**Национальный исследовательский университет «Высшая школа экономики»**

**Факультет компьютерных наук**

Департамент

**Программной инженерии**

***Контрольное домашнее задание   
по дисциплине   
«Программирование»***

Тема работы: CSV файлы

Выполнил: студент группы БПИ181(1)

Степанов Е.В.

тел. +7 910 40 16 982

e-mail адрес: evstepanov@edu.hse.ru

Преподаватель: Береснева Е.Н.

Москва, 2019 год. Модуль 3

**СОДЕРЖАНИЕ**

1. **Условие задачи3**
2. **Функции разрабатываемого приложения4**

2.1 Варианты использования4

2.2 Описание интерфейса4

1. **Структура приложения6**

3.1 Диаграмма классов 6

3.2 Описание классов, их полей и методов6

* 1. Архитектура решения10

1. **Распределение исходного кода по файлам проекта12**
2. **Контрольный пример и описание результатов12**
3. **Приложение А (Исходный код программы)14**
4. **Список литературы77**
5. **Условие задачи**

Вариант 14.

Разработать оконное приложение WPF, позволяющее:

1. Открыть CSV-файл (**\*.csv**) с исходными данными и проверить корректность данных в нём.
2. Загрузить данные из CSV-файла в объекты классов «Выход из тоннеля» с последующим преобразованием в класс «Модель выхода из тоннеля»
3. Отобразить данные из объектов в оконной форме с помощью элемента Data Grid
4. Создать новую запись о выходе из тоннеля
5. Удалить уже существующую запись о Выходе из тоннеля
6. Отредактировать текущую запись о выходе из тоннеля
7. Отсортировать данные по количеству районов в округах
8. Отфильтровать данные полям: Tunnel{global\_id}, AdmArea. Данные для фильтрации вводятся пользователем.
9. Сохранять результаты редактирований, сортировок и фильтраций в CSV-файл. Режимы сохранения в файл: создание нового файла, замена содержимого уже существующего файла, добавление сохраняемых данных к содержимому существующего файла.

Интерфейс приложения должен соответствовать следующим требованиям:

1. Для управления файлом использовать OpenFileDialog и SaveFileDialog
2. Для отображения данных использовать DataGrid
3. Количество отображаемых элементов выбирается пользователем и не превышает количества строк с данными в файле

Требования к устойчивости:

1. В случае ошибок при открытии/сохранении файла или некорректных данных программа должна выводить сообщение
2. Аварийные ситуации должны обрабатываться, пользователю должны выводиться сообщения об ошибке.
3. **Функции разрабатываемого приложения**

**2.1 Варианты использования**

Данное приложение может использоваться при работе с CSV – файлами, которые соответствуют данному в задании формату.

**2.2 Описание интерфейса**

Ниже приведен снимок экрана, на котором показан интерфейс программы.

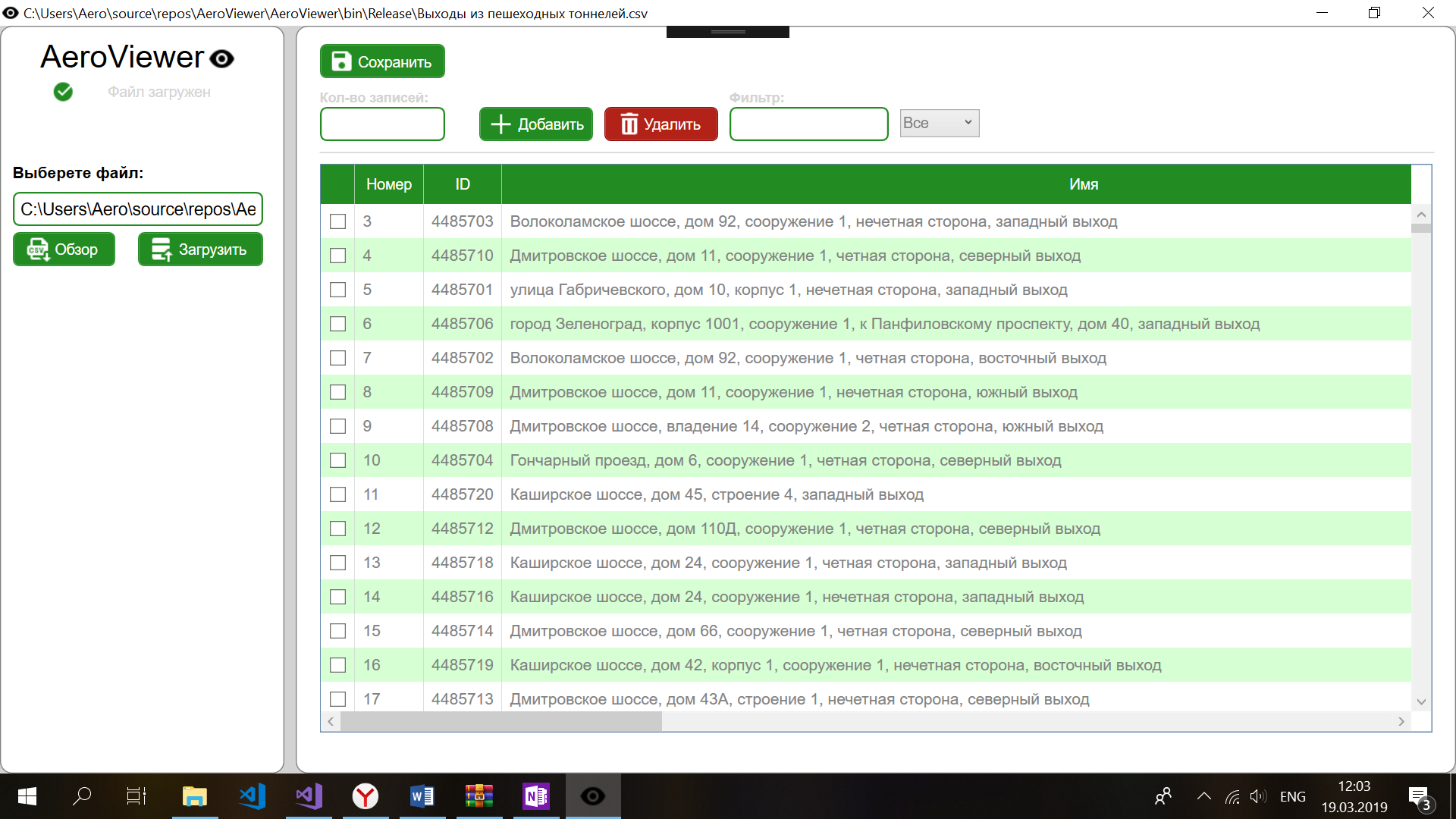


Рисунок 1 – интерфейс программы

В левом блоке расположены логотип программы, текстовое поле для написания пути файла, кнопка «Обзор», с помощью которой можно выбрать файл для открытия, кнопка «Загрузить», нажатие на которую инициализирует процесс загрузки данных из файла в таблицу.

В правом блоке расположена таблица, в которую выгружаются все данные из выбранного файла. Над таблицей расположена панель управления. В панели управления находятся следующие элементы: кнопка сохранения изменений, кнопка «Добавить», с помощью которой можно добавить новую запись, кнопка «Удалить», с помощью которой можно удалить все выбранные записи, текстовое поле для ввода фильтров, ComboBox для выбора режима фильтрации, TextBox для выбора количества видимых записей.

В таблице самый левый столбец служит для выбора элементов, при щелчке на заголовок каждой колонки, данные будут отсортированы по выбранной колонке, все колонки последней можно редактировать, так как последняя колонка показывает статус загрузки.

При нажатии на копку «Сохранить» перед пользователем появляется окно сохранения файла (Рисунок 2)

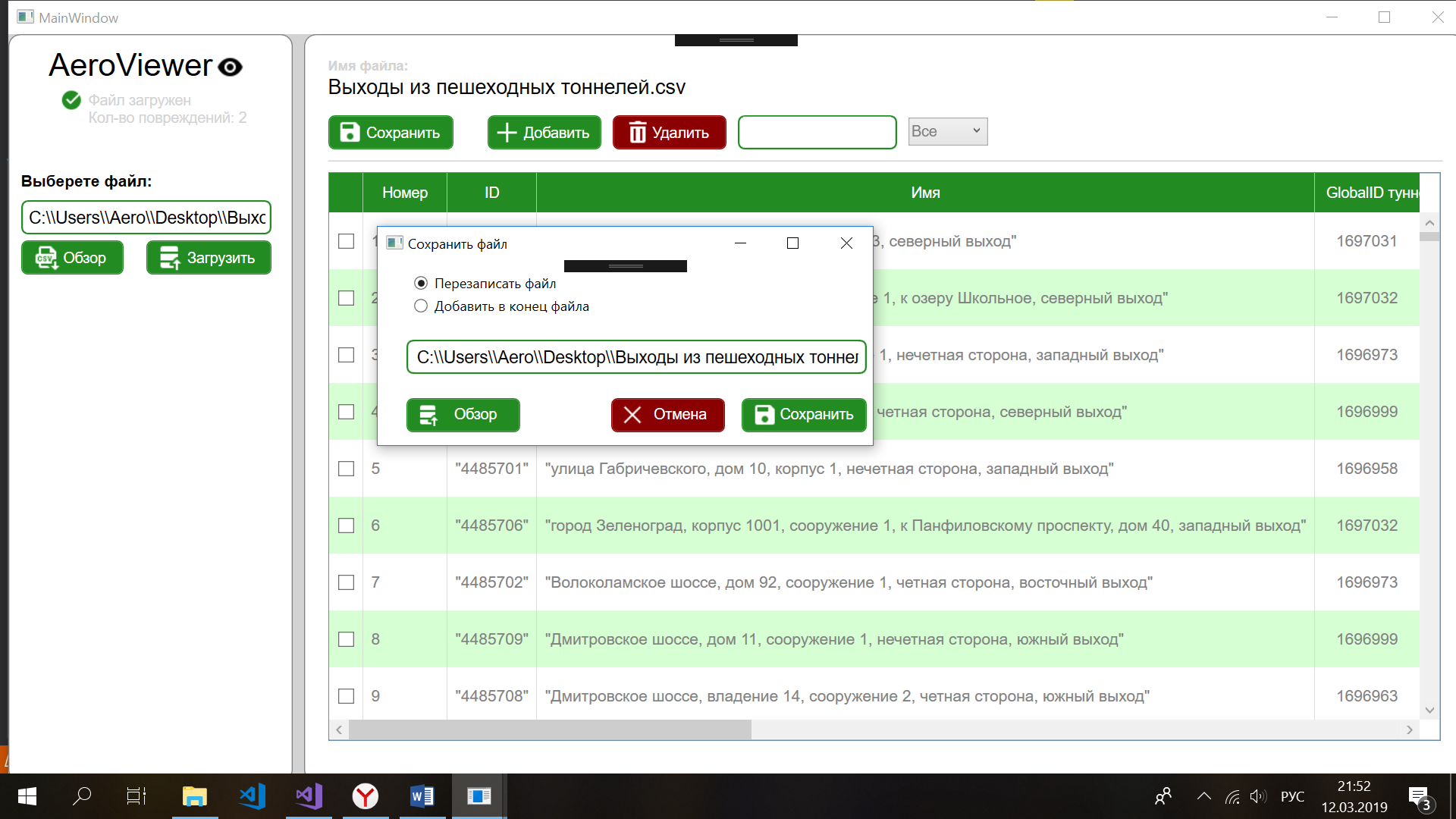


Рисунок 2 – окно для сохранения файла

В этом окне пользователь может выбрать режим сохранения файла, выбрать место сохранения и сохранить файл.

