**Национальный исследовательский университет**

**«Высшая школа экономики»**

**Факультет компьютерных наук**

**Департамент Программной инженерии**

Контрольное домашнее задание

по дисциплине

«**Программирование**»

Тема работы: **«Программа для чтения, редактирования и записи CSV файлов заданного формата»**

**Вариант 5**

Выполнил: студент группы БПИ184 (1)

Фомин Сергей Дмитриевич

Тел. +7 (988) 627-19-92

E-mail адрес: sofmin@edu.hse.ru

Преподаватель: Горденко Мария Константиновна,

Преподаватель Департамента

Программной инженерии

**Москва, 2019 год. Модуль 3**

Оглавление

[1. Условие задачи 3](#_Toc3927693)

[2. Функции разрабатываемого приложения 5](#_Toc3927694)

[2.1. Варианты использования 5](#_Toc3927695)

[2.2. Описание интерфейса пользователя 5](#_Toc3927696)

[2.2.1. Главное окно программы 5](#_Toc3927697)

[3. Структура приложения 6](#_Toc3927698)

[3.1. Диаграмма классов 6](#_Toc3927699)

[3.2. Описание методов, полей, свойств и классов 7](#_Toc3927700)

[3.2.1. MainWindow 7](#_Toc3927701)

[4. Распределение исходного кода по файлам проекта 7](#_Toc3927702)

[4.1. CSVMaker.cs 7](#_Toc3927703)

[4.2. School.cs 7](#_Toc3927704)

[4.3. SchoolForDataGrid.cs 7](#_Toc3927705)

[5. Контрольный пример и описание результатов 7](#_Toc3927706)

[6. Сообщения пользователю 8](#_Toc3927707)

[7. Текст (код) программы 9](#_Toc3927708)

[8. Список использованной литературы 9](#_Toc3927709)

# Условие задачи

Программа Контрольного домашнего задания представляет собой WindowsForms

приложение и предназначена для просмотра, обработки и сохранения результатов

обработки данных из файла.

Средняя образовательная школа версия 1.1 от 31.10.2013.csv

1. Требования к основным классам приложения

1.1. Основная информация о Средних образовательных школах хранится в объектах

класса Школа. Набор полей класса задаётся полями файла Средняя

образовательная школа версия 1.1 от 31.10.2013.csv, кроме полей, содержащих

информацию об административном округе и районе, представленных полем типа

Расположение. Класс Школа находится в отношении агрегации с классом

Расположение. Один из методов класса возвращает данной школы признак true,

если она государственная и false – в противном случае.

1.2. Класс Расположение представляет информацию об административном

расположении Школы, заданное полями CSV-файла: AdmArea, District.

1.3. Дополнительные классы, необходимые для решения задачи.

2. Приложение должно поддерживать следующие функции

2.1. Открыть CSV-файл (\*.csv) с исходными данными и проверить корректность

данных в нём.

2.2. Загрузить данные из CSV-файла в объекты классов Школа, Расположение (если

объект Расположение с данными об определённом районе существует, то он

является общим для всех объектов Школа этого района) и др.

2.3. Отобразить данные из объектов в оконной форме.

2.4. Создать новую запись о Школе.

2.5. Удалить уже существующую запись о Школе.

2.6. Отредактировать существующую запись о Школе.

2.7. Отсортировать данные по полям: rayon, okrug

2.8. Отсортировать данные по количеству государственных/негосударственных школ в

данном округе. Данные для сортировки вводятся пользователем

2.9. Отфильтровать данные по полям: form\_of\_incorporation, submission

2.10. Сохранять результаты редактирований, сортировок и фильтраций в CSVфайл. Режимы сохранения в файл: создание нового файла, замена содержимого

уже существующего файла, добавление сохраняемых данных к содержимому

существующего файла.

3. Требования к интерфейсу

3.1. При управлении файлом (загрузка, сохранение) использовать OpenFileDialog и

SaveFileDialog.

