Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “Об’єктно-орієнтоване програмування”

КАТРОТЕКА ІНТЕРПОЛУ

Керівник, асистент каф. ПІ, Ляпота В.М.

Студент гр. ПІ-15-2 Манаков К.А.

Комісія:

Зав. каф. ПІ, Дудар З.В.

к. т. н, проф., Бондарев Владимир Михайлович

Харків 2014

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: Об’єктно-орієнтоване програмування

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2*. Семестр 2 .

ЗАВДАННЯ

на курсовий проект студента

Манакова Костянтина Андрійовича

(Прізвище, Ім'я, По батькові)

1. Тема проекту:

Картотека інтерполу.

1. Термін здачі студентом закінченого проекту: ***“28” - травня - 2016р*.**
2. Вихідні дані до проекту:

Специфікація програми, методичні вказівки до виконання курсової роботи

4. Зміст розрахунково-пояснювальної записки:

Вступ, специфікація програми, проектна специфікація, посібник користувача, висновки

* Перелік графічного матеріалу

Екранні форми, схема об’єктної моделі, діаграми класів

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 20-05-2016р. |
| 2 | Аналіз предметної області | 26-05-2016р. |
| 3 | Розробка постановки задачі | 27-05-2016р. |
| 4 | Розробка об’єктної моделі | 27-05-2016р. |
| 5 | Кодування програмної системи | 27-05-2016р. - 28-05-2016р. |
| 6 | Тестування і доопрацювання розробленої програмної системи. | 28-05-2016р. |
| 7 | Оформлення пояснювальної записки | 28-05-2016р. |
| 8 | Публічний захист проекту перед комісією | 28-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Прізвище, Ім'я, По батькові)

« 27 »\_лютого\_\_ 2016р.

РЕФЕРАТ

Пояснювальна записка: 26 с., 11 рис., 1 додаток.

Мета роботи - створення програмної системи під назвою «Картотека інтерполу».

Метод розробки - Microsoft VisualStudio 2015, WPF, NET Framework 4.5

В результаті розробки отримана програмна система під назвою «Картотека інтерполу» для роботи з інформаційною базою даних. Система дає можливість додавати злочинців до бази даних с вихідними даними: ПІБ зочинця, дата народження, краіна проживання, знання мов, подільників. Програма також має пошук злочинців та формування різних видів списку.

КУРСОВА РОБОТА, ПРОГРАМНА ОБ`ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, MICROSOFT VISUAL STUDIO 2015, WPF, ІНТЕРПОЛ.

ЗМІСТ

ВСТУП……………………………………………………………...…6

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ…………………………………....8

2 АРХІТЕКТУРА ПРОГРАМИ..…………………………………...17

2.1 Загальна архітектура…….…………………………….…17

2.2 Інтерфейс користувача……..…………………………….21

3 ПОСІБНИК КОРИСТУВАЧА………………………………….....24

ВИСНОВКИ……………………………………………………….…26

ПЕРЕЛІК ПОСИЛАНЬ………………………………………………27

Додаток А Код програми……………………………………………28

ВВЕДЕНИЕ

Алгоритмизация в борьбе с общеуголовной преступностью важная цель на сегодняшний день. Программные системы для автоматизации поиска преступников позволят оптимизировать работу полиции, вести актуальную базу преступников. Внедрение таких систем даёт целый ряд преимуществ: увеличение эффективности работы полиции, упрощение поиска и редактирования данных о преступниках.

В данной курсовой работе реализуется программа на языке программирования C#. Целью данной работы есть написание программы «Картотека Интерпола» в среде Microsoft Visual Studio (MVS), для закрепления знаний полученных на лекциях данной дисциплины. Программа использует в качестве базы данных текстовой файл "data.txt", что позволяет хранить данные об активных преступниках в одном файле.

Программа «Картотека Интерпола», может быть применима в организациях по розыску людей, для ведения учета преступников, работы с их данными. Программа запрашивает данные, которые необходимо ввести, а далее сам пользователь манипулирует ими. Программа также контролирует пользователя о его действиях, индицируя и сигнализируя те или иные ошибки**.**

Программа также предусмотрена возможность выборочного поиска преступников, по всем основным характеристикам. Также программа отображает на экране всю интересующую информацию о преступнике, и позволяет определить занимается ли преступник на данный момент криминальной деятельностью.

Задание на выполнение работы:

а) Использование наследования для структуризации классов ;

б) Исследование предметной области с целью выявления основных сущностей и их атрибутов;

в) Применение принципа инкапсуляции к классам;

г) Временная реализация БД через JSON-сериализацию.

Объектно -ориентированный подход требует глубокого понимания основных принципов , или , иначе , концепций , на которых он базируется . К числу основных понятий ООП обычно относят наследование , инкапсуляцию и полиморфизм [ 1] . Объектно - ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования .

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015 Community. Язык программирования - C # .

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ

Спецификацией данной программы является создания/редактирования базы данных преступников, с работой над ней в программе (отображение, редактирование, удаление, поиском по нескольким категориям).

Главными в программе функциями есть:

* Функция для изменения информации о преступнике;
* Функция для добавления нового преступника;
* Функция удаления преступника;
* Функция для поиска преступника.
* Фунция для синхронизации базы с локальным веб-сервером

1.1 Перечень функций

В данном разделе выведен перечень всех функций программы:

а) Новый Преступник. Данная функция позволяет создать нового преступника и поместить его в базу данных программы. Для ее использования необходимо нажать кнопку "Add".