При возникновении ошибок в программе появляется окно с сообщением об ошибке (Рисунок 3)

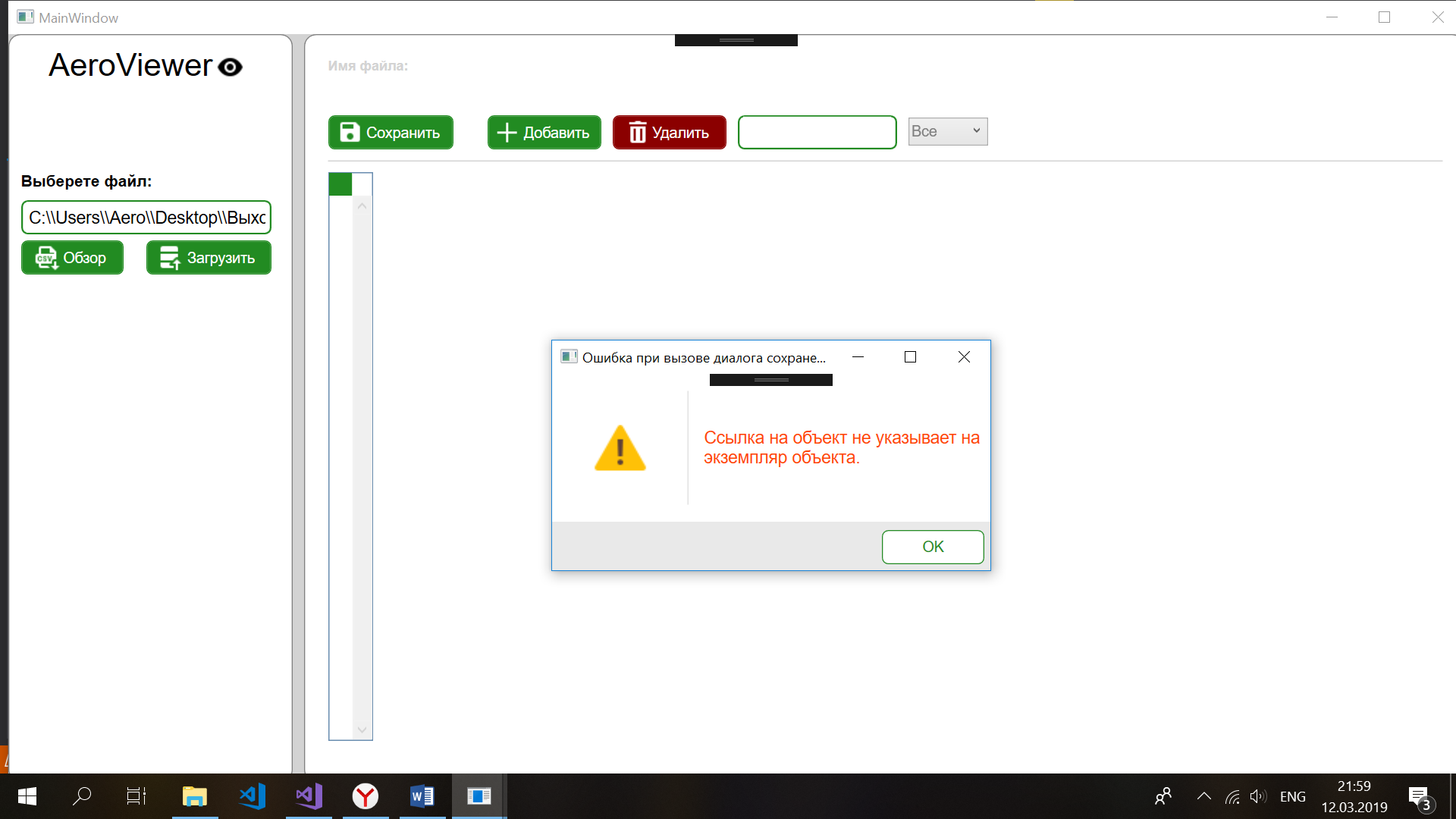


Рисунок 3 – пример окна с сообщением об ошибке

1. **Структура приложения**

**3.1 Диаграмма классов**

Ниже представлена диаграмма классов (Рисунок 4).

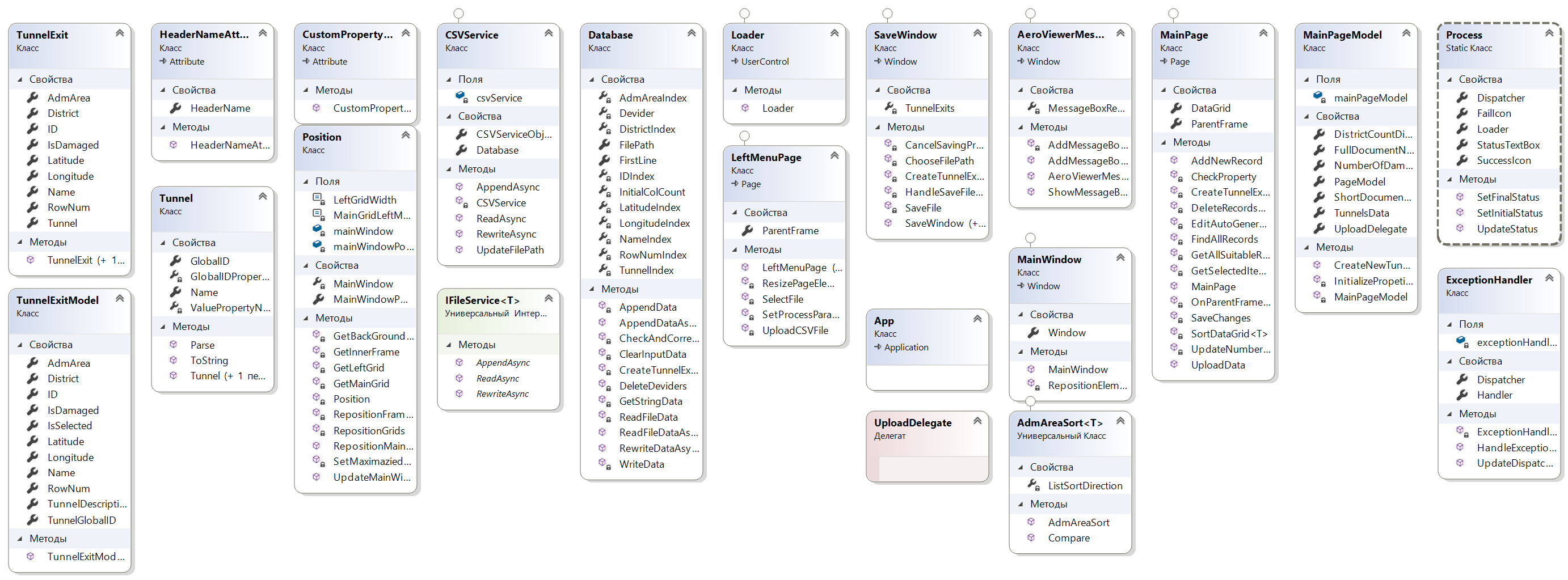


Рисунок 4 – диаграмма классов приложения

**3.2 Описание классов, их полей и методов**

Класс MainPage – логика взаимодействия главной страницы в приложении

Конструктор – инициализирует ParentFrame, MainPageModel.UploadDelegate, ParentFrame.SizeChanged

Свойства: ParentFrame – фрейм в котором находиться страница

Методы: UploadData – загружает данные в data grid, EditAutoGeneratedColumns – меняет параметры некоторых колонок, SortDataGrid – задает отдельную сортировку для колонки «Округ», OnParentFrameSizeChanged – меняет размеры некоторых элементов, SaveChanges – запускает процесс сохранения файла, DeleteRecrodsAsync – удаляет выбранные записи, GetSelectedItems – возвращает все выбранные объекты, AddNewRecord – добавляет новую запись, FindAllRecords – ищет все записис удовлетворяющие фильтру, GetAllSuitableRecords – находит все записи в коллекции в завсисимости от фильтра, CreateTunnelExitList – создает список всех подходящих объектов, CheckProperty – проверяет подходит ли свойство под фильтр

Атрибут CustomProperty – указывает что данной свойство не было в исходных данных.

Атрибут HeaderNameAttribute – указывает название свойства в таблице

Свойство HeaderName – имя колонки в таблице

Конструктор – инициализирует данное свойство

Класс Database – работает непосредственно с CSV файлом

Свойства: IDIndex – индекс id в файле, LatitudeIndex - индекс latitude в файле, LongtitudeIndex – индекс longitude в файле, DistrictIndex – индекс district в файле,

AdmAreaIndex - индекс admArea в файле, TunnelIndex - индекс tunnel в файле, NameIndex – индекс name в файле, RowNumIndex - индекс RowNum в файле, FilePath – путь текущего файла, FirstLine – заголовочная строка файла (названия столбцов)

Методы: ReadFileDataAsync – читает данные из файла, CreateTunnelExitObject создает объект TunnelExit, GetStringData – получает строковое представление данных, RewriteDataAsync – перезаписывает файл, AppendDataAsync – добавляет данные в конец файла

Интерфейс IFileService – определяет функционал сервисов для работы с файлами

Методы: ReadAsync – читает данные, RewriteAsync – перезаписывает данные, AppendAsync – добавляет данные в конец файла

Класс Tunnel – представляет тоннель

Свойства: GlobalIDPropertyName – имя параметра globalID в файловой кодировке, ValuePropertyName – имя параметра Value в файловой кодировке, GlobalID – значение GlobalID, Name – значение Value

Конструкторы – пустой конструктор, Tunnel с входными параметрами globalID, name –инициализируют свойства GlobalID, Name

Методы: Parse – создает объект Tunnel из входной строки из файла, ToString – возвращает закодированное в файловой кодировке.

Класс TunnelExit – представляет сущность (выход из тоннеля)

Свойства – RowNum – номер строки, Name – имя выхода, Tunnel – информация о тоннеле, AdmArea – округ, District – район, Longitude – долгота, Latitude – широта, ID – id, IsDamaged – был ли поврежден объект

Конструкторы: пустой конструктор, Конструктор принимающий на вход объект TunnelExitModel и строящий на основании данных из входного параметра объект TunnelExit.

LeftMenuPage – логика взаимодействия для страницы меню.

Свойства: ParentFrame – родительский фрейм

Конструкторы: Конструктор без параметров с InitializeComponent, конструктор принимающий родительский фрейм как параметр и инициализирующий свойство.

Методы: SetProcessParams – иницализирует свойства класса Process, ResizeElements – меняет размеры нескольких элементов, UploadCSVFile – запускает процесс загрузки CSV файла, SelectFile – выбирает файл для загрузки

Loader – Логика взаимодействия для лоадера.

Конструктор не принимающий входных параметров и запускающий анимацию лоадера.

MainWindow – логика взаимодействия для главного окна

Свойство Window – хранит текущее окно

Конструктор инициализирует фреймы и класс Position

Методы RepositionElements – меняет размер и положение некоторых элементов

SaveWindow – логика взаимодействия для окна сохранения изменений

Свойство TunnelExits – лист из объектов TunnelExit

Конструкторы: конструктор без параметров который инициализирует окно, конструктор принимающий в качестве входного параметра коллекцию из объектов TunnexExitModel.

Методы: CreateTunnelExitsList – создает лист из объектов TunnelExit, SaveFile – запускает процесс сохранения файла, HandleSaveFileExeption – обрабатывает исключения при сохранении файла, ChooseFilePath – выбирает куда сохранить файл, CancelSavingProcess – закрывет это окно

AeroViewerMessageBox – логика взаимодействия для AeroViewerMessageBox

Свойства MessageBoxResult – результат диалога

Конструктор – инициализирует окно окна сообщений

Методы: ShowMessageBox – показывает message box, AddMessageBoxButtons – добавляет кнопки на мессадж бокс, AddMessageBoxButton – добавляет одну кнопку на message box

Класс CSVService – сервис выполняющий операции с CSV файлами

Поля: csvService – хранит объект CSVService

Свойства: DataBase – ссылка на экземпляр класса Database, CSVServiceObject – дает доступ к объекту csvService.

Конструктор – приватный конструктор для создания объекта

Методы: UpdateFilePath – обновляет путь до файла, ReadAsync – читает данные из файла, RewriteAsync – перезаписывает данные в файл, AppendAsync – добавляет записи в конец файла.