3.2. Для отображения данных использовать сетку DataGridView. Количество

отображаемых в сетке элементов (N) выбирается пользователем, N > 1 и не

превышает количества записей в файле Средняя образовательная школа версия

1.1 от 31.10.2013.csv.

4. Требования к устойчивости приложения

4.1. В случае ошибок открывания/сохранения файла или некорректных данных

программа должна выводить сообщение.

4.2. Аварийные ситуации должны обрабатываться, пользователю должны выводиться

информативные сообщения

Общие для всех вариантов КДЗ функциональные требования:

1. Приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие стандартные операции со списком объектов:

1.1. Создание списка путем ввода данных пользователем

1.2. добавление в список нового объекта

1.3. удаление объекта из списка

1.4. вывод сведений об объектах заданного пользователем типа из списка на форму

1.5. сохранение списка объектов в указанном пользователем текстовом файле в режиме:

1.6. сохранения в файле текущего списка объектов

1.7. добавления в файл текущего списка объектов

1.8. построение списка объектов по данным, прочитанным из файла

2. Список объектов может быть реализован в виде массива или какой-либо коллекции, например, списка. Количество создаваемых объектов заранее не известно. Порядок следования объектов разных типов (если такие есть) при создании списка произвольный и определяется пользователем.

3. Приложение обязательно должно корректно открывать и позволять модифицировать, созданные с его помощью файлы с данными.

4. Исходные файлы с данными могут располагаться в любой папке компьютера.

5. Приложение должно корректно работать с файлами, в пути к которым содержатся символы национальных алфавитов, знаки препинания, # и проч., допустимые для путей к файлам в ОС Windows символы.

Общие требования к интерфейсу программы

1. Приложение должно представлять собой оконное Windows приложение.

2. При добавлении данных в ИСС исходные данные пользователь вводит с помощью экранных форм, содержащих поля для текстового ввода или списки значений.

3. При отсутствии явных ограничений в варианте задания, результаты отображаются с помощью экранных форм для вывода текста или элементов типа «сетка данных».

4. Сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях отображать в всплывающих окнах типа окон сообщений.

5. В приложении в обязательно должны быть реализованы следующие элементы управления: меню, инструментальная панель, всплывающие подсказки.

6. Остальные элементы управления используются по усмотрению разработчика.

Ограничения

1. При выполнении КДЗ требуется соблюдать корпоративные стандарты НИУ ВШЭ. В дисциплине «Программирование» стандартом в текущем учебном году является применение Microsoft VS 2017 (не выше).

2. В программной реализации не использовать вспомогательные компоненты и сторонние библиотеки, не входящие в стандартную библиотеку.

3. Не использовать массивы типа object[].

4. Не применять СУБД и базы данных.