б)Поменять Сведения. С помощью этой функции вы можете легко поменять сведения о выбранном преступнике в открывшемся окне. Для использования этой функции необходимо кликнуть правой кнопкной мыши по преступнику в таблице и выбрать пункт "View/Edit".

в)Удалить Анкету: Эта функция позволяет удалить выбранного преступника с базы данных программы. Для того, чтобы ее использовать, необходимо кликнуть правой кнопкой мыши по преступнику в таблице и выбрать пункт "Delete".

д)Поиск Преступника (ов). Функция позволяет найти преступника по введенной информации. Для использования функции необходимо ввести в поле «Global Search...» известную информацию или нажать на кнопку "Find" и получить полноценный интерфейс ввода данных.

г) Синхронизация с веб-сервером. Функция позволяет синхронизировать данные с разных клиентов с веб-сервером. Сервер принимает коллекцию преступников, отсутствующие копирует, те, что были удалены на клиенте, удаляет у себя, для всех остальных выбирает те данные, которые были обновлены позже. Для вызова нужно нажать кнопку "Sync".

1.2 Пользовательский Интерфейс

Программа использует класс «G» который загружает/сохраняет все данные из файла в программу, и при необходимости может быть легко изменен для иных целей.

При загрузке программа это загружает данные о преступниках. Если файл с данными о преступниках не будет найден, программа создаст новый. При работе с программой все данные о преступниках также хранит класс «G».

После удачного запуска программы откроется главное окно программы (рисунок 1.1).



Рисунок 1.1 – Главное окно программы

После завершения работы с программой следует нажать кнопку.

2 АРХИТЕКТУРА ПРОГРАММЫ

2.1 Архитектура программного продукта

В ООП главным элементом является класс, в состав котрого входит маса объектов с одинаковыми свойставми, операциями и отношениями [3].

В проекте есть три основных класса. В первом, "G" который отвечает за сохранение, загрузку и обработку данных о преступнике. . Метод «SaveToFile» c помощью Newton.JSON сериализации сохраняет базу в файл data.txt. Код метода:

public static void SaveToFile()

{

string json = JsonConvert.SerializeObject(new Package { Array = G.BadBoys, Deleted = G.Deleted });

using (StreamWriter file = new StreamWriter("data.txt", false))

{

file.Write(json);

}

Метод «LoadFromFile» загружает файл с данными для работы с ними. Код метода:

public static void ReadFromFile()

{

try

{

StreamReader file = new StreamReader("data.txt");

}

catch(FileNotFoundException)

{

StreamWriter file = new StreamWriter("data.txt", false);

}

using (StreamReader file = new StreamReader("data.txt"))

{

string json = file.ReadToEnd();

BadBoys = JsonConvert.DeserializeObject<Package>(json).Array;

if(BadBoys == null)

BadBoys = new List<BadBoy>();

}

}

За синхронизацию отвечает событие нажатия кнопки.

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

{

string url = "<http://sync.php/sync.php>";

HttpWebRequest request = (HttpWebRequest)

WebRequest.Create(url); request.KeepAlive = false;

request.ProtocolVersion = HttpVersion.Version10;

request.Method = "POST";

var json = JsonConvert.SerializeObject(new Package{Array = G.BadBoys, Deleted = G.Deleted});

byte[] postBytes = Encoding.UTF8.GetBytes(json);

request.ContentType = "application/json; charset=UTF-8";

request.Accept = "application/json";

request.ContentLength = postBytes.Length;

Stream requestStream = request.GetRequestStream();

requestStream.Write(postBytes, 0, postBytes.Length);

requestStream.Close();

HttpWebResponse response = (HttpWebResponse)request.GetResponse();

string result;

using (StreamReader rdr = new StreamReader(response.GetResponseStream()))

{

result = rdr.ReadToEnd();

}

G.BadBoys = JsonConvert.DeserializeObject<Package>(result).Array;

G.Deleted = new List<int>();

this.dataGrid.ItemsSource = null;

this.dataGrid.ItemsSource = G.BadBoys;

G.SaveToFile();

}

Серверная часть(php):

<?php  
/\*\*  
 \* Created by PhpStorm.  
 \* User: Monyk  
 \* Date: 28.05.2016  
 \* Time: 1:35  
 \*/  
  
$data = $HTTP\_RAW\_POST\_DATA;  
$dataLocal = json\_decode($data, true);  
$badBoysLocal = $dataLocal['Array'];  
$deleted = $dataLocal['Deleted']; //$dataLocal->Deleted;  
$badBoysServer = json\_decode(file\_get\_contents("data.txt"), true)['Array'];  
if ($badBoysServer == null)  
{  
 $badBoysServer = $badBoysLocal;  
}  
else  
{  
 if ($deleted != null)  
 {  
 foreach ($deleted as $value)  
 {  
 $id = $value;  
 array\_walk(  
 $badBoysServer, function ($el, $key) use ($id, &$badBoysServer)  
 {  
 if ($el['id'] == $id)  
 {  
 array\_splice($badBoysServer, $key, 1);  
 }  
 }  
 );  
 }  
 }  
 if ($badBoysLocal != null)  
 {  
 foreach ($badBoysLocal as $which => $badBoyLocal)  
 {  
 $id = $badBoyLocal['id'];  
 array\_walk(  
 $badBoysServer, function ($el, $k) use ($which, $id, $badBoysLocal, $badBoyLocal, &$badBoysServer)  
 {  
 if ($el['id'] == $id)  
 //foreach ($badBoyLocal as $key => $property)  
 //if(new DateTime($badBoyLocal[$key.'UpdateTime']) > new DateTime($el[$key.'UpdateTime']))  
 {  
 array\_walk(  
 $badBoyLocal, function ($val, $key) use ($which, $k, $id, $badBoyLocal, $el, &$badBoysServer)  
 {  
 $localTime = new DateTime($badBoyLocal[$key . 'UpdateTime']);  
 $serverTime = new DateTime($el[$key . 'UpdateTime']);  
 if ($localTime > $serverTime)  
 {  
 $badBoysServer[$k][$key] = $badBoyLocal[$key];  
 }  
 }  
 );  
 }  
  
