и пол.
- Да се използва валидация на данните за потребителя.



Файлова структура:



Приложения с графичен интерфейс

Мениджърът на потребителска информация е приложение, в което можете да съхранявате информация за контакт на клиентите си. Приложението показва списък с контакти и ви позволява да добавяте, променяте и изтривате съществуващи контакти. Всички клиенти имат лична карта, собствено име, фамилия и пол. Екранът, който операторът на това приложение използва, за да поддържа списъка си с клиенти, може да изглежда така:

Описание на компонентите:

```
this.txtID = new System.Windows.Forms.TextBox();
this.txtLastName = new System.Windows.Forms.TextBox();
this.lblID = new System.Windows.Forms.Label();
this.lblLastName = new System.Windows.Forms.Label();
this.btnRemove = new System.Windows.Forms.Button();
this.grpDetails = new System.Windows.Forms.GroupBox();
this.grbSex = new System.Windows.Forms.GroupBox();
this.rdFamele = new System.Windows.Forms.RadioButton();
this.rdMale = new System.Windows.Forms.RadioButton();
this.txtDepartment = new System.Windows.Forms.TextBox();
this.lblDepartment = new System.Windows.Forms.Label();
this.txtFirstName = new System.Windows.Forms.TextBox();
this.lblFirstName = new System.Windows.Forms.Label();
this.btnAdd = new System.Windows.Forms.Button();
this.btnRegister = new System.Windows.Forms.Button();
this.grdUsers = new System.Windows.Forms.ListView();
```

За да отделим логиката от изгледа, трябва да накараме изгледа да се чувства възможно най-безпомощен, така че бихме предпочели да накараме контролера да свърши цялата работа и просто да предадем на изгледа някои прости команди, които не изискват допълнителна обработка . Според дизайна, ще направим това, като дефинираме интерфейс, **IUsersView**, който View трябва да реализира. Този интерфейс съдържа само подписите на свойства/методи, които трябва да използваме.

```
using System;
using WinFormMVC.Model;

namespace WinFormMVC.Controller
{
   public interface IUsersView
   {
      void SetController(UsersController controller);
      void ClearGrid();
      void AddUserToGrid(User user);
      void UpdateGridWithChangedUser(User user);
      void RemoveUserFromGrid(User user);
      string GetIdOfSelectedUserInGrid();
      void SetSelectedUserInGrid(User user);

      string FirstName     { get; set; }
      string LastName      { get; set; }
      string ID    { get; set; }
      string Department    { get; set; }
      User.SexOfPerson Sex { get; set; }
      bool CanModifyID     {      set; }
```

```
    }
}
```

Сега имаме доста добър интерфейс с редица методи. Дори ако MVC моделът официално декларира, че контролерът трябва да получава събитията и да действа по изгледа, често е по-практично и по-лесно изгледът да се абонира за събитията и след това да делегира обработката на контролера.

Накрая показвам действителната реализация на контролера (вижте UsersController класа). Той свързва модела (User клас) с View (UserView клас).

```
public class UsersController
{

   IUsersView _view;
   IList      _users;
   User       _selectedUser;


   public UsersController(IUsersView view, IList users)
   {
     _view = view;
     _users = users;
     view.SetController(this);
   }

   public IList Users
   {
     get { return ArrayList.ReadOnly(_users); }
   }

   private void updateViewDetailValues(User usr)
   {
     _view.FirstName   =  usr.FirstName;
     _view.LastName    =  usr.LastName;
     _view.ID  =  usr.ID;
     _view.Department  =  usr.Department;
     _view.Sex =  usr.Sex;
   }

   private void updateUserWithViewValues(User usr)
   {
     usr.FirstName     =  _view.FirstName;
     usr.LastName      =  _view.LastName;
     usr.ID    =  _view.ID;
     usr.Department    =  _view.Department;
     usr.Sex   =  _view.Sex;
   }

   public void LoadView()
   {
     _view.ClearGrid();
     foreach (User usr in _users)
     _view.AddUserToGrid(usr);
```

```
    _view.SetSelectedUserInGrid((User)_users[0]);
  }
  public void SelectedUserChanged(string selectedUserId)
  {
    foreach (User usr in this._users)
    {
      if (usr.ID == selectedUserId)
      {
        _selectedUser = usr;
        updateViewDetailValues(usr);
        _view.SetSelectedUserInGrid(usr);
        this._view.CanModifyID = false;
        break;
      }
    }
  }

  public void AddNewUser()
  {
    _selectedUser = new User(""   /*firstname*/,
    ""  /*lastname*/,
    "" /*id*/,
    ""/*department*/,
    User.SexOfPerson.Male/*sex*/);

    this.updateViewDetailValues(_selectedUser);
    this._view.CanModifyID = true;
  }

  public void RemoveUser()
  {
    string id = this._view.GetIdOfSelectedUserInGrid();
    User userToRemove = null;

     if (id != "")
     {
       foreach (User usr in this._users)
       {
         if (usr.ID == id)
         {
           userToRemove = usr;
         break;
         }
       }
       if (userToRemove != null)
       {
         int newSelectedIndex = this._users.IndexOf(userToRemove);

         this._users.Remove(userToRemove);
```