Класс AdmAreaSort – отвечает за кастомную сортировку колонки Округ

Свойство ListSortDirection – отвечает за направление сортировки

Конструктор инициализирует это свойство

Метод Compare сравнивает два объекта

Класс MainPageModel – модель для MainPage

Поля: mainPageModel – хранит объект MainPageModel

Свойства: ShortDocumentName – короткое имя документа, FullDocumentName – полное имя документа, TunnelsData – лист из TunnelExitModel, UploadDelegate – делегает который выполняется после обновления TunnelsData, NumberOfDamagedRecords – количество поврежденных записей, DistrictDictionaryCount – словарь в котором храниться количество районов каждого округа, PageModel – доступ к mainPageModel

Приватный конструктор – инициализирует TunnelsData

Методы: CreateNewTunnelData – заполняет TunnelsData, InitializeProperties – инициализирует свойства.

Класс TunnelExitModel – модель выхода из тоннеля

Свойтсва: IsSelected – выбрана ли запись, RowNum – номер строки, ID – id, Name – имя выхода из тоннеля, TunneGlobalID – глобал id, TunnelDescriprion – описание тоннеля, AdmArea – округ, District – район, Longitude – долгота, Latitude – широта, IsDamaged – повреждена ли запись

Конструкторы – пустой конструктор, конструктор – строит объект этого класса по объекту TunnelExit

Класс ExceptionHandler – через этот класс обрабатываются все исключения

Поле: exceptionHamder – ссылка на объект ExceptionHandler

Свойства: Dispatcher – диспетчер потока который создал UI, Handler – возыращает объект ExceptionHandler

Приватный пустой конструктор

Методы: HandleExceptionWithMessageBox – обрабатывает исключение с помощью окна ошибки, UpdateDispatcher обновляет свойство диспетчера

Position – отвечает за размеры и позицию элементов

Константы: LeftGridWidth – длина левого Grid, MainGridLeftMargin – левый отступ левого Grid

Поля: mainWindowPosition – ссылка на объект Posotion, mainWindow – ссылка на главное окно

Свойства: MainWindowPosition – доступ к объекту позиции, MainWindow – доступ к mainWindow

Приватный пустой конструктор

Методы: SetMaximaziedParams – задает максимальные размеры окну, UpdateMainWindow обновляет объект главного окна, RepositionMainGrid меняет размер и позицию некоторых Grid и Frame, RepositionGrids – меняет размер и позицию нескольких Grid, GetLeftGrid – получает левый Grd, GetBackgrundGrid – получает фоновую Grid, GetMainGrid – получает MainGrid, RepositionFrames меняет размер и позицию некоторых Frameов, GetInnerFrame – получает внутренний фрейм в Grid.

Класс Process отвечает за обновление статуса процесса.

Свойства: Loader – ссылка на объект лоадера, StatusTextBox –ссылка на объект TextBox, SuccesIcon –иконка при удачном завершении процесса, FailIcon – иконка при неудачном завершении процесса, Dispatcher – ссылка на объект dispathcer/

Методы: SetInitialStatus – задает начальный статус,UpdateStatus –обновляет статус, SetFinalStatus – задает финальный статус

**3.3 Архитектура приложения**

В данном разделе описаны основные паттерны и решения, которые были использованы при создании приложения

1. MVVM (Model – View – ViewModel)

В рамках этого паттерна реализован класс TunnelExit, который является моделью данных, в объекты данного класса записывается информация после прочтения из файла. Далее, при отображении данных на UI используется класс TunnelExitModel, данные из которого и отображаются в таблице. Схематично реализация паттерна изображена на Рисунке 5.



Рисунок 5 – схема реализации MVVM

1. Singleton

В классах MainPageModel, Position, ExceptionHandler, CSVService реализован паттерн Singleton, то есть гарантируется наличие максимум одного объекта этих классов в программе.

1. Сервисная архитектура

Для реализации работы с CSV файлом создана следующая архитектура: класс Database отвечает за непосредственное чтение и редактирование файла. Класс CSVService является API, посредством которого идет осуществление операций, определенных в IFileService (CSVService реализует IFileService). Схема этой архитектуры представлена ниже (Рисунок 6):

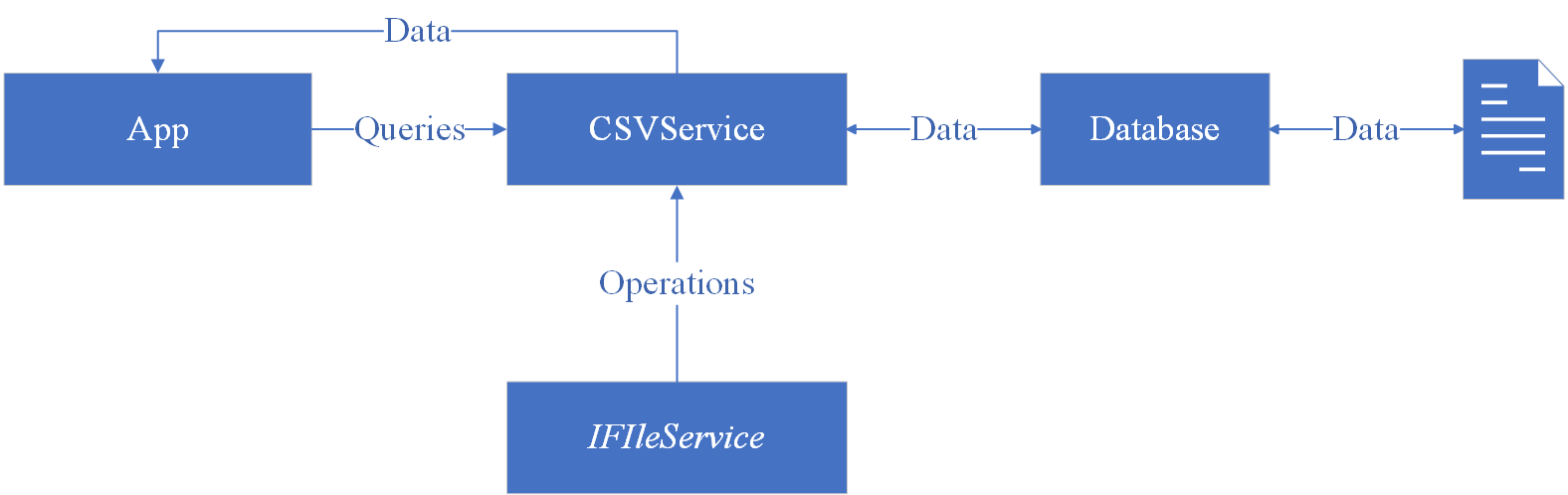


Рисунок 6 – архитектура работы с данными

1. Атрибуты, исключения и рефлексия

Все исключения обрабатываются с помощью методов класса ExceptionHandler, что дает возможность централизованно задавать методы обработки и в дальнейшем вести записи об ошибках.

С помощью атрибутов помечаются свойства классов TunnelExit и TunnelExitModel, что дает возможность легко изменять названия столбцов в таблице и определять, является ли свойство представлением данных из источника данных (CSV файл) или созданным программистом для других целей.

Для создания данных для записи/добавления в файл используется рефлексия.

1. Отклонения от спецификации

Мною было принято и реализовано несколько решений которые не соответствуют спецификации. Ниже объяснена причина этих решений:

1. Так как в файле представлена информация о выходах из пешеходных тоннелей, модель имеет название не «Тоннель», а «Выход из тоннеля»
2. В моделе данных не создавался отдельный класс «Район», так как в файле поля, составляющие этот класс представлены двумя отдельными колонками. Создание класса «Район» затрудняет передачу данных между моделью и источником данных и автоматическое (с помощью рефлексии) создание новых данных для файлов.
3. В моделе было принято решение сделать поле «Тоннель» - объектом класса «Tunnel», так как в таблице этот столбец представлен двумя полями (Global ID и Value)
4. **Распределение исходного кода по файлам проекта**

Каждому классу проекта соответствует отдельный файл с именем этого класса.

1. **Контрольный пример и описание результата**

В этом разделе описана последовательность шагов для демонстрации работы программы:

1. Откройте программу
2. Нажмите на кнопку «Обзор» в левой части экрана, выберете файл для открытия, нажмите кнопку загрузить. Программа выведет содержимое файла в таблице (Рисунок 7)

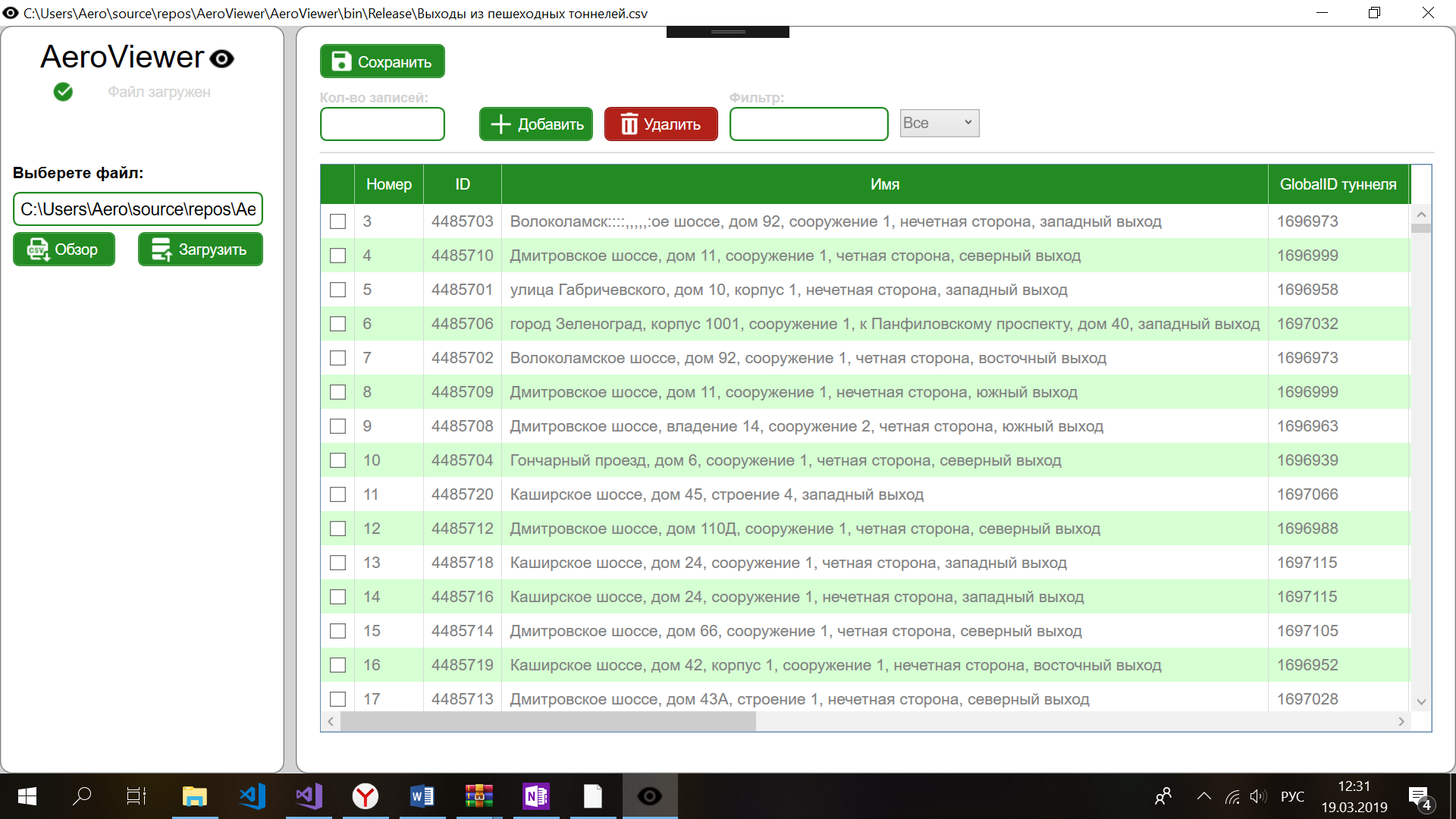


Рисунок 7 – файл открыт

1. Введите цифру 10 в поле для ввода для изменения количества видимых записей. Программа отобразит первые 10 записей.