 }  
 );  
 }  
 }  
 if ($badBoysLocal != null)  
 {  
 foreach ($badBoysLocal as $badBoy)  
 {  
 $isThere = false;  
 array\_walk(  
 $badBoysServer, function ($el) use ($badBoy, &$isThere)  
 {  
 if ($el['id'] == $badBoy['id'])  
 {  
 $isThere = $el;  
 }  
 }  
 );  
 if (false == $isThere)  
 {  
 $badBoysServer[] = $badBoy;  
 }  
 }  
 }  
}  
file\_put\_contents("data.txt", json\_encode(array("Array" => $badBoysServer, "Deleted" => array())));  
echo json\_encode(array("Array" => $badBoysServer, "Deleted" => array()));

Класс BadBoy отвечает за преступника и хранит все данные в полях, а так же сохраняет время изменения каждого поля в соответствущем поле при изменения свойств.

public class BadBoy

{

public DateTime \_birthDate;

public List<string> \_distinguishingTraits;

public List<string> \_languages;

public string \_name, \_surname, \_nickname,\_eyeColor,\_hairColor, \_citizenship;

public DateTime \_nameUpdateTime;

public string \_birthPlace;

public DateTime \_birthPlaceUpdateTime;

public DateTime \_surnameUpdateTime;

public int \_height;

public DateTime \_heightUpdateTime;

public DateTime \_eyeColorUpdateTime;

public DateTime \_hairColorUpdateTime;

public string \_lastLivingPlace;

public DateTime \_lastLivingPlaceUpdateTime;

public string \_lastCase;

public DateTime \_lastCaseUpdateTime;

public string \_profession;

public DateTime \_professionUpdateTime;

public DateTime \_citizenshipUpdateTime;

public DateTime \_distinguishingTraitsUpdateTime;

public DateTime \_languagesUpdateTime;

public DateTime \_birthDateUpdateTime;

public DateTime \_nicknameUpdateTime;

public List<int> \_accomplices;

public int id;

public DateTime \_accomplicesUpdateTime;

public List<Crimes> \_Crimes;

[JsonIgnore]

public string Name

{

get { return \_name; }

set

{

\_name = value;

\_nameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Surname

{

get { return \_surname; }

set

{

\_surname = value;

\_surnameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Nickname

{

get { return \_nickname; }

set

{

\_nickname = value;

\_nicknameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Height

{

get { return \_height.ToString(); }

set

{

try

{

\_height = Int32.Parse(value);

}

catch

{

\_height = 0;

}

\_heightUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string EyeColor

{

get { return \_eyeColor; }

set

{

\_eyeColor = value;

\_eyeColorUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string HairColor

{

get { return \_hairColor; }

set

{

\_hairColor = value;

\_hairColorUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Citizenship

{

get { return \_citizenship; }

set

{

\_citizenship = value;

\_citizenshipUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string BirthPlace

{

get { return \_birthPlace; }

set

{

\_birthPlace = value;

\_birthPlaceUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string LastLivingPlace

{

get { return \_lastLivingPlace; }

set

{

\_lastLivingPlace = value;

\_lastLivingPlaceUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string LastCase

{

get { return \_lastCase; }

set

{

\_lastCase = value;

\_lastCaseUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Profession

{

get { return \_profession;}

set

{

\_profession = value;

\_professionUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string BirthDate

{

get { return \_birthDate == DateTime.Now ? "" : \_birthDate.Date.ToString("d"); }

set

{

try

{

\_birthDate = DateTime.Parse(value);

}

catch

{

}

\_birthDateUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string DistinguishingTraits

{

get

{

if (\_distinguishingTraits != null)

return String.Join(",", \_distinguishingTraits.ToArray());

return null;

}

set

{

\_distinguishingTraits = value.Split(',').ToList();

\_distinguishingTraitsUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public List<BadBoy> Accomplices

{

get

{

var toGet = new List<BadBoy>();

if(\_accomplices != null)

foreach (var i in \_accomplices)

{

toGet.Add(G.BadBoys[G.BadBoys.FindIndex(x => x.id == i)]);

}

return toGet;

}

set

{

var toSet = new List<int>();

foreach (var i in value)

{

toSet.Add(i.id);

}

\_accomplices = toSet;

\_accomplicesUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Languages

{

get

{

if (\_languages != null)

return String.Join(",", \_languages.ToArray());

return null;

}

set

{

\_languages = value.Split(',').ToList();

\_languagesUpdateTime = DateTime.Now;

}

}//Properties and fields

public BadBoy()

{

id = G.BadBoys.Count > 0? G.BadBoys.Last().id + 1 : 0;

}

public bool IsFound(string s)

{

PropertyInfo[] properties = typeof(BadBoy).GetProperties();

foreach (var i in properties)

{

if (i.GetValue(this) != null && i.GetValue(this).ToString().ToLower().Contains(s.ToLower()))

return true;

}

return false;

}

public override string ToString()

{

return $"{Name} \"{Nickname}\" {Surname}";

}

}

2.2 Пользовательский интерфейс

Проект содержит 4 окна (рисунок 2.2). «MainWindow.xaml» является главным окном проекта, с помощью которого можно перейти к остальным. В этом окне отображаются данные с действующими преступниками в виде таблицы. Здесь находятся основные кнопки, глобальный поиск. «AddWindow.xaml» - окно, которая открывается при необходимости добавить или найти преступника. В форме есть текстовые поля, куда пользователь должен ввести известную информацию, а также выбрать подельников.. «EditWindow.xaml» - редактирование и просмотр данных об одном преступнике. А также «About.xaml» - форма вывода справки.

