Пояснительная записка к контрольному домашнему заданию по дисциплине «Программирование»

Тема проекта: Реестр ГИБДД

Адрес репозитория: github.com/igeudesnev

Аннотация: Добавление, редактирование, удаление и чтение в таблице списка автолюбителей

Используемые технологии: сериализация с помощью json

Используемые библиотеки: System.Collections.Generic – использование коллекций,

System.Windows – использование окон в том числе диалоговых,

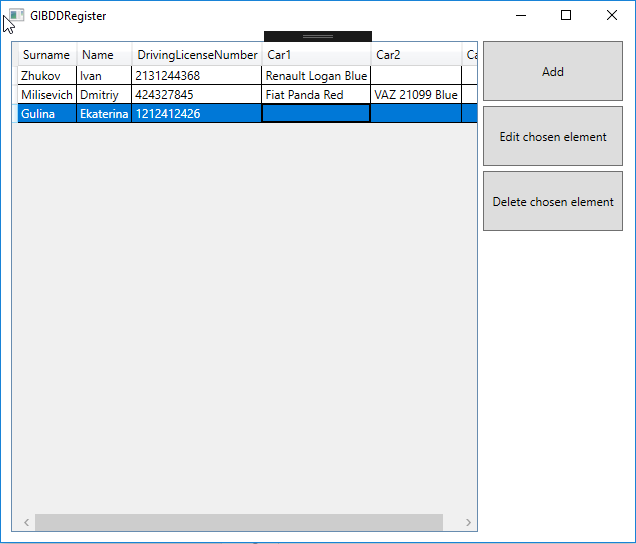
System.IO – работа с файловыми потоками,

System.Runtime.Serialization – использование сериализации при записи и чтения файла,

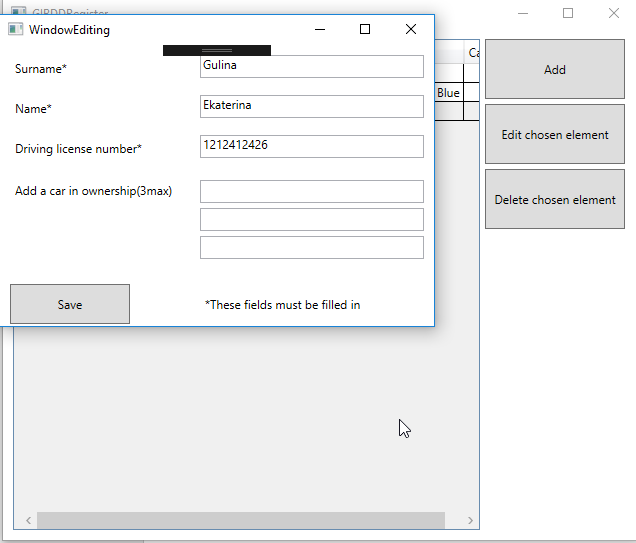
System.Runtime.Serialization.Json – использование именной сериализации Json.

Интерфейс программы:

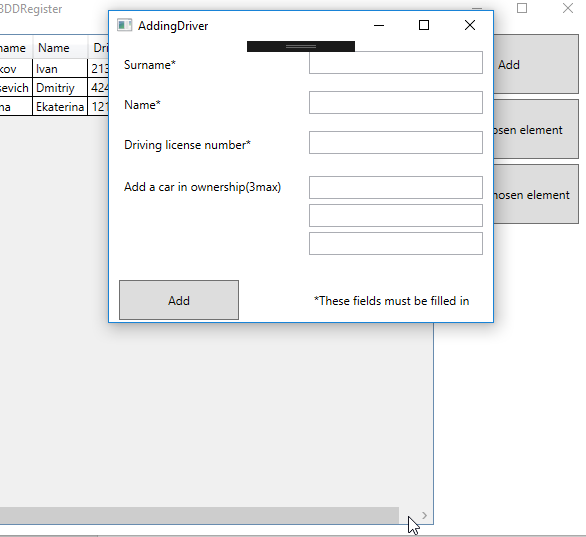
Главное окно с таблицей и кнопками «Добавить», «Редактировать» и «Удалить»



Высплывающее диалоговое окно редактирования



Всплывающее диалоговое окно добавление объекта



Содержание классов:

public class Driver

{

private string \_surname;

[DataMember]

public string Surname

{

get { return \_surname; }

set { \_surname = value; }

}

private string \_name;

[DataMember]

public string Name

{

get { return \_name; }

set { \_name = value; }

}

private int \_drivingLicenseNumber;

[DataMember]

public int DrivingLicenseNumber

{

get { return \_drivingLicenseNumber; }

set { \_drivingLicenseNumber = value; }

}

private string \_car1;

[DataMember]

public string Car1

{

get { return \_car1; }

set { \_car1 = value; }

}

private string \_car2;

[DataMember]

public string Car2

{

get { return \_car2; }

set { \_car2 = value; }

}

private string \_car3;