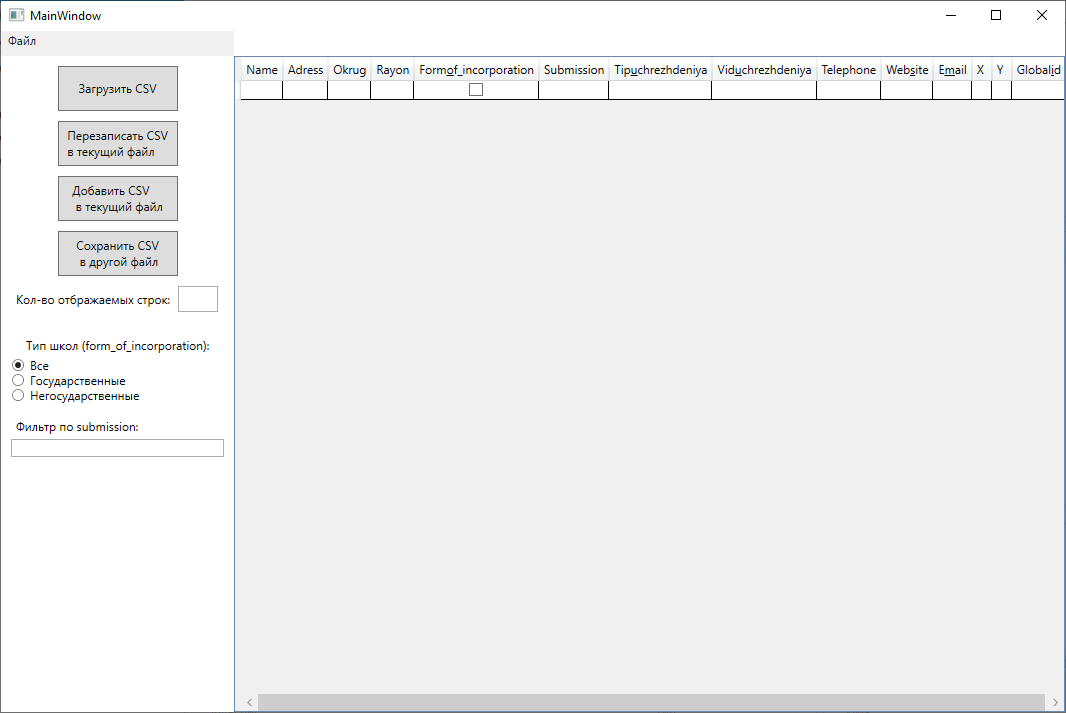
# Функции разрабатываемого приложения

## Варианты использования

Программа используется для обработки csv файлов, содержащих информацию о школах.

## Описание интерфейса пользователя

### Главное окно программы



4

3

2

1

Рисунок 1 Главное окно программы

1 – файл

2 – загрузка и сохранение файлов

3 – фильтры

4 – сетка, отображающее содержимое csv файла

# Структура приложения

## Диаграмма классов

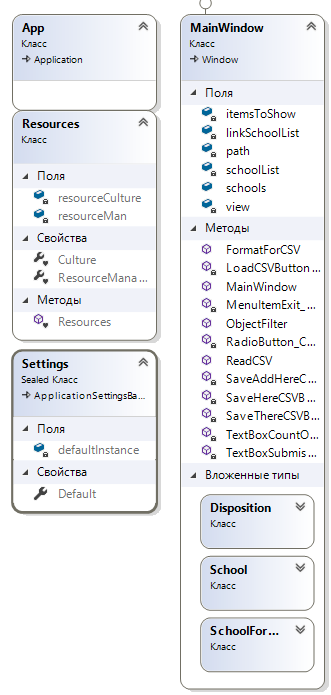


Рисунок 2 Диаграмма классов

## Описание методов, полей, свойств и классов

### MainWindow

#### Поля

itemsToShow – количество элементов для отображения

path – путь к текущему файлу

schoolList – List, хранящий все элементы

view – штука для фильтров

#### Методы

FormatForCSV – отформатировать данные для CSV

LoadCSVButton – загрузить CSV файл

MainWindow – MainWindow

MenuItemExit – выход из программы

ObjectFilter – фильтр

RadioButton\_Click – клик по кнопке выбора типа школ

ReadCSV – прочитать CSV файл

SaveAddHereClick – добавить в текущий файл

SaveHereCSVButton – сохранить в текущий файл

SaveThereCSVButton – сохранить в другой файл

TextBoxCount – количество отображаемых объектов

TextBoxSubmission – фильтр по submission

# Распределение исходного кода по файлам проекта

## CSVMaker.cs

Обрабатывает CSV файл.

## School.cs

Класс содержит в себе поля из файла

## SchoolForDataGrid.cs

Преобразует данные для отображения в DataGrid

# Контрольный пример и описание результатов

На рисунке 3 показана работа программы. Пользователь открыл файл, содержащий в себе данные о школах, добавил запись и сохранил всё в другой файл.