2.3 Форматы Данных

Данные загружаются из текстового файла в программу с помощью JSONFormatter. В программе они отображаются в сетке (компонент «DataGrid»). Программа работает с данными обращаясь сначала к списку данных, они записаны в объекты (каждый объект имеет свои характеристики), далее к выбранному преступнику, позволяя удалить его из базы, показывать подельников и редактировать.

Во время работы программы, для сохранения данных в файл используется метод, который записывает информацию о преступниках в текстовой файл.

В программе данные по преступникам находятся в классe «BadBoy.cs», который содержит такие свойства:

Здесь каждая строчка хранит в себе какой-ту отдельную информацию о преступнике. Информация о преступнике загружается с класса «G», где содержется список преступников.

3 ИНСТРУКЦИЯ ПО ПРИМЕНЕНИЮ

В данном разделе находится информация об особенностях программы, ее установке, применению и использованию. Программа демонстрирует работу с данными, их отображением и работу с ними. Программа может быть использована в разных учебных проектах, каталогах, милиции, поисковых службах, больницах и т.д.

3.1 Инструкция по Установке

Для установки программы необходимо запустить файл «InterpolWPF.sln», и скомпилировать.

Чтобы начать работать с программой переместите из папки InterpolWPF на свой компьютер само приложение InterpolWPF.exe. Запустите Course.exe. У вас на компьютере должен быть установлен Microsoft. NET Framework 4.5.

После запуска программы на экране появится главное окно программы.

3.2 Инструкция Пользователя

После запуска программы перед вами откроется главное окно программы(рисунок 3.1).



Рисунок 3.1 – Главное окно программы.

В главном окне программы содержатся все необходимые элементы управления. Кнопки Add и Find вызывают одно и то же окно, но с соответствующем ему названием и результатом работы.



Рисунок 3.2 – Окна Add и Find.

Два нижних поля отвечают за выбор преступлений из списка иеарархий преступлений и выбор подельников из уже добавленных в базу преступников, которая на данной стадии окончательно не реализована.

Удаление и редактирование вызываются из соответствующих кнопок контекстного меню при клике по строке таблицы - View/Edit и Delete.



Рисунок 3.2 – Окно Edit.

ВЫВОДЫ

В процессе разработки данной программы, я усвоил на практике знания, полученные на лекциях данного курса. Программа является полностью работоспособной для использования ее в различных организациях, где необходимо вести/знать информацию о людях. Планируется добавить такие функции как «Загрузка информации с Интернета», «Раскраска опасных, менее опасных преступников». Написание программы на языке C# в среде MVS, сводится к легкому кодированию, но подобные программы ограничивают среду применения, так как MVS позволяет использовать программу только в операционной системы Microsoft Windows. Также был усвоен такой навык как «Культура кодирования». Именно благодаря ему код программы становиться восприимчивый к быстрому пониманию и не нуждается в комментариях.

ПЕРЕЧЕНЬ ССЫЛОК

1. Бондарев, В.М. Объектно-ориентированное программирование на С#. Учебное пособие [Текст] / В. М. Бондарев. – Харьков : СМИТ, 2009. – 224 с.

2. Шилдт, Г. C# 3.0: Руководство для начинающих [Текст] : пер. з англ. – М.: Издательский дом «Вильямс», 2009. – 688 с.

3. Шилдт, Г. Полный справочник по C# [Текст] : пер. з англ. – М.: Издательский дом «Вильямс», 2004. – 752 с.

4. Троельсен, Э. C# и платформа .NET. Библиотека программиста [Текст] : пер. з англ. – СПб.: Питер, 2005. – 796 с.

5. Троельсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 [Текст] : пер. з англ. – СПб.: Питер, 2009. – 1311 с.

Приложение А

Код программы

Класс «BadBoy.cs»

namespace Interpol.Models

{

using System;

[Serializable()]

public class Human

{

public Guid Id { get; private set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Patronymic { get; set; }

public uint Age { get; set; }

public DateTime BirthDate { get; set; }

public string Profession { get; set; }

public string FullName

{

get { return string.Format("{0} {1} {2}", FirstName, LastName, Patronymic); }

}

public Human()

{

this.Id = Guid.NewGuid();

}

}

}

Класс «Criminal.cs»

public class BadBoy

{

public DateTime \_birthDate;

public List<string> \_distinguishingTraits;

public List<string> \_languages;

public string \_name, \_surname, \_nickname,\_eyeColor,\_hairColor, \_citizenship;

public DateTime \_nameUpdateTime;

public string \_birthPlace;

public DateTime \_birthPlaceUpdateTime;

public DateTime \_surnameUpdateTime;

public int \_height;

public DateTime \_heightUpdateTime;

public DateTime \_eyeColorUpdateTime;

public DateTime \_hairColorUpdateTime;

public string \_lastLivingPlace;

public DateTime \_lastLivingPlaceUpdateTime;

public string \_lastCase;

public DateTime \_lastCaseUpdateTime;

public string \_profession;

public DateTime \_professionUpdateTime;

public DateTime \_citizenshipUpdateTime;

public DateTime \_distinguishingTraitsUpdateTime;

public DateTime \_languagesUpdateTime;

public DateTime \_birthDateUpdateTime;

public DateTime \_nicknameUpdateTime;

public List<int> \_accomplices;

public int id;

public DateTime \_accomplicesUpdateTime;

public List<Crimes> \_Crimes;