Част от изгледа:

Зареждане на изгледа със списъка с потребители.
Изгледа трябва да внедри интерфейса IUsersView. Подмножество от реализацията е показано в следния код:

```
namespace WinFormMVC.View
{
   public partial class UsersView : Form, IUsersView
   {
     public void SetController(UsersController controller)
      {
        _controller = controller;
      }
```

Функцията SetController()член на UsersView ни позволява да кажем на View към коя инстанция на контролера трябва да препрати събитията и всички манипулатори на събития просто извикват съответния метод „event“ на контролера. Както можете да видите тук, UsersView също зависи от абстракциите.

Ще се реализират няколко метода от интерфейса IUsersView, които използват обекта User:

```
public void AddUserToGrid(User usr)
{
   ListViewItem parent;
   parent = this.grdUsers.Items.Add(usr.ID);
   parent.SubItems.Add(usr.FirstName);
   parent.SubItems.Add(usr.LastName);
   parent.SubItems.Add(usr.Department);
   parent.SubItems.Add(Enum.GetName(typeof(User.SexOfPerson), usr.Sex));
}

public void UpdateGridWithChangedUser(User usr)
{
   ListViewItem rowToUpdate = null;

   foreach (ListViewItem row in this.grdUsers.Items)
   {
      if (row.Text == usr.ID)
      {
         rowToUpdate = row;
      }
   }

   if (rowToUpdate != null)
   {
      rowToUpdate.Text = usr.ID;
      rowToUpdate.SubItems[1].Text = usr.FirstName;
      rowToUpdate.SubItems[2].Text = usr.LastName;
      rowToUpdate.SubItems[3].Text = usr.Department;
      rowToUpdate.SubItems[4].Text = Enum.GetName(typeof(User.SexOfPerson), usr.Sex);
   }
}
```

```
public void RemoveUserFromGrid(User usr)
{

   ListViewItem rowToRemove = null;

   foreach (ListViewItem row in this.grdUsers.Items)
   {
      if (row.Text == usr.ID)
      {
         rowToRemove = row;
      }
   }

   if (rowToRemove != null)
   {
      this.grdUsers.Items.Remove(rowToRemove);
      this.grdUsers.Focus();
   }
}

public string GetIdOfSelectedUserInGrid()
{
   if (this.grdUsers.SelectedItems.Count > 0)
      return this.grdUsers.SelectedItems[0].Text;
   else
      return "";
}

public void SetSelectedUserInGrid(User usr)
{
   foreach (ListViewItem row in this.grdUsers.Items)
   {
      if (row.Text == usr.ID)
      {
         row.Selected = true;
      }
   }
}

public string FirstName
{
   get { return this.txtFirstName.Text; }
   set { this.txtFirstName.Text = value; }
}

public string LastName
{
   get { return this.txtLastName.Text; }
   set { this.txtLastName.Text = value; }
}

public string ID
{
   get { return this.txtID.Text; }
```

```
    set { this.txtID.Text = value; }
}


public string Department
{
    get { return this.txtDepartment.Text; }
    set { this.txtDepartment.Text = value; }
}

public User.SexOfPerson Sex
{
    get
    {
        if (this.rdMale.Checked)
            return User.SexOfPerson.Male;
        else
            return User.SexOfPerson.Female;
    }
    set
    {
        if (value == User.SexOfPerson.Male)
            this.rdMale.Checked = true;
        else
            this.rdFamele.Checked = true;
    }
}

public bool CanModifyID
{
    set { this.txtID.Enabled = value; }
}

        ...}
```