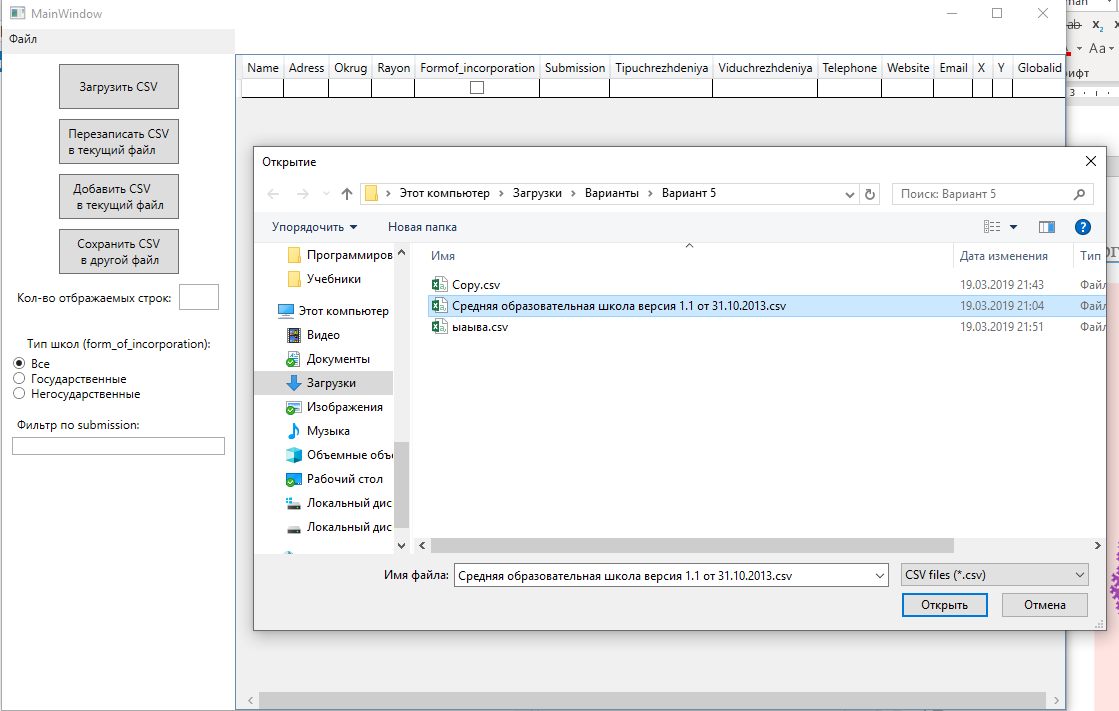


Рисунок 3 Пример работы программы

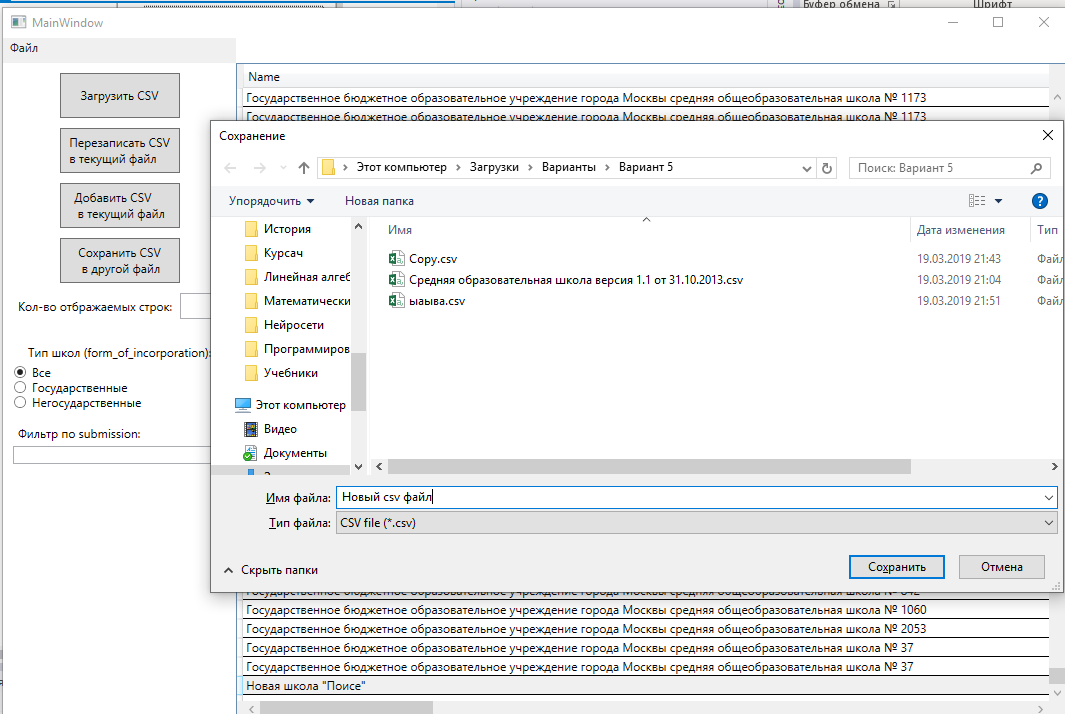


Рисунок 4 Сохранённое изображение фрактала

# Сообщения пользователю

Сообщения о том, что файл загрузился и сохранился нормально

# Текст (код) программы

## MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.IO;

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.Navigation;

using System.Windows.Shapes;

using Microsoft.VisualBasic.FileIO;

using Microsoft.Win32;

namespace SchoolListEditor

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

ObservableCollection<SchoolForDataGrid> schoolList = new ObservableCollection<SchoolForDataGrid>();

List<School> linkSchoolList = new List<School>();

int itemsToShow = int.MaxValue;

ICollectionView view;

string path = "";

public List<School> schools = null;

/// <summary>

/// Собственно MainWindow

/// </summary>

public MainWindow()

{

InitializeComponent();

mainDataGrid.ItemsSource = schoolList;

}

/// <summary>

/// Загрузить CSV файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "CSV files (\*.csv)|\*.csv";

if (openFileDialog.ShowDialog() == true)

{

path = openFileDialog.FileName;

schools = new List<School>();

using (var fs = openFileDialog.OpenFile())

ReadCSV((FileStream)fs, schools);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Изменить количество отображаемых элементов

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

itemsToShow = -1;

int.TryParse(((TextBox)sender).Text, out itemsToShow);

if (itemsToShow < 1) itemsToShow = int.MaxValue;

view?.Refresh();

}

/// <summary>

/// Отобразить выбранный тип школ

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void RadioButton\_Checked(object sender, RoutedEventArgs e)

{

view?.Refresh();

}

/// <summary>

/// Отфильтровать по submission

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

view?.Refresh();

}

/// <summary>

/// Сохранить в текущий CSV файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

if (path == "")

{

MessageBox.Show("Файл не найден, сохраните в отдельный файл");

return;

}

using (FileStream fs = File.Open(path, FileMode.Create, FileAccess.Write))

{

using (StreamWriter sw = new StreamWriter(fs))

{

int cnt = 0;

sw.WriteLine("ROWNUM;name;adress;okrug;rayon;form\_of\_incorporation;submission;" +

"tip\_uchrezhdeniya;vid\_uchrezhdeniya;telephone;web\_site;e\_mail;X;Y;global\_id");

foreach (SchoolForDataGrid school in schoolList)