[JsonIgnore]

public string Name

{

get { return \_name; }

set

{

\_name = value;

\_nameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Surname

{

get { return \_surname; }

set

{

\_surname = value;

\_surnameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Nickname

{

get { return \_nickname; }

set

{

\_nickname = value;

\_nicknameUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Height

{

get { return \_height.ToString(); }

set

{

try

{

\_height = Int32.Parse(value);

}

catch

{

\_height = 0;

}

\_heightUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string EyeColor

{

get { return \_eyeColor; }

set

{

\_eyeColor = value;

\_eyeColorUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string HairColor

{

get { return \_hairColor; }

set

{

\_hairColor = value;

\_hairColorUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Citizenship

{

get { return \_citizenship; }

set

{

\_citizenship = value;

\_citizenshipUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string BirthPlace

{

get { return \_birthPlace; }

set

{

\_birthPlace = value;

\_birthPlaceUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string LastLivingPlace

{

get { return \_lastLivingPlace; }

set

{

\_lastLivingPlace = value;

\_lastLivingPlaceUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string LastCase

{

get { return \_lastCase; }

set

{

\_lastCase = value;

\_lastCaseUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Profession

{

get { return \_profession;}

set

{

\_profession = value;

\_professionUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string BirthDate

{

get { return \_birthDate == DateTime.Now ? "" : \_birthDate.Date.ToString("d"); }

set

{

try

{

\_birthDate = DateTime.Parse(value);

}

catch

{

}

\_birthDateUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string DistinguishingTraits

{

get

{

if (\_distinguishingTraits != null)

return String.Join(",", \_distinguishingTraits.ToArray());

return null;

}

set

{

\_distinguishingTraits = value.Split(',').ToList();

\_distinguishingTraitsUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public List<BadBoy> Accomplices

{

get

{

var toGet = new List<BadBoy>();

if(\_accomplices != null)

foreach (var i in \_accomplices)

{

toGet.Add(G.BadBoys[G.BadBoys.FindIndex(x => x.id == i)]);

}

return toGet;

}

set

{

var toSet = new List<int>();

foreach (var i in value)

{

toSet.Add(i.id);

}

\_accomplices = toSet;

\_accomplicesUpdateTime = DateTime.Now;

}

}

[JsonIgnore]

public string Languages

{

get

{

if (\_languages != null)

return String.Join(",", \_languages.ToArray());

return null;

}

set

{

\_languages = value.Split(',').ToList();

\_languagesUpdateTime = DateTime.Now;

}

}//Properties and fields

public BadBoy()

{

id = G.BadBoys.Count > 0? G.BadBoys.Last().id + 1 : 0;

}

public bool IsFound(string s)

{

PropertyInfo[] properties = typeof(BadBoy).GetProperties();

foreach (var i in properties)

{

if (i.GetValue(this) != null && i.GetValue(this).ToString().ToLower().Contains(s.ToLower()))

return true;

}

return false;

}

public override string ToString()

{

return $"{Name} \"{Nickname}\" {Surname}";

}

}

Класс «G»

public class G

{

public static List<BadBoy> BadBoys = new List<BadBoy>();

public static List<int> Deleted = new List<int>();

public static void SaveToFile()

{

string json = JsonConvert.SerializeObject(new Package { Array = G.BadBoys, Deleted = G.Deleted });

using (StreamWriter file = new StreamWriter("data.txt", false))

{

file.Write(json);

}

}

public static void ReadFromFile()

{

try

{

StreamReader file = new StreamReader("data.txt");

}

catch(FileNotFoundException)

{

StreamWriter file = new StreamWriter("data.txt", false);

}

using (StreamReader file = new StreamReader("data.txt"))

{

string json = file.ReadToEnd();

BadBoys = JsonConvert.DeserializeObject<Package>(json).Array;

if(BadBoys == null)

BadBoys = new List<BadBoy>();

}

}

}

public class Package

{

public List<BadBoy> Array;

public List<int> Deleted;

}

Класс «Crimes»

public class Crimes

{

public static Crimes AllCrimes = new Crimes

{

Name = "Crime",

Code = "0000",

Children = new ObservableCollection<Crimes>

{

new Crimes

{

Name = "Acts leading to death or intending to cause death",

Code = "01",

Children = new ObservableCollection<Crimes>

{

new Crimes

{

Name = "Intentional homicide",

Code = "0101"

},

new Crimes

{

Name = "Attempted intentional homicide",

Code = "0102"

},

new Crimes

{

Name = "Non-intentional homicide",

Code = "0103",

Children = new ObservableCollection<Crimes>

{

new Crimes

{

Name = "Non-negligent manslaughter",

Code = "01031"

},

new Crimes

{

Name = "Negligent manslaughter",

Code = "01032",

Children = new ObservableCollection<Crimes>

{

new Crimes

{

Name = "Vehicular homicide",

Code = "010321"

},

new Crimes

{

Name = "Non-vehicular homicide",

Code = "010322"

}

}

}

}

},

new Crimes

{

Name = "Assisting or instigating suicide",

Code = "0104",

Children = new ObservableCollection<Crimes>

{

new Crimes

{

Name = "Assisting suicide",

Code = "01041"