Част от модела – User.cs:

Този потребителски клас е клас модел. Моделът е независим от потребителския интерфейс. Той не знае дали се използва от текстов, графичен или уеб интерфейс. Моделът поддържа само състоянието в паметта в структуриран формат. Както можете да видите, класът съдържа само членове на частни данни и публичните интерфейси (свойства), достъпни за клиентския код:

```
using System;

namespace WinFormMVC.Model
{
 public class User
 {
   public enum SexOfPerson
   {
     Male   = 1,
     Female = 2
   }
```

```
private string    _FirstName;
public string FirstName
{
 get { return _FirstName; }
 set
 {
  if (value.Length > 50)
   Console.WriteLine("Error! FirstName must be less than 51 characters!");
  else
   _FirstName = value;
 }
}

private string _LastName;
public string LastName
{
  get { return _LastName; }
  set
  {
   if (value.Length > 50)
    Console.WriteLine("Error! LastName must be less than 51 characters!");
   else
    _LastName = value;
 }
}

private string _ID;
public string ID
{
 get { return _ID; }
 set
 {
   if (value.Length > 9)
     Console.WriteLine("Error! ID must be less than 10 characters!");
   else
    _ID = value;
 }
}

private string _Department;
public string Department
{
    get { return _Department; }
    set { _Department = value; }
}

private SexOfPerson _Sex;
public SexOfPerson Sex
{
    get { return _Sex; }
    set { _Sex = value; }
}
```

```
  public User(string firstname, string lastname, string id, string department, SexOfPerson sex)
  {
     FirstName   = firstname;
     LastName    = lastname;
     ID  = id;
     Department  = department;
     Sex = sex;
  }
 } }
```

Част от Клиента – Program.cs:

```
using System.Collections;
using  WinFormMVC.Model;
using  WinFormMVC.View;
using  WinFormMVC.Controller;

namespace UseMVCApplication
{
static class Program
{

   [STAThread]
   static void Main()
   {
      //Създаване на изглед View
      UsersView view = new UsersView();
      view.Visible = false;

      //Списък с потребители
      IList users = new ArrayList();

      users.Add(new User("Vladimir",   "Putin",
    "122",    "Government of Russia",
         User.SexOfPerson.Male));
      users.Add(new User("Barack",     "Obama",
          "123",    "Government of USA",
          User.SexOfPerson.Male));
      users.Add(new User("Stephen",    "Harper",
         "124",    "Government of Canada",
         User.SexOfPerson.Male));
      users.Add(new User("Jean",       "Charest",
         "125",    "Government of Quebec",
          User.SexOfPerson.Male));
      users.Add(new User("David",      "Cameron",
         "126",    "Government of United Kingdom",
         User.SexOfPerson.Male));
      users.Add(new User("Angela",     "Merkel",
         "127",    "Government of Germany",
         User.SexOfPerson.Female));
      users.Add(new User("Nikolas",    "Sarkozy",\
         "128",    "Government of France",
         User.SexOfPerson.Male));
```

```
        users.Add(new User("Silvio",     "Berlusconi",
            "129",    "Government of Italy",
            User.SexOfPerson.Male));
        users.Add(new User("Yoshihiko",  "Noda",
            "130",    "Government of Japan",
            User.SexOfPerson.Male));

        //Създаване на Controller и преминаване през двата
        //параметъра: изгледа и list от потребители (models)
        UsersController controller = new UsersController(view, users);
        controller.LoadView();
        view.ShowDialog();
        }
    }}
```