{

string s = "";

s += ++cnt + ";";

s += FormatForCSV(school.Name);

s += FormatForCSV(school.Adress);

s += FormatForCSV(school.Okrug);

s += FormatForCSV(school.Rayon);

if (school.Form\_of\_incorporation) s += ("Государственное;");

else s += ("Негосударственное;");

s += FormatForCSV(school.Submission);

s += FormatForCSV(school.Tip\_uchrezhdeniya);

s += FormatForCSV(school.Vid\_uchrezhdeniya);

s += FormatForCSV(school.Telephone);

s += FormatForCSV(school.Web\_site);

s += FormatForCSV(school.E\_mail);

s += FormatForCSV(school.X);

s += FormatForCSV(school.Y);

s += FormatForCSV(school.Global\_id);

s = s.Substring(0, s.Length - 1);

int t = 0;

foreach (char i in s)

if (i == ';') ++t;

sw.WriteLine(s);

}

}

}

MessageBox.Show("Успешно сохранено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Отформатировать для сохранения в CSV

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string FormatForCSV(string s) => "\"" + s?.Replace("\"", "\"\"") + "\";";

/// <summary>

/// Добавить в текущий файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

List<School> schools = new List<School>();

using (FileStream fs = File.Open(path, FileMode.Open, FileAccess.Read))

using (StreamReader reader = new StreamReader(fs))

{

using (TextFieldParser parser = new TextFieldParser(reader))

{

parser.TextFieldType = FieldType.Delimited;

parser.SetDelimiters(";");

string[] fields = parser.ReadFields();

// TODO: check zagolovok

while (!parser.EndOfData)

{

//Process row

fields = parser.ReadFields();

School school = new School();

school.ROWNUM = int.Parse(fields[0]);

school.Name = fields[1];

school.Location.Adress = fields[2];

school.Location.Okrug = fields[3];

school.Location.Rayon = fields[4];

school.Form\_of\_incorporation = (fields[5] == "Негосударственное") ? false : true;

school.Submission = fields[6];

school.Tip\_uchrezhdeniya = fields[7];

school.Vid\_uchrezhdeniya = fields[8];

school.Telephone = fields[9];

school.Web\_site = fields[10];

school.E\_mail = fields[11];

school.X = fields[12];

school.Y = fields[13];

school.Global\_id = fields[14];

schools.Add(school);

}

}

}

ObservableCollection<SchoolForDataGrid> fileSchoolList = SchoolForDataGrid.Convert(schools), tmpSchoolList;

using (FileStream fs = File.Open(path, FileMode.Create, FileAccess.Write))

{

using (StreamWriter sw = new StreamWriter(fs, Encoding.Unicode))

{

int cnt = 0;

sw.WriteLine("ROWNUM;name;adress;okrug;rayon;form\_of\_incorporation;submission;" +

"tip\_uchrezhdeniya;vid\_uchrezhdeniya;telephone;web\_site;e\_mail;X;Y;global\_id");

tmpSchoolList = schoolList;

schoolList = fileSchoolList;

foreach (SchoolForDataGrid school in schoolList)

{

string s = "";

s += ++cnt + ";";

s += FormatForCSV(school.Name);

s += FormatForCSV(school.Adress);

s += FormatForCSV(school.Okrug);

s += FormatForCSV(school.Rayon);

if (school.Form\_of\_incorporation) s += ("Государственное;");

else s += ("Негосударственное;");

s += FormatForCSV(school.Submission);

s += FormatForCSV(school.Tip\_uchrezhdeniya);

s += FormatForCSV(school.Vid\_uchrezhdeniya);

s += FormatForCSV(school.Telephone);

s += FormatForCSV(school.Web\_site);

s += FormatForCSV(school.E\_mail);

s += FormatForCSV(school.X);

s += FormatForCSV(school.Y);

s += FormatForCSV(school.Global\_id);

s = s.Substring(0, s.Length);

sw.WriteLine(s);

}

schoolList = tmpSchoolList;

foreach (SchoolForDataGrid school in schoolList)