},

new Crimes

{

Name = "Other acts of assisting or instigating suicide ",

Code = "01049"

}

}

},

new Crimes

{

Name = "Euthanasia",

Code = "0105"

},

new Crimes

{

Name = "Illegal feticide",

Code = "0106"

},

new Crimes

{

Name = "Unlawful killing associated with armed conflict",

Code = "0107"

},

new Crimes

{

Name = "Other acts leading to death or intending to cause death",

Code = "0108"

}

}

}

}

};

public string Name { get; set; }

public string Code { get; set; }

public ObservableCollection<Crimes> Children { get; set; }

}

Окно "MainWindow"

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Runtime.Remoting.Channels;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

using Newtonsoft.Json;

namespace InterpolWPF

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

G.ReadFromFile();

dataGrid.ItemsSource = G.BadBoys;

SaveButton.Click += (object sender, RoutedEventArgs e) =>

{

G.SaveToFile();

};

}

private void SearchTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SearchTextBox.Text != "")

{

dataGrid.ItemsSource = new List<BadBoy>(G.BadBoys.Where(boy => boy.IsFound(SearchTextBox.Text)).ToList());

}

else

{

dataGrid.ItemsSource = G.BadBoys;

}

}

private void SearchTextBox\_GotFocus(object sender, RoutedEventArgs e)

{

SearchTextBox.Text = "";

}

private void SearchTextBox\_LostFocus(object sender, RoutedEventArgs e)

{

SearchTextBox.TextChanged -= SearchTextBox\_TextChanged;

SearchTextBox.Text = "Global Search...";

SearchTextBox.TextChanged += SearchTextBox\_TextChanged;

}

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

{

Window editWindow = new AddWindow(false);

editWindow.Show();

}

public void Add(BadBoy toAdd)

{

G.BadBoys.Add(toAdd);

dataGrid.Items.Refresh();

}

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

{

Window findWindow = new AddWindow(true);

findWindow.Show();

}

public void Find(BadBoy toFind)

{

var properties = typeof(BadBoy).GetProperties();

var toDisplay = new List<BadBoy>();

foreach (var i in G.BadBoys)

{

bool isFine = true;

foreach (var propertyInfo in properties)

{

if (propertyInfo.Name != "Languages" && propertyInfo.Name != "DistinguishingTraits" &&

propertyInfo.Name != "BirthDate" && propertyInfo.Name != "Accomplices")

{

if (propertyInfo.GetValue(toFind).ToString() != "" && propertyInfo.GetValue(toFind).ToString() != "0" &&

!propertyInfo.GetValue(i).ToString().ToLower().Contains(propertyInfo.GetValue(toFind).ToString().ToLower()))

{

isFine = false;

break;

}

}

if (toFind.\_distinguishingTraits.Count > 0)

foreach (var trait in toFind.\_distinguishingTraits)

{

if (trait != "" && i.\_distinguishingTraits != null && !i.\_distinguishingTraits.Contains(trait))

{

isFine = false;

break;

}

}

if (toFind.\_languages.Count > 0)

foreach (var language in toFind.\_languages)

{

if (language != "" && i.\_languages != null && !i.\_languages.Contains(language))

{

isFine = false;

break;

}

}

if (toFind.Accomplices.Count > 0)

foreach (var accomplice in toFind.Accomplices)

{

if (!i.Accomplices.Contains(accomplice))

{

isFine = false;

break;

}

}

if (!(toFind.\_birthDate == DateTime.MinValue) && !(i.\_birthDate == DateTime.MinValue) &&

i.BirthDate != toFind.BirthDate)

{

isFine = false;

break;

}

}

if (isFine)

toDisplay.Add(i);

}

dataGrid.ItemsSource = toDisplay;

}

private void Edit\_OnClick(object sender, RoutedEventArgs e)

{

var editWindow = new EditWindow((BadBoy)dataGrid.SelectedItem);

editWindow.Show();

}

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

{

string url = "<http://sync.php/sync.php>";

HttpWebRequest request = (HttpWebRequest)

WebRequest.Create(url); request.KeepAlive = false;

request.ProtocolVersion = HttpVersion.Version10;

request.Method = "POST";

var json = JsonConvert.SerializeObject(new Package{Array = G.BadBoys, Deleted = G.Deleted});

byte[] postBytes = Encoding.UTF8.GetBytes(json);

request.ContentType = "application/json; charset=UTF-8";

request.Accept = "application/json";

request.ContentLength = postBytes.Length;

Stream requestStream = request.GetRequestStream();

requestStream.Write(postBytes, 0, postBytes.Length);

requestStream.Close();

HttpWebResponse response = (HttpWebResponse)request.GetResponse();

string result;

using (StreamReader rdr = new StreamReader(response.GetResponseStream()))

{

result = rdr.ReadToEnd();

}

G.BadBoys = JsonConvert.DeserializeObject<Package>(result).Array;

G.Deleted = new List<int>();

this.dataGrid.ItemsSource = null;

this.dataGrid.ItemsSource = G.BadBoys;

G.SaveToFile();

}

private void Delete\_OnClick(object sender, RoutedEventArgs e)

{

var toDelete = (BadBoy) (dataGrid.SelectedItem);

if (G.Deleted != null)

G.Deleted.Add(toDelete.id);

G.BadBoys.RemoveAt(G.BadBoys.IndexOf(toDelete));

dataGrid.ItemsSource = null;

dataGrid.ItemsSource = G.BadBoys;

}

/\*private void dataGrid\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

var toDelete = (BadBoy)(dataGrid.SelectedItem);