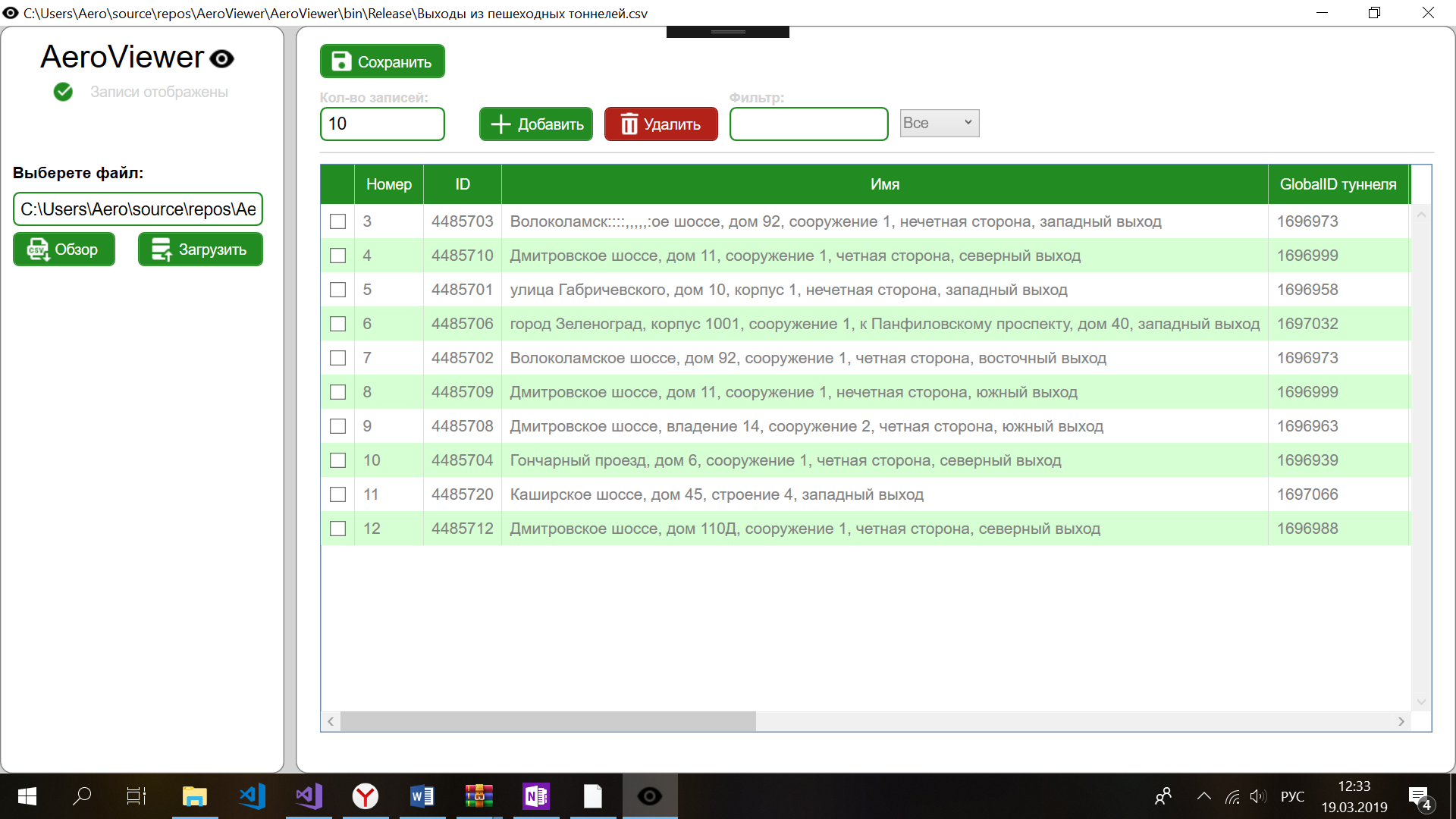


Рисунок 8 – регулирование количества видимых записей

1. Нажмите кнопку «Добавить». В конец данных добавиться новая запись, которую можно редактировать.

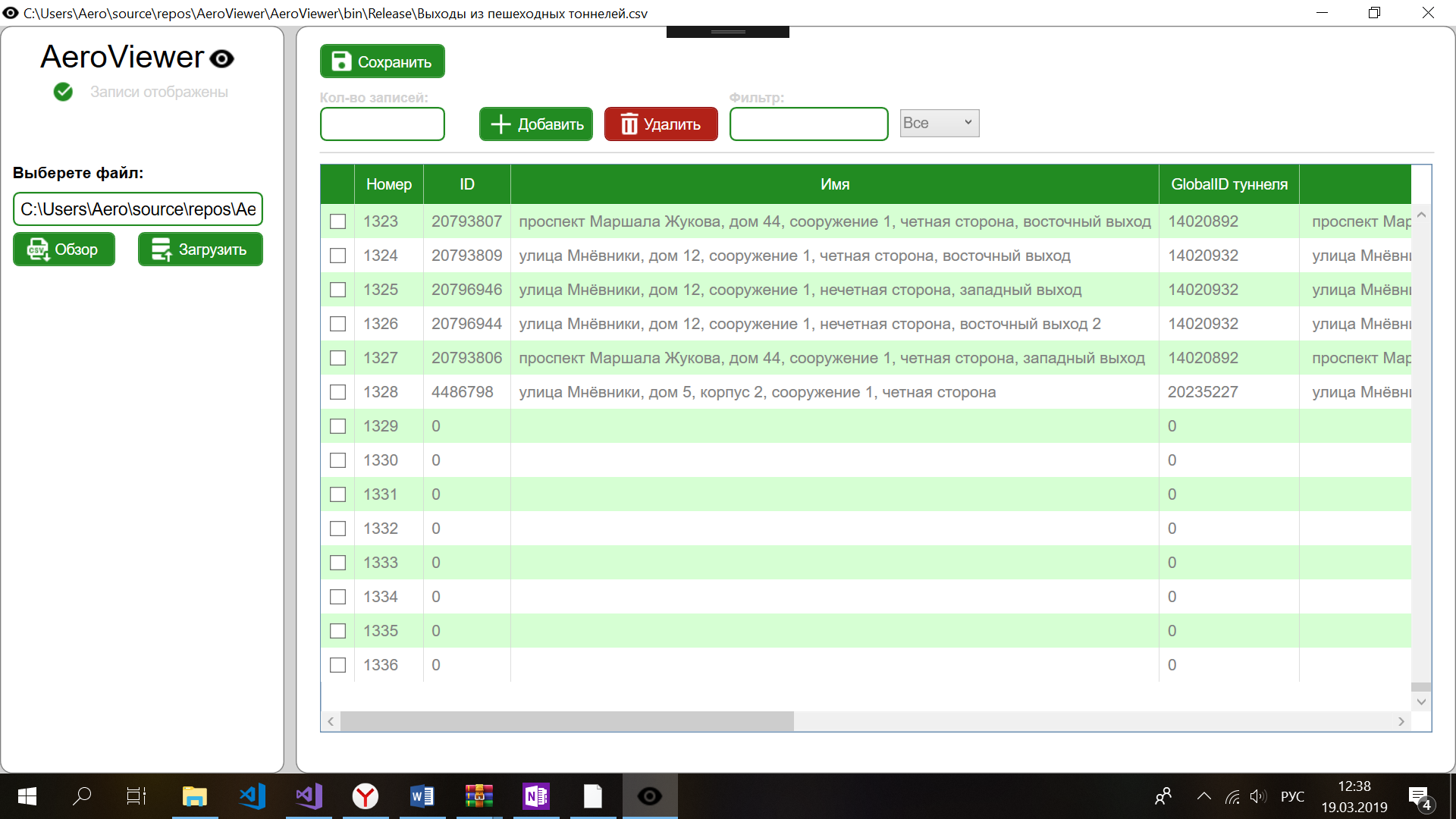


Рисунок 9 – добавление новой записи

1. Вернитесь к началу списка. Испытайте возможность сортировки каждой колонки (нужно нажать на шапку колонки)
2. Выберете первую запись (поставьте флажок в CheckBox в левой колонке) и нажмите кнопку удалить. Запись будет удалена
3. Испытайте возможность фильтрации (фильтр можно ввести в поле ввода для фильтра)
4. Отредактируйте первую запись (отредактировать любую ячейку, кроме столбца «Статус загрузки», можно путем двойного нажатия на ячейку)
5. Нажмите на кнопку сохранить, выберете пункт «Перезаписать файл» и нажмите сохранить. Откройте исходный файл, все изменения были сохранены. Также можно создать новый файл, или добавить записи в конец другого файла.

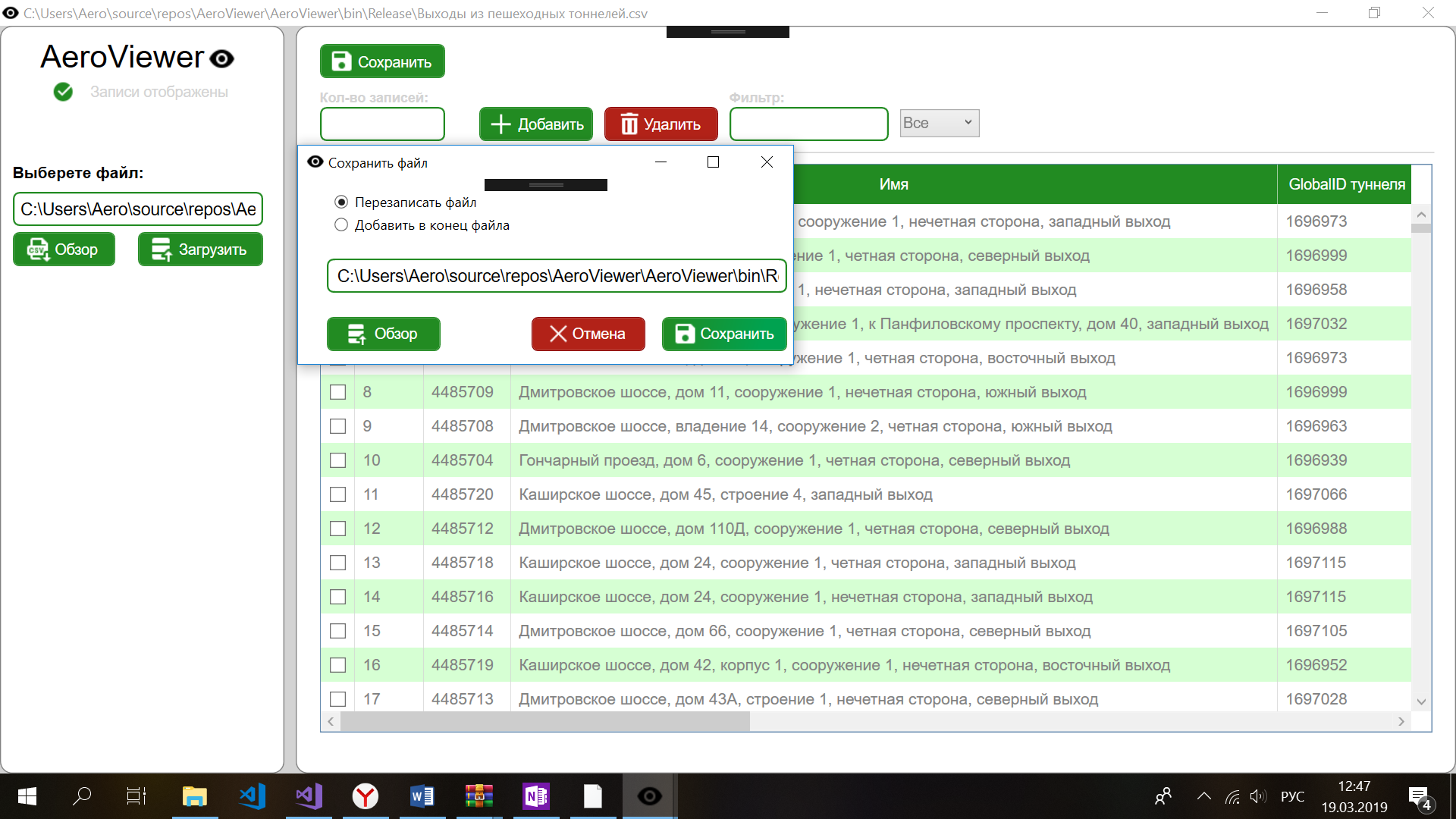


Рисунок 10 – окно для сохранения файла

**Приложение А**

**Исходный код программы**

**CustomPropertyAttribute.cs**

using System;

namespace AeroViewer.Attributes

{

/// <summary>

/// Attribute which marks properties in a class, which represents CSV data,

/// which are added by a programmer (so they are not in a file)