{

string s = "";

s += ++cnt + ";";

s += FormatForCSV(school.Name);

s += FormatForCSV(school.Adress);

s += FormatForCSV(school.Okrug);

s += FormatForCSV(school.Rayon);

if (school.Form\_of\_incorporation) s += ("Государственное;");

else s += ("Негосударственное;");

s += FormatForCSV(school.Submission);

s += FormatForCSV(school.Tip\_uchrezhdeniya);

s += FormatForCSV(school.Vid\_uchrezhdeniya);

s += FormatForCSV(school.Telephone);

s += FormatForCSV(school.Web\_site);

s += FormatForCSV(school.E\_mail);

s += FormatForCSV(school.X);

s += FormatForCSV(school.Y);

s += FormatForCSV(school.Global\_id);

s = s.Substring(0, s.Length);

sw.WriteLine(s);

}

}

}

schools = new List<School>();

using (FileStream fs = File.Open(path, FileMode.Open, FileAccess.Read))

ReadCSV(fs, schools);

view.Refresh();

MessageBox.Show("Успешно сохранено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Сохранить в другой файл

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "CSV file (\*.csv)|\*.csv";

if (saveFileDialog.ShowDialog() == true)

{

using (FileStream fs = File.Open(saveFileDialog.FileName, FileMode.Create, FileAccess.Write))

{

using (StreamWriter sw = new StreamWriter(fs))

{

int cnt = 0;

sw.WriteLine("ROWNUM;name;adress;okrug;rayon;form\_of\_incorporation;submission;" +

"tip\_uchrezhdeniya;vid\_uchrezhdeniya;telephone;web\_site;e\_mail;X;Y;global\_id");

foreach (SchoolForDataGrid school in schoolList)

{

string s = "";

s += ++cnt + ";";

s += FormatForCSV(school.Name);

s += FormatForCSV(school.Adress);

s += FormatForCSV(school.Okrug);

s += FormatForCSV(school.Rayon);

if (school.Form\_of\_incorporation) s += ("Государственное;");

else s += ("Негосударственное;");

s += FormatForCSV(school.Submission);

s += FormatForCSV(school.Tip\_uchrezhdeniya);

s += FormatForCSV(school.Vid\_uchrezhdeniya);

s += FormatForCSV(school.Telephone);

s += FormatForCSV(school.Web\_site);

s += FormatForCSV(school.E\_mail);

s += FormatForCSV(school.X);

s += FormatForCSV(school.Y);

s += FormatForCSV(school.Global\_id);

s = s.Substring(0, s.Length - 1);

int t = 0;

foreach (char i in s)

if (i == ';') ++t;

sw.WriteLine(s);

}

}

}

}

} catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Выход

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.Close();

}

}

}

## CSVMaker.cs

using Microsoft.VisualBasic.FileIO;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Data;

namespace SchoolListEditor

{

public partial class MainWindow

{

/// <summary>

/// Считываем CSV файл

/// </summary>

/// <param name="openFileDialog"></param>

/// <param name="schools"></param>

public void ReadCSV(FileStream fileStream, List<School> schools)

{

try

{

using (StreamReader reader = new StreamReader(fileStream))

{

using (TextFieldParser parser = new TextFieldParser(reader))

{

parser.TextFieldType = FieldType.Delimited;

parser.SetDelimiters(";");

string[] fields = parser.ReadFields();

// TODO: check zagolovok

while (!parser.EndOfData)

{

//Process row

fields = parser.ReadFields();

School school = new School();

school.ROWNUM = int.Parse(fields[0]);

school.Name = fields[1];

school.Location.Adress = fields[2];

school.Location.Okrug = fields[3];

school.Location.Rayon = fields[4];

school.Form\_of\_incorporation = (fields[5] == "Негосударственное") ? false : true;

school.Submission = fields[6];

school.Tip\_uchrezhdeniya = fields[7];

school.Vid\_uchrezhdeniya = fields[8];

school.Telephone = fields[9];

school.Web\_site = fields[10];

school.E\_mail = fields[11];

school.X = fields[12];

school.Y = fields[13];

school.Global\_id = fields[14];

schools.Add(school);