G.Deleted.Add(toDelete.id);

G.BadBoys.RemoveAt(G.BadBoys.IndexOf(toDelete));

dataGrid.ItemsSource = null;

dataGrid.ItemsSource = G.BadBoys;

}

}\*/

private void MenuItem\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

var toDelete = (BadBoy)(dataGrid.SelectedItem);

G.Deleted.Add(toDelete.id);

G.BadBoys.RemoveAt(G.BadBoys.IndexOf(toDelete));

dataGrid.ItemsSource = null;

dataGrid.ItemsSource = G.BadBoys;

}

}

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

{

var aboutWindow = new About();

aboutWindow.Show();

}

}

}

Окно "AddWindow"

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace InterpolWPF

{

/// <summary>

/// Логика взаимодействия для AddWindow.xaml

/// </summary>

public partial class AddWindow : Window

{

private bool dateChanged = false;

public AddWindow(bool isSearch)

{

InitializeComponent();

if (isSearch)

{

SaveButton.Content = "Find";

SaveButton.Click -= SaveButton\_Click;

SaveButton.Click += FindButton\_Click;

this.Title = "Find...";

}

AccomplicesListBox.ItemsSource = G.BadBoys;

treeView.Items.Add(Crimes.AllCrimes);

}

private void FindButton\_Click(object sender, RoutedEventArgs routedEventArgs)

{

var toFind = ExtractBadBoy();

((MainWindow)App.Current.MainWindow).Find(toFind);

Close();

}

public void TextBox\_GotFocus(object sender, RoutedEventArgs e)

{

var target = (TextBox)e.Source;

if (!Equals(target.Foreground, Brushes.Black))

{

var brush = target.Foreground;

target.Foreground = Brushes.Black;

string text = target.Text;

target.Text = "";

target.LostFocus += (o, args) =>

{

if (target.Text == "")

{

target.Foreground = brush;

target.Text = text;

}

};

}

}

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

{

var toAdd = ExtractBadBoy();

MainWindow mainWindow = (MainWindow)App.Current.MainWindow;

mainWindow.Add(toAdd);

G.SaveToFile();

Close();

mainWindow.dataGrid.Items.Refresh();

}

public BadBoy ExtractBadBoy()

{

var badBoy = new BadBoy

{

Name = Equals(NameTextBox.Foreground, Brushes.Black) ? NameTextBox.Text : "",

Surname = Equals(SurnameTextBox.Foreground, Brushes.Black) ? SurnameTextBox.Text : "",

Nickname = Equals(NicknameTextBox.Foreground, Brushes.Black) ? NicknameTextBox.Text : "",

Height = Equals(HeightTextBox.Foreground, Brushes.Black) ? HeightTextBox.Text : "0",

EyeColor = Equals(EyeColorTextBox.Foreground, Brushes.Black) ? EyeColorTextBox.Text : "",

BirthDate = dateChanged ? BirthDatePicker.SelectedDate.ToString() : DateTime.Now.ToString(),

Citizenship = Equals(CitizenshipTextBox.Foreground, Brushes.Black) ? CitizenshipTextBox.Text : "",

BirthPlace = Equals(BirthPlaceTextBox.Foreground, Brushes.Black) ? BirthPlaceTextBox.Text : "",

DistinguishingTraits = Equals(DistinguishingTraitsTextBox.Foreground, Brushes.Black) ? DistinguishingTraitsTextBox.Text : "",

Languages = Equals(LanguagesTextBox.Foreground, Brushes.Black) ? LanguagesTextBox.Text : "",

HairColor = Equals(HairColorTextBox.Foreground, Brushes.Black) ? HairColorTextBox.Text : "",

LastLivingPlace = Equals(LastLivingPlaceTextBox.Foreground, Brushes.Black) ? LastLivingPlaceTextBox.Text : "",

Profession = Equals(ProfessionTextBox.Foreground, Brushes.Black) ? ProfessionTextBox.Text : "",

LastCase = Equals(LastCaseTextBox.Foreground, Brushes.Black) ? LastCaseTextBox.Text : ""

};

var accomplices = new List<BadBoy>();

foreach (var i in AccomplicesListBox.SelectedItems)

{

accomplices.Add((BadBoy) i);

}

badBoy.Accomplices = accomplices;

return badBoy;

}

private void BirthDatePicker\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

dateChanged = true;

}

private void MenuItem\_Add(object sender, RoutedEventArgs e)

{

}

}

}

Окно "EditWindow"

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace InterpolWPF

{

/// <summary>

/// Логика взаимодействия для EditWindow.xaml

/// </summary>

public partial class EditWindow : Window

{

private BadBoy current;

public EditWindow(BadBoy badBoy)

{

InitializeComponent();

current = badBoy;

var properties = typeof(BadBoy).GetProperties();

foreach (var property in properties)

{

if (property.Name != "Accomplices" && property.Name != "BirthDate")

{

var textBox =

(TextBox)

(this.GetType()

.GetField(property.Name + "TextBox", BindingFlags.NonPublic | BindingFlags.Instance)

.GetValue(this));

textBox.Text = property.GetValue(badBoy).ToString();

}

}

AccomplicesListBox.ItemsSource = G.BadBoys;

foreach (var accomplice in badBoy.Accomplices)

{

AccomplicesListBox.SelectedItems.Add(accomplice);

}

BirthDatePicker.DisplayDate = badBoy.\_birthDate;

BirthDatePicker.Text = BirthDatePicker.DisplayDate.ToString();