/// </summary>

public class CustomPropertyAttribute : Attribute

{

public CustomPropertyAttribute() { }

}

}

**HeaderNameAttribute.cs**

using System;

namespace AeroViewer.Attributes

{

/// <summary>

/// Stores the name which will displayed in a data grid (as column header)

/// </summary>

public class HeaderNameAttribute : Attribute

{

public string HeaderName { get; set; }

public HeaderNameAttribute(string headerName) =>

HeaderName = headerName;

}

}

**Database.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading.Tasks;

using AeroViewer.Models;

using System.IO;

using System.Reflection;

using AeroViewer.Attributes;

using System.Linq;

namespace AeroViewer.Data

{

/// <summary>

/// Class where all methods which directly connect to the CSV file are placed

/// </summary>

public class Database

{

#region CSV File properties

private int InitialColCount { get; set; }

private char Devider { get; } = ';';

private int IDIndex { get; } = 7;

private int LatitudeIndex { get; } = 6;

private int LongitudeIndex { get; } = 5;

private int DistrictIndex { get; } = 4;

private int AdmAreaIndex { get; } = 3;

private int TunnelIndex { get; } = 2;

private int NameIndex { get; } = 1;

private int RowNumIndex { get; } = 0;

#endregion

#region Properties

public string FilePath { get; set; } = string.Empty;

/// <summary>

/// First (Head) line of a file, where column names are

/// </summary>

public string[] FirstLine { get; private set; } = new string[1];

#endregion

public async Task<List<TunnelExit>> ReadFileDataAsync()

{

return await Task.Run(() =>

{

List<TunnelExit> tunnelExitsList = new List<TunnelExit>();

List<string> tunnelExitsStringData = ReadFileData(FilePath);

ClearInputData(tunnelExitsStringData);

FirstLine = tunnelExitsStringData[0].Split(Devider);

InitialColCount = FirstLine.Length;

for (int i = 1; i < tunnelExitsStringData.Count; i++)

tunnelExitsList.Add(CreateTunnelExitObject(tunnelExitsStringData[i]));

return tunnelExitsList;

});

}

private List<string> ReadFileData(string filePath)

{

List<string> stringData = new List<string>();

using (FileStream fs = new FileStream(filePath, FileMode.Open))

using (StreamReader sr = new StreamReader(fs, Encoding.UTF8))

{

string line;

while ((line = sr.ReadLine()) != null)

stringData.Add(line);

}

return stringData;

}

private void ClearInputData(List<string> tunnelExitsStringData)

{

for (int i = 0; i < tunnelExitsStringData.Count; i++)

{

if (tunnelExitsStringData[i][tunnelExitsStringData[i].Length - 1] == ';')

{

tunnelExitsStringData[i] = tunnelExitsStringData[i].Remove(tunnelExitsStringData[i].Length - 1, 1);

}

}

}

/// <summary>

/// Creates a TunnelExit object out of a CSV file string

/// </summary>

private TunnelExit CreateTunnelExitObject(string tunnelExitString)

{

TunnelExit tunnelExit = new TunnelExit();

try

{

string[] tunnelExitStringData = tunnelExitString.Split(Devider);

CheckAndCorrectInputStringData(tunnelExitStringData);

tunnelExit = new TunnelExit

{

ID = long.Parse(tunnelExitStringData[IDIndex]),

RowNum = int.Parse(tunnelExitStringData[RowNumIndex]),

Name = tunnelExitStringData[NameIndex],

Tunnel = Tunnel.Parse(tunnelExitStringData[TunnelIndex]),

AdmArea = tunnelExitStringData[AdmAreaIndex],

District = tunnelExitStringData[DistrictIndex],

Latitude = double.Parse(tunnelExitStringData[LatitudeIndex].Replace('.', ',')),

Longitude = double.Parse(tunnelExitStringData[LongitudeIndex].Replace('.', ',')),

IsDamaged = "OK"

};

return tunnelExit;

}

catch (ArgumentNullException)

{

tunnelExit.IsDamaged = "Damaged";

return tunnelExit;

}

catch (Exception)

{

tunnelExit.IsDamaged = "Damaged";

return tunnelExit;

}

}

private void CheckAndCorrectInputStringData(string[] tunnelExitStringData)

{

if (tunnelExitStringData.Length != InitialColCount)

throw new ArgumentException();

for (int i = 0; i < InitialColCount; i++)

{

while (tunnelExitStringData[i].IndexOf("\"") > -1)

tunnelExitStringData[i] = tunnelExitStringData[i].Remove(

tunnelExitStringData[i].IndexOf("\""), 1);

}

}

public async Task RewriteDataAsync(List<TunnelExit> tunnelExits)

{

await Task.Run(() =>

{

string[][] data = GetStringData(tunnelExits, true);

string[] csvData = data.Select(x => string.Join(";", x)).ToArray();

WriteData(csvData);

});

}

private void WriteData(string[] data)

{

using (FileStream fs = new FileStream(FilePath, FileMode.OpenOrCreate))

using (StreamWriter sw = new StreamWriter(fs, Encoding.UTF8))

{

foreach (string dataString in data)

sw.WriteLine(dataString);

}

}

/// <summary>

/// Creates a jagged array out of a Tunnel exit list, where each array is data for CSV data string

/// </summary>

private string[][] GetStringData(List<TunnelExit> tunnelExits, bool? addHeaderString)

{

string[][] data;

int startIndex = 0;

if (addHeaderString == true)

{

data = new string[tunnelExits.Count + 1][];

data[0] = FirstLine;

startIndex = 1;

}

else

{

data = new string[tunnelExits.Count][];

startIndex = 0;

}

PropertyInfo[] modelProperties = typeof(TunnelExit).GetProperties(

BindingFlags.Public | BindingFlags.Instance);

for (int i = startIndex; i < data.GetLength(0); i++)

{

List<string> dataString = new List<string>();

foreach (PropertyInfo propertyInfo in modelProperties)

if (propertyInfo.GetCustomAttribute<CustomPropertyAttribute>() == null)

{

string propertyStringValue = DeleteDeviders(propertyInfo.GetValue(

tunnelExits[i - startIndex]).ToString());

dataString.Add(propertyStringValue);

}

data[i] = dataString.ToArray();

}

return data;

}

private string DeleteDeviders(string propertyStringValue)

{

while (propertyStringValue.IndexOf(Devider) > -1)

propertyStringValue = propertyStringValue.Remove(propertyStringValue.IndexOf(Devider), 1);

return propertyStringValue;

}

public async Task AppendDataAsync(List<TunnelExit> tunnelExits, bool? addHeaderString)

{

await Task.Run(() =>

{

string[][] data = GetStringData(tunnelExits, addHeaderString);

string[] csvData = data.Select(x => string.Join(";", x)).ToArray();

AppendData(csvData);

});

}

private void AppendData(string[] data)

{

using (FileStream fs = new FileStream(FilePath, FileMode.Append))

using (StreamWriter sw = new StreamWriter(fs, Encoding.UTF8))

{

foreach (string dataString in data)

sw.WriteLine(dataString);

}

}

}

}

**IFileService.cs**

using System.Collections.Generic;

using System.Threading.Tasks;

namespace AeroViewer.Interfaces

{

/// <summary>

/// Interface which defines the methods which must be implemented by all

/// services that work with files

/// </summary>

/// <typeparam name="T">File data model class</typeparam>

interface IFileService<T>

{

Task<List<T>> ReadAsync();

Task RewriteAsync(List<T> data);

Task AppendAsync(List<T> data, bool? addHeaderString);

}

}

namespace AeroViewer.Models

{

public class Tunnel

{

#region Parse properties

private static string GlobalIDPropertyName { get; } = "global\_id:";

private static string ValuePropertyName { get; } = "value:";

#endregion

#region Properties

public long GlobalID { get; set; }

public string Name { get; set; } = string.Empty;

#endregion

#region Constructors

public Tunnel() { }

public Tunnel(long globalID, string name)

{

GlobalID = globalID;

Name = name;

}

#endregion

/// <summary>

/// Creates a tunnel object from a string from CSV file

/// </summary>

public static Tunnel Parse(string tunnelData)

{

int globalIDStartIndex = tunnelData.IndexOf(GlobalIDPropertyName) +

GlobalIDPropertyName.Length;

int valueStartIndex = tunnelData.IndexOf(ValuePropertyName) +

ValuePropertyName.Length;

long globalID = long.Parse(tunnelData.Substring(globalIDStartIndex,

tunnelData.IndexOf(",") - globalIDStartIndex));

string value = tunnelData.Substring(valueStartIndex,

tunnelData.IndexOf("}") - valueStartIndex);

return new Tunnel

{

GlobalID = globalID,

Name = value

};

}

/// <summary>

/// Returns the data representation in a format which is defined in the initial CSV file

/// </summary>

public override string ToString() =>

"\"{ \"\"global\_id\"\": " + GlobalID + ", \"\"value\"\": \"\"" + Name + "\"\" }\"";

}

}

using AeroViewer.Attributes;

using AeroViewer.ViewModels;

using System;

namespace AeroViewer.Models

{

public class TunnelExit

{

#region Properties

public int RowNum { get; set; }

public string Name { get; set; } = string.Empty;

public Tunnel Tunnel { get; set; }

public string AdmArea { get; set; } = string.Empty;

public string District { get; set; } = string.Empty;

public double Longitude { get; set; }

public double Latitude { get; set; }

public long ID { get; set; }

[CustomProperty]

public string IsDamaged { get; set; }

#endregion

public TunnelExit() { }

public TunnelExit(TunnelExitModel tunnelExitModel)

{

try

{

RowNum = tunnelExitModel.RowNum;

ID = tunnelExitModel.ID;

Name = tunnelExitModel.Name;

AdmArea = tunnelExitModel.AdmArea;

District = tunnelExitModel.District;

Tunnel = new Tunnel(tunnelExitModel.TunnelGlobalID, tunnelExitModel.Name);

Longitude = tunnelExitModel.Latitude;

Latitude = tunnelExitModel.Longitude;

}

catch (Exception)

{

throw new ArgumentException("Неправильно введены данные");

}

}

}

}

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace AeroViewer

{

public partial class AeroViewerMessageBox : Window

{

#region Properties

private MessageBoxResult MessageBoxResult { get; set; } = MessageBoxResult.None;

#endregion

public AeroViewerMessageBox()

{

InitializeComponent();

}

public static MessageBoxResult ShowMessageBox(string title, string message,

MessageBoxButton buttons)

{

AeroViewerMessageBox messageBox = new AeroViewerMessageBox { Title = title };

messageBox.messageTextBlock.Text = message;

messageBox.AddMessageBoxButtons(buttons);

messageBox.ShowDialog();

return messageBox.MessageBoxResult;

}

#region Add buttons methods

/// <summary>

/// Adds the buttons which were defined by the developer to the messagebox

/// </summary>

public void AddMessageBoxButtons(MessageBoxButton buttons)

{

switch (buttons)

{

case MessageBoxButton.OK:

AddMessageBoxButton("OK", MessageBoxResult.OK, isDefault: true);

break;

case MessageBoxButton.OKCancel:

AddMessageBoxButton("Cancel", MessageBoxResult.Cancel,

isCancel: true);

AddMessageBoxButton("OK", MessageBoxResult.OK, isDefault: true);

break;

case MessageBoxButton.YesNo:

AddMessageBoxButton("Yes", MessageBoxResult.Yes, isDefault: true);

AddMessageBoxButton("No", MessageBoxResult.No);

break;

case MessageBoxButton.YesNoCancel:

AddMessageBoxButton("No", MessageBoxResult.No);

AddMessageBoxButton("Yes", MessageBoxResult.Yes, isDefault: true);

AddMessageBoxButton("Cancel", MessageBoxResult.Cancel,

isCancel: true);

break;

}

}

private void AddMessageBoxButton(string text, MessageBoxResult messageBoxResult,

bool isCancel = false, bool isDefault = false)

{

Button button = new Button()

{

Content = text,

IsCancel = isCancel,

IsDefault = isDefault

};

button.Click += (sender, e) =>

{

MessageBoxResult = messageBoxResult;

DialogResult = true;

};

buttonsGrid.Children.Add(button);

}

#endregion

}

}

<Window x:Class="AeroViewer.AeroViewerMessageBox"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

Title="AeroViewerMessageBox"

MinHeight="210"

MinWidth="400"

MaxHeight="210"

MaxWidth="400">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="OverridesDefaultStyle" Value="True"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Width" Value="90"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="BorderThickness" Value="2"/>

<Setter Property="BorderBrush" Value="{StaticResource darkMainThemeBrush}"/>

<Setter Property="Background" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="Margin" Value="0, 0, 20, 0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border CornerRadius="5"

BorderBrush="{StaticResource darkMainThemeBrush}"

BorderThickness="1"

SnapsToDevicePixels="True">

<Border.Background>

<LinearGradientBrush>

<GradientStop Color="{StaticResource darkMainThemeColor}"></GradientStop>

<GradientStop Color="{StaticResource darkMainThemeColor}" Offset="0.8"></GradientStop>

</LinearGradientBrush>

</Border.Background>

<ContentPresenter HorizontalAlignment="Center"

VerticalAlignment="Center"/>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="#00a250"

AutoReverse="False"/>

</Storyboard>

</VisualState>

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="{StaticResource darkMainThemeColor}"

AutoReverse="False"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Window.Content>

<Grid VerticalAlignment="Top"

HorizontalAlignment="Left"

Background="White"

Width="400"

Height="210">

<Grid x:Name="warningIconGrid"

Width="120"

Height="130"

Background="White"

VerticalAlignment="Top"

HorizontalAlignment="Left">

<Image x:Name="warningIcon"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Width="60"

Height="60"

Source="../Resources/alertIcon.png"/>

<Rectangle Width="1"

Height="100"

VerticalAlignment="Center"

HorizontalAlignment="Right"

Fill="#E1E1E1"/>

</Grid>

<Grid x:Name="messageGrid"

VerticalAlignment="Top"

HorizontalAlignment="Left"

Background="White"

Width="270"

Height="130"

Margin="120, 0, 0, 0">

<TextBlock x:Name="messageTextBlock"

FontFamily="Arial, Verdana"

FontSize="15"

Margin="10, 10, 10, 10"

TextWrapping="WrapWithOverflow"

Foreground="OrangeRed"

VerticalAlignment="Center"

HorizontalAlignment="Center"/>

</Grid>

<Grid VerticalAlignment="Top"

HorizontalAlignment="Center"

Width="400"

Height="44"

Background="#E9E9E9"

Margin="0,130,0,0">

<StackPanel x:Name="buttonsGrid"

VerticalAlignment="Top"

HorizontalAlignment="Right"

Height="44"

Orientation="Horizontal"

Background="#E9E9E9"/>

</Grid>

</Grid>

</Window.Content>

</Window>

<Application x:Class="AeroViewer.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:AeroViewer"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Color x:Key="darkMainThemeColor">#228b22</Color>

<Color x:Key="dangerousActionsColor">#b22218</Color>

<SolidColorBrush x:Key="backgroundBrush" Color="Gray"/>

<SolidColorBrush x:Key="defaultTextBrush" Color="Black"/>

<SolidColorBrush x:Key="hintBrushText" Color="LightGray"/>

<SolidColorBrush x:Key="lightMainThemeBrush" Color="SpringGreen"/>

<SolidColorBrush x:Key="darkMainThemeBrush" Color="#228B22"/>

<SolidColorBrush x:Key="buttonTextBrush" Color="White"/>

<SolidColorBrush x:Key="dangerousActionsBrush" Color="#8B0000"/>

<SolidColorBrush x:Key="evenCellBorderBackground" Color="#79F8F8"/>

<SolidColorBrush x:Key="oddCellBorderBackground" Color="White"/>

<Style x:Key="appDefaultButtonStyle" TargetType="Button">

<Setter Property="OverridesDefaultStyle" Value="True"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Width" Value="110"/>

<Setter Property="Foreground" Value="{StaticResource buttonTextBrush}"/>

<Setter Property="Background" Value="{StaticResource darkMainThemeBrush}"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border x:Name="buttonBorder"

CornerRadius="5"

BorderBrush="{StaticResource darkMainThemeBrush}"

BorderThickness="1"

SnapsToDevicePixels="True">

<Border.Background>

<LinearGradientBrush>

<GradientStop Color="{StaticResource darkMainThemeColor}"></GradientStop>

<GradientStop Color="{StaticResource darkMainThemeColor}" Offset="0.8"></GradientStop>

</LinearGradientBrush>

</Border.Background>

<ContentPresenter HorizontalAlignment="Left"

VerticalAlignment="Top"

Height="30"

Width="110"/>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="#00a250"

AutoReverse="False"/>

</Storyboard>

</VisualState>

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="{StaticResource darkMainThemeColor}"

AutoReverse="False"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="appDangerousBtnStyle" TargetType="Button">

<Setter Property="OverridesDefaultStyle" Value="True"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Width" Value="110"/>

<Setter Property="Foreground" Value="{StaticResource buttonTextBrush}"/>

<Setter Property="Background" Value="{StaticResource dangerousActionsBrush}"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border CornerRadius="5"

BorderBrush="{StaticResource dangerousActionsBrush}"

BorderThickness="1"

SnapsToDevicePixels="True">

<Border.Background>

<LinearGradientBrush>

<GradientStop Color="{StaticResource dangerousActionsColor}"></GradientStop>

<GradientStop Color="{StaticResource dangerousActionsColor}" Offset="0.8"></GradientStop>

</LinearGradientBrush>

</Border.Background>

<ContentPresenter HorizontalAlignment="Left"

VerticalAlignment="Top"

Height="30"

Width="{TemplateBinding Width}"/>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="#C4453B"

Duration="0:0:6"

AutoReverse="False"/>

</Storyboard>

</VisualState>

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetProperty="Background.GradientStops[1].Color"

SpeedRatio="100"

To="{StaticResource dangerousActionsColor}"

Duration="0:0:6"

AutoReverse="False"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="appTextBoxStyle" TargetType="TextBox">

<Setter Property="OverridesDefaultStyle" Value="False"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="15"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Width" Value="220"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Padding" Value="5, 4.5, 5, 0"/>

<Setter Property="CaretBrush" Value="Black"/>

<Setter Property="Margin" Value="10, 145, 0, 0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="TextBox">

<Border BorderThickness="1.5, 1.5, 1.5, 1.5"

BorderBrush="{StaticResource darkMainThemeBrush}"

CornerRadius="5"

SnapsToDevicePixels="True">

<ScrollViewer Name="PART\_ContentHost"

Background="White"

Margin="{TemplateBinding Padding}"

Width="{TemplateBinding Width}"

Height="30"

HorizontalAlignment="Center"

VerticalAlignment="Center"/>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation To="LimeGreen"

SpeedRatio="100"

Storyboard.TargetProperty="BorderBrush.Color"/>

</Storyboard>

</VisualState>

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation To="{StaticResource darkMainThemeColor}"

SpeedRatio="100"

Storyboard.TargetProperty="BorderBrush.Color"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Application.Resources>

</Application>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace AeroViewer

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}

<Page x:Class="AeroViewer.LeftMenuPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

Title="LeftMenuPage"

MinWidth="250"

MinHeight="450"

Height="450"

Background="{StaticResource hintBrushText}"

VerticalAlignment="Top"

Width="250"

SizeChanged="ResizePageElements">

<Page.Resources>

<Style x:Key="backgroundGridStyle" TargetType="Grid">

<Setter Property="Width" Value="250"/>

<Setter Property="Height" Value="450"/>

<Setter Property="Background" Value="{StaticResource backgroundBrush}"/>

</Style>

<Style x:Key="appNameTextBlockStyle" TargetType="TextBlock">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="28"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="Margin" Value="0, 10, 35, 0"/>

</Style>

<Style x:Key="outterBorderStyle" TargetType="Border">

<Setter Property="Width" Value="250"/>

<Setter Property="Height" Value="450"/>

<Setter Property="Background" Value="White"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="CornerRadius" Value="10"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="BorderBrush" Value="{StaticResource backgroundBrush}"/>

</Style>

<Style x:Key="appLogoImageStyle" TargetType="Image">

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="Height" Value="24"/>

<Setter Property="Width" Value="24"/>

<Setter Property="Margin" Value="140, 16, 0, 0"/>

<Setter Property="Source" Value="../Resources/appLogoIcon.png"/>

</Style>

<Style x:Key="processTextBlockStyle" TargetType="TextBlock">

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="Margin" Value="30, 50, 0, 0"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="Foreground" Value="{StaticResource hintBrushText}"/>

</Style>

<Style x:Key="optionNameTextBlock" TargetType="TextBlock">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="Margin" Value="10, 120, 0, 0"/>

</Style>

<Style x:Key="processDoneImageIcon" TargetType="Image">

<Setter Property="Width" Value="20"/>

<Setter Property="Height" Value="20"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Margin" Value="44.5, 47, 0, 0"/>

</Style>

</Page.Resources>

<Grid x:Name="mainGrid">

<Border x:Name="outterBorder"

Style="{StaticResource outterBorderStyle}">

<Grid>

<TextBlock Style="{StaticResource appNameTextBlockStyle}"

Text="AeroViewer"/>

<Image Style="{StaticResource appLogoImageStyle}"/>

<TextBlock x:Name="statusTextBlock"

Style="{StaticResource processTextBlockStyle}"/>

<local:Loader x:Name="loader"

Margin="0,42,139,0"

VerticalAlignment="Top"

HorizontalAlignment="Center"

Visibility="Hidden"

Height="30"

Width="30"/>

<Image x:Name="processSuccessImage"

Style="{StaticResource processDoneImageIcon}"

Source="../Resources/processDoneIcon.png"

Visibility="Collapsed"/>

<Image x:Name="processFailIcon"

Style="{StaticResource processDoneImageIcon}"

Source="../Resources/failIcon.png"

Visibility="Collapsed"/>

<TextBlock Text="Выберете файл: "

Style="{StaticResource optionNameTextBlock}"/>

<TextBox x:Name="filePathTextBox"

Style="{StaticResource appTextBoxStyle}"

Text="C:\Users\Aero\Desktop\Выходы из пешеходных тоннелей.csv"/>

<Button Style="{StaticResource appDefaultButtonStyle}"

Margin="10, 180, 0, 0"

Width="90"

Click="SelectFile">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="10, 0, 0, 2"

Source="../Resources/importCSVButtonIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Text="Обзор"/>

</Grid>

</Button.Content>

</Button>

<Button Style="{StaticResource appDefaultButtonStyle}"

Margin="120, 180, 0, 0"

Width="110"

Click="UploadCSVFile">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="8, 0, 0, 2"

Source="../Resources/uploadCSVButtonIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="20, 0, 0, 0"

Text="Загрузить"/>

</Grid>

</Button.Content>

</Button>

</Grid>

</Border>

</Grid>

</Page>

using System;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using AeroViewer.Models;

using AeroViewer.Services;

using AeroViewer.Interfaces;

using AeroViewer.ViewModels;

using Microsoft.Win32;

namespace AeroViewer

{

public partial class LeftMenuPage : Page

{

public Frame ParentFrame { get; }

public LeftMenuPage() { InitializeComponent(); }

public LeftMenuPage(Frame parentFrame)