[DataMember]

public string Car3

{

get { return \_car3; }

set { \_car3 = value; }

}

public Driver(string surname, string name, int drivingLicenseNumber)

{

\_surname = surname;

\_name = name;

\_drivingLicenseNumber = drivingLicenseNumber;

}

public Driver(string surname, string name, int drivingLicenseNumber, string car1)

: this(surname, name, drivingLicenseNumber)

{

\_car1 = car1;

}

public Driver(string surname, string name, int drivingLicenseNumber, string car1, string car2)

: this(surname, name, drivingLicenseNumber, car1)

{

\_car2 = car2;

}

public Driver(string surname, string name, int drivingLicenseNumber, string car1, string car2, string car3)

: this(surname, name, drivingLicenseNumber, car1, car2)

{

\_car3 = car3;

}

}

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

ReadToList();

ListToGrid();

}

DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(List<Driver>));

static public List<Driver> \_drivers = new List<Driver>();

private void ListToGrid()

{

dataGridDrivers.ItemsSource = null;

dataGridDrivers.ItemsSource = \_drivers;

}

private void ListToFile()

{

using (FileStream fs = new FileStream("drivers.txt", FileMode.Create, FileAccess.Write))

{

jsonFormatter.WriteObject(fs, \_drivers);

}

}

private void ListToFileAndGrid()

{

ListToFile();

ListToGrid();

}

private void ReadToList()

{

if (File.Exists("drivers.txt"))

{

using (FileStream fs = new FileStream("drivers.txt", FileMode.Open, FileAccess.Read))

{

\_drivers = (List<Driver>)jsonFormatter.ReadObject(fs);

}

}

}

private void buttonAdding\_Click(object sender, RoutedEventArgs e)

{

AddingDriver addingdriver = new AddingDriver();

addingdriver.ShowDialog();

ListToFileAndGrid();

}

private void buttonDelete\_Click(object sender, RoutedEventArgs e)

{

if (dataGridDrivers.SelectedIndex != -1)

{

\_drivers.RemoveAt(dataGridDrivers.SelectedIndex);

}

else { MessageBox.Show("Choose string"); }

ListToFileAndGrid();

}

private void buttonEdit\_Click(object sender, RoutedEventArgs e)

{

int index = dataGridDrivers.SelectedIndex;

if (index != -1)

{

WindowEditing editingdriver = new WindowEditing(dataGridDrivers.SelectedItem as Driver,

dataGridDrivers.SelectedIndex);

editingdriver.ShowDialog();

}

else MessageBox.Show("Choose string");

ListToFileAndGrid();

}

}

public partial class AddingDriver : Window

{

DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(List<Driver>));

public AddingDriver()

{

InitializeComponent();

}

private void buttonAdd\_Click(object sender, RoutedEventArgs e)

{

try

{

if (string.IsNullOrWhiteSpace(textBoxSurname.Text) != true &&

string.IsNullOrWhiteSpace(textBoxName.Text) != true &&

string.IsNullOrWhiteSpace(textBoxDrivinglicense.Text) != true)

{

textBoxDrivinglicense.Text.Replace(" ", "");

Driver driver = new Driver(textBoxSurname.Text, textBoxName.Text, int.Parse(textBoxDrivinglicense.Text),

textBoxCar1.Text, textBoxCar2.Text, textBoxCar3.Text);

MainWindow.\_drivers.Add(driver);

MessageBox.Show("Saved");

Close();

}

else { MessageBox.Show("Fields should not be empty"); }

}

catch { MessageBox.Show("Data entered incorrectly"); }

}

}

public partial class WindowEditing : Window

{

int index;

public WindowEditing(Driver driver, int index)

{

this.index = index;

InitializeComponent();

textBoxSurname.Text = driver.Surname;

textBoxName.Text = driver.Name;

textBoxDrivinglicense.Text = driver.DrivingLicenseNumber.ToString();

textBoxCar1.Text = driver.Car1;

textBoxCar2.Text = driver.Car2;

textBoxCar3.Text = driver.Car3;

}

private void buttonSave\_Click(object sender, RoutedEventArgs e)

{

try

{

if (string.IsNullOrWhiteSpace(textBoxSurname.Text) != true &&

string.IsNullOrWhiteSpace(textBoxName.Text) != true &&

string.IsNullOrWhiteSpace(textBoxDrivinglicense.Text) != true)

{

textBoxDrivinglicense.Text.Replace(" ", "");

Driver driver = new Driver(textBoxSurname.Text, textBoxName.Text, int.Parse(textBoxDrivinglicense.Text),

textBoxCar1.Text, textBoxCar2.Text, textBoxCar3.Text);

MainWindow.\_drivers[index] = driver;

MessageBox.Show("Saved");

Close();

}

else { MessageBox.Show("Fields should not be empty"); }

}

catch { MessageBox.Show("Data entered incorrectly"); }

}

}

Список источников:

<https://msdn.microsoft.com/library>

<https://metanit.com/sharp>