}

}

}

schoolList = SchoolForDataGrid.Convert(schools);

view = CollectionViewSource.GetDefaultView(schoolList);

view.Filter = ObjectFilter;

mainDataGrid.ItemsSource = schoolList;

MessageBox.Show("Успешно загружено");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Фильтр

/// </summary>

/// <param name="obj"></param>

/// <returns></returns>

public bool ObjectFilter(object obj)

{

SchoolForDataGrid school = (SchoolForDataGrid)obj;

bool value = schoolList.IndexOf(school) < itemsToShow;

if (radioButtonIncorporationTrue.IsChecked == true && !school.Form\_of\_incorporation) value = false;

if (radioButtonIncorporationFalse.IsChecked == true && school.Form\_of\_incorporation) value = false;

if (textBoxSubmission.Text != "")

if (!school.Submission.Contains(textBoxSubmission.Text)) value = false;

return value;

}

}

}

## School.cs

namespace SchoolListEditor

{

public partial class MainWindow

{

public class School

{

public int ROWNUM { get; set; }

public string Name { get; set; }

public Disposition Location { get; set; } = new Disposition();

public bool Form\_of\_incorporation { get; set; }

public string Submission { get; set; }

public string Tip\_uchrezhdeniya { get; set; }

public string Vid\_uchrezhdeniya { get; set; }

public string Telephone { get; set; }

public string Web\_site { get; set; }

public string E\_mail { get; set; }

public string X { get; set; }

public string Y { get; set; }

public string Global\_id { get; set; }

}

public class Disposition

{

public string Adress { get; set; }

public string Okrug { get; set; }

public string Rayon { get; set; }

}

}

}

## SchoolForDataGrid.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Windows;

namespace SchoolListEditor

{

public partial class MainWindow

{

public class SchoolForDataGrid

{

public string Name { get; set; }

public string Adress { get; set; }

public string Okrug { get; set; }

public string Rayon { get; set; }

public bool Form\_of\_incorporation { get; set; }

public string Submission { get; set; }

public string Tip\_uchrezhdeniya { get; set; }

public string Vid\_uchrezhdeniya { get; set; }

public string Telephone { get; set; }

public string Web\_site { get; set; }

public string E\_mail { get; set; }

public string X { get; set; }

public string Y { get; set; }

public string Global\_id { get; set; }

/// <summary>

/// Отконвертировать для отображения в DataGrid

/// </summary>

/// <param name="schools"></param>

/// <returns></returns>

public static ObservableCollection<SchoolForDataGrid> Convert(List<School> schools)

{

try

{

ObservableCollection<SchoolForDataGrid> result = new ObservableCollection<SchoolForDataGrid>();

foreach (var i in schools)

{

result.Add(new SchoolForDataGrid()

{

Name = i.Name,

Adress = i.Location.Adress,

Okrug = i.Location.Okrug,

Rayon = i.Location.Rayon,

Form\_of\_incorporation = i.Form\_of\_incorporation,

Submission = i.Submission,

Tip\_uchrezhdeniya = i.Tip\_uchrezhdeniya,

Vid\_uchrezhdeniya = i.Vid\_uchrezhdeniya,

Telephone = i.Telephone,

Web\_site = i.Web\_site,

E\_mail = i.E\_mail,

X = i.X,

Y = i.Y,

Global\_id = i.Global\_id

});

}

schools.Clear();

return result;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

}

}

}

# Список использованной литературы

**Metanit.com. Руководство по WPF** [В Интернете]. - https://metanit.com/sharp/wpf/.

**Microsoft Windows Presentation Foundation** [В Интернете]. - https://docs.microsoft.com/en-us/dotnet/framework/wpf/.

**Справочник по C#** [В Интернете]. - https://docs.microsoft.com/ru-ru/previous-versions/618ayhy6(v=vs.120).