{

InitializeComponent();

SetProcessParams();

ParentFrame = parentFrame;

parentFrame.SizeChanged += ResizePageElements;

}

/// <summary>

/// Initializes the Process class properties,

/// which then used to show the stage of each process

/// </summary>

private void SetProcessParams()

{

Process.StatusTextBox = statusTextBlock;

Process.Dispatcher = Dispatcher;

Process.Loader = loader;

Process.SuccessIcon = processSuccessImage;

Process.FailIcon = processFailIcon;

}

/// <summary>

/// Changes the size of some elements when the size of the window is changed

/// </summary>

private void ResizePageElements(object sender, SizeChangedEventArgs e)

{

Width = ParentFrame.Width;

Height = ParentFrame.Height;

outterBorder.Width = Width;

outterBorder.Height = Height;

}

private async void UploadCSVFile(object sender, RoutedEventArgs e)

{

Button uploadFileBtn = sender as Button;

uploadFileBtn.IsEnabled = false;

try

{

Process.SetInitialStatus("Открытие файла");

CSVService.UpdateFilePath(filePathTextBox.Text);

IFileService<TunnelExit> csvService = CSVService.CSVServiceObject;

Process.UpdateStatus("Чтение данных");

var tunnelsList = await csvService.ReadAsync();

Process.UpdateStatus("Отображение данных");

await Task.Run(() => MainPageModel.PageModel.CreateNewTunnelData(tunnelsList));

await MainPageModel.UploadDelegate();

int damagedRecNum = MainPageModel.PageModel.NumberOfDamagedRecords;

Process.SetFinalStatus($"Файл загружен\n" + ((damagedRecNum == 0) ? string.Empty

: $"Кол-во повреждений: {damagedRecNum}"), true);

}

catch (Exception ex)

{

Process.SetFinalStatus("Файл не загружен", false);

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при загрузке файла");

}

finally

{

uploadFileBtn.IsEnabled = true;

}

}

private void SelectFile(object sender, RoutedEventArgs e)

{

try

{

OpenFileDialog openFileDialog = new OpenFileDialog()

{

Filter = "CSV files(\*csv)|\*csv",

Title = "Выберете CSV файл для открытия"

};

if (openFileDialog.ShowDialog() != true)

return;

filePathTextBox.Text = openFileDialog.FileName;

}

catch (NullReferenceException ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при выборе файла");

}

catch (Exception ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при выборе файла");

}

}

}

}

<UserControl x:Class="AeroViewer.Loader"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

Width="30"

Height="30">

<UserControl.Resources>

<Style x:Key="ellipseLoader" TargetType="Ellipse">

<Setter Property="Width" Value="5"/>

<Setter Property="Height" Value="5"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="Fill" Value="Gray"/>

</Style>

<DoubleAnimationUsingKeyFrames x:Key="ellipse1Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.250"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.750"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.125"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse2Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.250"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.75"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.25"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse3Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.750"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.25"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.375"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse4Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.75"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.250"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.5"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse5Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.75"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.25"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.625"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse6Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.75"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.25"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.75"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse7Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.25"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.75"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="0.875"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames x:Key="ellipse8Animation"

AccelerationRatio="0"

RepeatBehavior="Forever"

SpeedRatio="1.1">

<EasingDoubleKeyFrame KeyTime="0:0:0"

Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.25"

Value="0.125"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.5"

Value="0.25"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.75"

Value="0.375"/>

<EasingDoubleKeyFrame KeyTime="0:0:1"

Value="0.5"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.25"

Value="0.625"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5"

Value="0.750"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.75"

Value="0.875"/>

<EasingDoubleKeyFrame KeyTime="0:0:2"

Value="1"/>

</DoubleAnimationUsingKeyFrames>

</UserControl.Resources>

<Grid VerticalAlignment="Top"

HorizontalAlignment="Left"

Height="30"

Width="30">

<Ellipse x:Name="ellipse1"

Style="{StaticResource ellipseLoader}"

Margin="0, 0, 0, 24"/>

<Ellipse x:Name="ellipse2"

Style="{StaticResource ellipseLoader}"

Margin="20,4,4,20"/>

<Ellipse x:Name="ellipse3"

Style="{StaticResource ellipseLoader}"

Margin="24,12,0,12"/>

<Ellipse x:Name="ellipse4"

Style="{StaticResource ellipseLoader}"

Margin="20,20,4,4"/>

<Ellipse x:Name="ellipse5"

Style="{StaticResource ellipseLoader}"

Margin="12,24,12,0"/>

<Ellipse x:Name="ellipse6"

Style="{StaticResource ellipseLoader}"

Margin="4,20,20,4"/>

<Ellipse x:Name="ellipse7"

Style="{StaticResource ellipseLoader}"

Margin="0,12,24,12"/>

<Ellipse x:Name="ellipse8"

Style="{StaticResource ellipseLoader}"

Margin="4,4,20,20"/>

</Grid>

</UserControl>

using System.Windows.Controls;

using System.Windows.Media.Animation;

namespace AeroViewer

{

public partial class Loader : UserControl

{

public Loader()

{

InitializeComponent();

ellipse1.BeginAnimation(OpacityProperty,

Resources["ellipse1Animation"] as DoubleAnimationUsingKeyFrames);

ellipse2.BeginAnimation(OpacityProperty,

Resources["ellipse2Animation"] as DoubleAnimationUsingKeyFrames);

ellipse3.BeginAnimation(OpacityProperty,

Resources["ellipse3Animation"] as DoubleAnimationUsingKeyFrames);

ellipse4.BeginAnimation(OpacityProperty,

Resources["ellipse4Animation"] as DoubleAnimationUsingKeyFrames);

ellipse5.BeginAnimation(OpacityProperty,

Resources["ellipse5Animation"] as DoubleAnimationUsingKeyFrames);

ellipse6.BeginAnimation(OpacityProperty,

Resources["ellipse6Animation"] as DoubleAnimationUsingKeyFrames);

ellipse7.BeginAnimation(OpacityProperty,

Resources["ellipse7Animation"] as DoubleAnimationUsingKeyFrames);

ellipse8.BeginAnimation(OpacityProperty,

Resources["ellipse8Animation"] as DoubleAnimationUsingKeyFrames);

}

}

}

<Page x:Class="AeroViewer.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

d:DesignHeight="450"

d:DesignWidth="800"

Title="{Binding FullDocumentName}">

<Page.Resources>

<Style x:Key="propertyValueTextBlockStyle" TargetType="TextBlock">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="18"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="FontWeight" Value="ExtraLight"/>

</Style>

<Style x:Key="outterBorderStyle" TargetType="Border">

<Setter Property="Background" Value="White"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="CornerRadius" Value="10"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="BorderBrush" Value="{StaticResource backgroundBrush}"/>

</Style>

<Style x:Key="propertyNameTextBlockStyle" TargetType="TextBlock">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="12"/>

<Setter Property="Foreground" Value="{StaticResource hintBrushText}"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="FontWeight" Value="DemiBold"/>

</Style>

<Style x:Key="devisionRectanleStyle" TargetType="Rectangle">

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Height" Value="1"/>

<Setter Property="Fill" Value="{StaticResource hintBrushText}"/>

</Style>

<Style x:Key="dataGridStyle" TargetType="DataGrid">

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Height" Value="500"/>

<Setter Property="Background" Value="Transparent"/>

<Setter Property="RowBackground" Value="White"/>

<Setter Property="AlternatingRowBackground" Value="#d6ffd3"/>

<Setter Property="IsReadOnly" Value="False"/>

<Setter Property="GridLinesVisibility" Value="None"/>

<Setter Property="HeadersVisibility" Value="Column"/>

<Setter Property="MinRowHeight" Value="30"/>

<Setter Property="VerticalScrollBarVisibility" Value="Visible"/>

<Setter Property="CanUserDeleteRows" Value="False"/>

<Setter Property="CanUserResizeColumns" Value="False"/>

<Setter Property="CanUserSortColumns" Value="True"/>

<Setter Property="CanUserReorderColumns" Value="True"/>

<Setter Property="CanUserResizeColumns" Value="True"/>

<Setter Property="CanUserAddRows" Value="False"/>

<Setter Property="Visibility" Value="Collapsed"/>

</Style>

<Style x:Key="dataGridHeaderStyle" TargetType="DataGridColumnHeader">

<Setter Property="Background" Value="{StaticResource darkMainThemeBrush}"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="BorderThickness" Value="0, 0, 0.5, 0"/>

<Setter Property="BorderBrush" Value="White"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="Padding" Value="10"/>

<Setter Property="FontWeight" Value="Light"/>

<Setter Property="IsEnabled" Value="True"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="ToolTip" Value="Нажмите для сортировки"/>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="dataGridCellStyle" TargetType="DataGridCell">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="Foreground" Value="Gray"/>

<Setter Property="FontStyle" Value="Normal"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type DataGridCell}">

<Border x:Name="border"

BorderBrush="LightGray"

BorderThickness="0, 0, 0.5, 0"

Background="Transparent"

SnapsToDevicePixels="True">

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetName="border"

Storyboard.TargetProperty="(Border.Background).(SolidColorBrush.Color)"

To="#F0FFE9"

SpeedRatio="100"/>

</Storyboard>

</VisualState>

<VisualState x:Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetName="border"

Storyboard.TargetProperty="(Border.Background).(SolidColorBrush.Color)"

To="Transparent"

SpeedRatio="100"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

<VisualStateGroup x:Name="FocusStates">

<VisualState x:Name="Unfocused" />

<VisualState x:Name="Focused" />

</VisualStateGroup>

<VisualStateGroup x:Name="CurrentStates">

<VisualState x:Name="Regular" />

<VisualState x:Name="Current"/>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

<ContentPresenter SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"

VerticalAlignment="Center"

HorizontalAlignment="Stretch"

Margin="5">

</ContentPresenter>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="filterComboBoxStyle" TargetType="ComboBox">

<Setter Property="BorderBrush" Value="{StaticResource darkMainThemeBrush}"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Width" Value="70"/>

<Setter Property="Height" Value="25"/>

<Setter Property="Foreground" Value="Gray"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Padding" Value="3, 5, 0, 0"/>

<Setter Property="SelectedIndex" Value="0"/>

</Style>

</Page.Resources>

<Grid x:Name="backgroundGrid">

<Border x:Name="outterBorder"

Style="{StaticResource outterBorderStyle}">

<Grid>

<Grid x:Name="topOptionsGrid">

<Button x:Name="saveChangesBtn"

Style="{StaticResource appDefaultButtonStyle}"

Margin="20, 15, 0, 0"

Click="SaveChanges">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="7, 0, 0, 2"

Source="../Resources/saveChangesIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="20, 0, 0, 0"

Text="Сохранить"/>

</Grid>

</Button.Content>

</Button>

<TextBlock Style="{StaticResource propertyNameTextBlockStyle}"

Text="Кол-во записей:"

Margin="20, 55, 0, 0"/>

<TextBox x:Name="numberOfVisibleRecordsTextBox"

Style="{StaticResource appTextBoxStyle}"

Width="110"

Margin="20, 70, 0, 0"

TextChanged="UpdateNumberOfVisibleRecords"/>

<Button x:Name="addNewRecordBtn"

Style="{StaticResource appDefaultButtonStyle}"

Margin="160, 70, 0, 0"

Width="100"

Click="AddNewRecord">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="7, 0, 0, 1"

Source="../Resources/addRecordIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="15, 0, 0, 0"

Text="Добавить"/>

</Grid>

</Button.Content>

</Button>

<Button x:Name="deleteRecordBtn"

Style="{StaticResource appDangerousBtnStyle}"

Margin="270, 70, 0, 0"

Width="100"

Click="DeleteRecordsAsync">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="10, 0, 0, 2"

Source="../Resources/deleteRecordsIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="18, 0, 0, 0"

Text="Удалить"/>

</Grid>

</Button.Content>

</Button>

<TextBlock Style="{StaticResource propertyNameTextBlockStyle}"

Text="Фильтр: "