Обработка на събитията – UsersView.cs:

```
using WinFormMVC.Controller;
using WinFormMVC.Model;

namespace WinFormMVC.View
{
    public partial class UsersView : Form, IUsersView
    {
        public UsersView()
        {
            InitializeComponent();
        }

        UsersController _controller;

        #region Events raised  back to controller

        private void btnAdd_Click(object sender, EventArgs e)
        {
            this._controller.AddNewUser();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            this._controller.RemoveUser();
        }

        private void btnRegister_Click(object sender, EventArgs e)
        {
            this._controller.Save();
        }

        private void grdUsers_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (this.grdUsers.SelectedItems.Count > 0)
                this._controller.SelectedUserChanged(this.grdUsers.SelectedItems[0].Text);
        }
```

```
#endregion

#region ICatalogView implementation

public void SetController(UsersController controller)
{
   _controller = controller;
}

public void ClearGrid()
{

   this.grdUsers.Columns.Clear();

   this.grdUsers.Columns.Add("Id",          150,  HorizontalAlignment.Left);
   this.grdUsers.Columns.Add("First Name",  150,  HorizontalAlignment.Left);
   this.grdUsers.Columns.Add("Lastst Name", 150,  HorizontalAlignment.Left);
   this.grdUsers.Columns.Add("Department",  150,  HorizontalAlignment.Left);
   this.grdUsers.Columns.Add("Sex",         50,   HorizontalAlignment.Left);


   this.grdUsers.Items.Clear();
}

public void AddUserToGrid(User usr)
{
   ListViewItem parent;
   parent = this.grdUsers.Items.Add(usr.ID);
   parent.SubItems.Add(usr.FirstName);
   parent.SubItems.Add(usr.LastName);
   parent.SubItems.Add(usr.Department);
   parent.SubItems.Add(Enum.GetName(typeof(User.SexOfPerson), usr.Sex));
}

public void UpdateGridWithChangedUser(User usr)
{
   ListViewItem rowToUpdate = null;

   foreach (ListViewItem row in this.grdUsers.Items)
   {
      if (row.Text == usr.ID)
      {
         rowToUpdate = row;
      }
   }

   if (rowToUpdate != null)
   {
      rowToUpdate.Text = usr.ID;
      rowToUpdate.SubItems[1].Text = usr.FirstName;
      rowToUpdate.SubItems[2].Text = usr.LastName;
      rowToUpdate.SubItems[3].Text = usr.Department;
      rowToUpdate.SubItems[4].Text = Enum.GetName(typeof(User.SexOfPerson), usr.Sex);
   }
```

```
    }

    public void RemoveUserFromGrid(User usr)
    {

        ListViewItem rowToRemove = null;

        foreach (ListViewItem row in this.grdUsers.Items)
        {
            if (row.Text == usr.ID)
            {
                rowToRemove = row;
            }
        }

        if (rowToRemove != null)
        {
            this.grdUsers.Items.Remove(rowToRemove);
            this.grdUsers.Focus();
        }
    }

    public string GetIdOfSelectedUserInGrid()
    {
        if (this.grdUsers.SelectedItems.Count > 0)
            return this.grdUsers.SelectedItems[0].Text;
        else
            return "";
    }

    public void SetSelectedUserInGrid(User usr)
    {
        foreach (ListViewItem row in this.grdUsers.Items)
        {
            if (row.Text == usr.ID)
            {
                row.Selected = true;
            }
        }
    }

    public string FirstName
    {
        get { return this.txtFirstName.Text; }
        set { this.txtFirstName.Text = value; }
    }

    public string LastName
    {
        get { return this.txtLastName.Text; }
        set { this.txtLastName.Text = value; }
    }

    public string ID
    {
```

```
            get { return this.txtID.Text; }
            set { this.txtID.Text = value; }
        }


        public string Department
        {
            get { return this.txtDepartment.Text; }
            set { this.txtDepartment.Text = value; }
        }

        public User.SexOfPerson Sex
        {
            get
            {
                if (this.rdMale.Checked)
                    return User.SexOfPerson.Male;
                else
                    return User.SexOfPerson.Female;
            }
            set
            {
                if (value == User.SexOfPerson.Male)
                    this.rdMale.Checked = true;
                else
                    this.rdFamele.Checked = true;
            }
        }

        public bool CanModifyID
        {
            set { this.txtID.Enabled = value; }
        }

        #endregion
    }
}
```