}

private void TextBox\_GotFocus(object sender, RoutedEventArgs e)

{

/\*var target = (TextBox)e.Source;

if (!Equals(target.Foreground, Brushes.Black))

{

var brush = target.Foreground;

target.Foreground = Brushes.Black;

string text = target.Text;

target.Text = "";

target.LostFocus += (o, args) =>

{

if (target.Text == "")

{

target.Foreground = brush;

target.Text = text;

}

};

}\*/

}

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

{

ExtractBadBoy();

MainWindow mainWindow = (MainWindow)App.Current.MainWindow;

mainWindow.dataGrid.Items.Refresh();

G.SaveToFile();

Close();

}

public void ExtractBadBoy()

{

{

current.Name = Equals(NameTextBox.Foreground, Brushes.Black) ? NameTextBox.Text : "";

current.Surname = Equals(SurnameTextBox.Foreground, Brushes.Black) ? SurnameTextBox.Text : "";

current.Nickname = Equals(NicknameTextBox.Foreground, Brushes.Black) ? NicknameTextBox.Text : "";

current.Height = Equals(HeightTextBox.Foreground, Brushes.Black) ? HeightTextBox.Text : "0";

current.EyeColor = Equals(EyeColorTextBox.Foreground, Brushes.Black) ? EyeColorTextBox.Text : "";

current.BirthDate = BirthDatePicker.DisplayDate.ToString();

current.Citizenship = Equals(CitizenshipTextBox.Foreground, Brushes.Black)

? CitizenshipTextBox.Text

: "";

current.BirthPlace = Equals(BirthPlaceTextBox.Foreground, Brushes.Black) ? BirthPlaceTextBox.Text : "";

current.DistinguishingTraits =

Equals(DistinguishingTraitsTextBox.Foreground, Brushes.Black)

? DistinguishingTraitsTextBox.Text

: "";

current.Languages = Equals(LanguagesTextBox.Foreground, Brushes.Black) ? LanguagesTextBox.Text : "";

current.HairColor = Equals(HairColorTextBox.Foreground, Brushes.Black) ? HairColorTextBox.Text : "";

current.LastLivingPlace = Equals(LastLivingPlaceTextBox.Foreground, Brushes.Black)

? LastLivingPlaceTextBox.Text

: "";

current.Profession = Equals(ProfessionTextBox.Foreground, Brushes.Black) ? ProfessionTextBox.Text : "";

current.LastCase = Equals(LastCaseTextBox.Foreground, Brushes.Black) ? LastCaseTextBox.Text : "";

var accomplices = new List<BadBoy>();

foreach (var i in AccomplicesListBox.SelectedItems)

{

accomplices.Add((BadBoy) i);

}

current.Accomplices = accomplices;

}

}

}

}

Серверный код

<?php  
/\*\*  
 \* Created by PhpStorm.  
 \* User: Monyk  
 \* Date: 28.05.2016  
 \* Time: 1:35  
 \*/  
  
$data = $HTTP\_RAW\_POST\_DATA;  
$dataLocal = json\_decode($data, true);  
$badBoysLocal = $dataLocal['Array'];  
$deleted = $dataLocal['Deleted']; //$dataLocal->Deleted;  
$badBoysServer = json\_decode(file\_get\_contents("data.txt"), true)['Array'];  
if ($badBoysServer == null)  
{  
 $badBoysServer = $badBoysLocal;  
}  
else  
{  
 if ($deleted != null)  
 {  
 foreach ($deleted as $value)  
 {  
 $id = $value;  
 array\_walk(  
 $badBoysServer, function ($el, $key) use ($id, &$badBoysServer)  
 {  
 if ($el['id'] == $id)  
 {  
 array\_splice($badBoysServer, $key, 1);  
 }  
 }  
 );  
 }  
 }  
 if ($badBoysLocal != null)  
 {  
 foreach ($badBoysLocal as $which => $badBoyLocal)  
 {  
 $id = $badBoyLocal['id'];  
 array\_walk(  
 $badBoysServer, function ($el, $k) use ($which, $id, $badBoysLocal, $badBoyLocal, &$badBoysServer)  
 {  
 if ($el['id'] == $id)  
 //foreach ($badBoyLocal as $key => $property)  
 //if(new DateTime($badBoyLocal[$key.'UpdateTime']) > new DateTime($el[$key.'UpdateTime']))  
 {  
 array\_walk(  
 $badBoyLocal, function ($val, $key) use ($which, $k, $id, $badBoyLocal, $el, &$badBoysServer)  
 {  
 $localTime = new DateTime($badBoyLocal[$key . 'UpdateTime']);  
 $serverTime = new DateTime($el[$key . 'UpdateTime']);  
 if ($localTime > $serverTime)  
 {  
 $badBoysServer[$k][$key] = $badBoyLocal[$key];  
 }  
 }  
 );  
 }  
  
 }  
 );  
 }  
 }  
 if ($badBoysLocal != null)  
 {  
 foreach ($badBoysLocal as $badBoy)  
 {  
 $isThere = false;  
 array\_walk(  
 $badBoysServer, function ($el) use ($badBoy, &$isThere)  
 {  
 if ($el['id'] == $badBoy['id'])  
 {  
 $isThere = $el;  
 }  
 }  
 );  
 if (false == $isThere)  
 {  
 $badBoysServer[] = $badBoy;  
 }  
 }  
 }  
}  
file\_put\_contents("data.txt", json\_encode(array("Array" => $badBoysServer, "Deleted" => array())));  
echo json\_encode(array("Array" => $badBoysServer, "Deleted" => array()));