Margin="380, 55, 0, 0"/>

<TextBox x:Name="filterTextBox"

Style="{StaticResource appTextBoxStyle}"

Margin="380, 70, 0, 0"

Width="140"

TextChanged="FindAllRecords"/>

<ComboBox x:Name="filterComboBox"

Margin="530, 72, 0, 0"

Style="{StaticResource filterComboBoxStyle}">

<TextBlock>Все</TextBlock>

<TextBlock>Округ</TextBlock>

<TextBlock>Global ID</TextBlock>

</ComboBox>

<Rectangle x:Name="topDevisionRectangle"

Style="{StaticResource devisionRectanleStyle}"

Margin="20, 110, 0, 0"/>

</Grid>

<Grid x:Name="dataBackgroundGrid"

Margin="20, 120, 20, 0">

<DataGrid x:Name="csvDataGrid"

ItemsSource="{Binding TunnelsData}"

Style="{StaticResource dataGridStyle}"

ColumnHeaderStyle="{StaticResource dataGridHeaderStyle}"

CellStyle="{StaticResource dataGridCellStyle}">

</DataGrid>

</Grid>

</Grid>

</Border>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using AeroViewer.ViewModels;

using AeroViewer.Attributes;

using System.Reflection;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Windows.Data;

namespace AeroViewer

{

public partial class MainPage : Page

{

#region Properties

public Frame ParentFrame { get; set; }

public DataGrid DataGrid { get; }

#endregion

public MainPage(Frame parentFrame)

{

InitializeComponent();

ParentFrame = parentFrame;

ParentFrame.SizeChanged += OnParentFrameSizeChanged;

DataGrid = csvDataGrid;

DataGrid.AutoGeneratedColumns += EditAutoGeneratedColumns<TunnelExitModel>;

DataGrid.Sorting += SortDataGrid<TunnelExitModel>;

MainPageModel.UploadDelegate += UploadData;

}

/// <summary>

/// Uploads the data from CSV file to UI and changes the headers of all columns

/// This method is called when the MainPageModel has changed, through UploadDelegate

/// </summary>

public async Task UploadData()

{

await Task.Run(async () =>

{

await Dispatcher.BeginInvoke(new Action(() =>

{

csvDataGrid.Visibility = Visibility.Visible;

DataContext = null;

DataContext = MainPageModel.PageModel;

MainWindow.Window.Title = MainPageModel.PageModel.FullDocumentName;

csvDataGrid.ItemsSource = MainPageModel.PageModel.TunnelsData;

}));

});

}

#region Columns edition

/// <summary>

/// Changes the headers of all columns to headers which were defined in the attributes.

/// Also makes sets width of a first column (selection column) to 30, and makes the last column

/// (upload status) readonly

/// </summary>

/// <typeparam name="T">ViewModel type</typeparam>

private void EditAutoGeneratedColumns<T>(object sender, EventArgs e)

where T : TunnelExitModel

{

DataGrid dataGrid = sender as DataGrid;

PropertyInfo[] properties = typeof(T).GetProperties(BindingFlags.Instance

| BindingFlags.Public);

//change width of the selection column (the first one)

dataGrid.Columns[0].Width = 30;

//Load status column must be readonly

dataGrid.Columns[csvDataGrid.Columns.Count - 1].IsReadOnly = true;

for (int i = 0; i < properties.Length; i++)

{

dataGrid.Columns[i].Header =

properties[i].GetCustomAttribute<HeaderNameAttribute>().HeaderName;

}

}

/// <summary>

/// Customly sorts the AdmArea column basing on the amount of distrcits in each AdmArea

/// </summary>

/// <typeparam name="T">The data grid model type</typeparam>

private void SortDataGrid<T>(object sender, DataGridSortingEventArgs e)

where T : TunnelExitModel

{

try

{

DataGrid dataGrid = sender as DataGrid;

DataGridColumn sortedColumn = e.Column;

if (sortedColumn.Header.ToString() == typeof(T).GetProperty("AdmArea").GetCustomAttribute

<HeaderNameAttribute>().HeaderName)

{

e.Handled = true;

ListSortDirection sortDirection = (sortedColumn.SortDirection == ListSortDirection.Ascending) ?

ListSortDirection.Descending : ListSortDirection.Ascending;

sortedColumn.SortDirection = sortDirection;

ListCollectionView dataList = (ListCollectionView)CollectionViewSource.

GetDefaultView(dataGrid.ItemsSource);

dataList.CustomSort = new AdmAreaSort<T>(sortDirection);

}

}

catch (NullReferenceException ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex, "Ошибка");

}

catch (Exception ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex, "Ошибка");

}

}

#endregion

#region Event handlers

private void OnParentFrameSizeChanged(object sender, EventArgs e)

{

Width = ParentFrame.Width;

Height = ParentFrame.Height;

outterBorder.Width = Width;

outterBorder.Height = Height;

topDevisionRectangle.Width = Width - 2 \* topDevisionRectangle.Margin.Left;

}

#endregion

#region Save/Delete/Create/Filter items operations

private async void UpdateNumberOfVisibleRecords(object sender, EventArgs e)

{

try

{

Process.SetInitialStatus("Пересчет элементов");

TextBox itemsCountTextBox = sender as TextBox;

int visibleItemsCount;

if (itemsCountTextBox.Text == string.Empty)

visibleItemsCount = MainPageModel.PageModel.TunnelsData.Count;

else

visibleItemsCount = int.Parse(itemsCountTextBox.Text);

if (visibleItemsCount > MainPageModel.PageModel.TunnelsData.Count)

throw new IndexOutOfRangeException("Неверное число записей");

Process.UpdateStatus("Создание списка");

ObservableCollection<TunnelExitModel> visibleItems =

new ObservableCollection<TunnelExitModel>();

await Task.Run(() =>

{

for (int i = 0; i < visibleItemsCount; i++)

visibleItems.Add(MainPageModel.PageModel.TunnelsData[i]);

});

csvDataGrid.ItemsSource = visibleItems;

Process.SetFinalStatus("Записи отображены", true);

}

catch (IndexOutOfRangeException)

{

Process.SetFinalStatus("Ошибка отображения", false);

}

catch (Exception)

{

Process.SetFinalStatus("Ошибка отображения", false);

}

}

private void SaveChanges(object sender, RoutedEventArgs e)

{

try

{

SaveWindow saveDataWindow = new SaveWindow(csvDataGrid.ItemsSource

as ObservableCollection<TunnelExitModel>);

saveDataWindow.ShowDialog();

}

catch (ArgumentException ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при вызове диалога сохранения");

}

catch (NullReferenceException ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при вызове диалога сохранения");

}

catch (Exception ex)

{

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex,

"Ошибка при вызове диалога сохранения");

}

}

/// <summary>

/// Deletes all selected records from the list

/// </summary>

private async void DeleteRecordsAsync(object sender, EventArgs e)

{

await Task.Run(async () =>

{

List<TunnelExitModel> selectedItems = GetSelectedItems().ToList();

if (selectedItems.Count == 0)

{

Dispatcher.Invoke(() => AeroViewerMessageBox.ShowMessageBox("Ошибка удаления", "Вы выбрали 0 записей",

MessageBoxButton.OK));

return;

}

foreach (TunnelExitModel tunnelExit in selectedItems)

await Dispatcher.BeginInvoke(new Action(() =>

MainPageModel.PageModel.TunnelsData.Remove(tunnelExit)));

await Dispatcher.BeginInvoke(new Action(() =>

{

csvDataGrid.ItemsSource = MainPageModel.PageModel.TunnelsData;

numberOfVisibleRecordsTextBox.Text = string.Empty;

}));

});

}

private IEnumerable<TunnelExitModel> GetSelectedItems() =>

MainPageModel.PageModel.TunnelsData.Where((t) => t.IsSelected == true);

public void AddNewRecord(object sender, EventArgs e)

{

csvDataGrid.ItemsSource = MainPageModel.PageModel.TunnelsData;

filterTextBox.Text = string.Empty;

numberOfVisibleRecordsTextBox.Text = string.Empty;

TunnelExitModel tunnelExitModel = new TunnelExitModel();

if (MainPageModel.PageModel.TunnelsData.Count != 0)

{

tunnelExitModel.RowNum = MainPageModel.PageModel.TunnelsData[

MainPageModel.PageModel.TunnelsData.Count - 1].RowNum + 1;

}

tunnelExitModel.IsDamaged = "OK";

MainPageModel.PageModel.TunnelsData.Add(tunnelExitModel);

csvDataGrid.ScrollIntoView(tunnelExitModel);

}

#region Filter methods

/// <summary>

/// Finds all records which satisfies the given filter condition and updates the UI

/// in real time

/// </summary>

private async void FindAllRecords(object sender, TextChangedEventArgs e)

{

try

{

PropertyInfo[] propertyInfos = typeof(TunnelExitModel).GetProperties(

BindingFlags.Instance | BindingFlags.Public);

string filterText = filterTextBox.Text;

int selectionMode = filterComboBox.SelectedIndex;

await Task.Run(async () =>

{

Process.SetInitialStatus("Поиск записей");

ObservableCollection<TunnelExitModel> suitableObjects =

GetAllSuitableRecords(selectionMode, propertyInfos, filterText);

Process.UpdateStatus("Отображение записей");

await Dispatcher.BeginInvoke(new Action(() =>

csvDataGrid.ItemsSource = suitableObjects));

Process.SetFinalStatus("Записи отображены", true);

});

}

catch (NullReferenceException ex)

{

Process.SetFinalStatus("Ошибка при поиске", false);

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex, "Ошибка при поиске записей");

}

catch (Exception ex)

{

Process.SetFinalStatus("Ошибка при поиске", false);

ExceptionHandler.Handler.HandleExceptionWithMessageBox(ex, "Ошибка при поиске записей");

}

}

private ObservableCollection<TunnelExitModel> GetAllSuitableRecords(

int selectionMode, PropertyInfo[] propertyInfos, string filterText)

{

ObservableCollection<TunnelExitModel> suitableObjects =

new ObservableCollection<TunnelExitModel>();

switch (selectionMode)

{

case 0:

suitableObjects = CreateTunnelExitsList(string.Empty, propertyInfos,

filterText);

break;

case 1:

suitableObjects = CreateTunnelExitsList("AdmArea", propertyInfos,

filterText);

break;

case 2:

suitableObjects = CreateTunnelExitsList("TunnelGlobalID", propertyInfos,

filterText);

break;

}

return suitableObjects;

}

/// <summary>

/// Creates an observable collection of TunnelExitModel objects which satisfies the given filter

/// condition

/// </summary>

private ObservableCollection<TunnelExitModel> CreateTunnelExitsList(

string selectedProperty, PropertyInfo[] propertyInfos, string filterText)

{

ObservableCollection<TunnelExitModel> suitableObjects =

new ObservableCollection<TunnelExitModel>();

foreach (TunnelExitModel tunnelExitModel in MainPageModel.PageModel.TunnelsData)

{

foreach (PropertyInfo propertyInfo in propertyInfos)

{

if (propertyInfo.GetValue(tunnelExitModel).ToString().IndexOf(filterText) > -1

&& propertyInfo.GetCustomAttribute<CustomPropertyAttribute>() == null

&& CheckProperty(selectedProperty, propertyInfo))

{

suitableObjects.Add(tunnelExitModel);

break;

}

}

}

return suitableObjects;

}

/// <summary>

/// Checks if the property satisfies current filter conditions

/// </summary>

private bool CheckProperty(string selectedProperty, PropertyInfo propertyInfo)

{

if (selectedProperty == string.Empty)

return true;

if (selectedProperty == propertyInfo.Name)

return true;

return false;

}

#endregion

#endregion

}

}

<Window x:Class="AeroViewer.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

MinWidth="900"

MinHeight="450"

Width="900"

Height="450"

Title="AeroViewer"

Background="{StaticResource hintBrushText}"

SizeChanged="RepositionElements">

<Window.Resources>

<Style x:Key="leftGridStyle" TargetType="Grid">

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Height" Value="450"/>

<Setter Property="Width" Value="250"/>

<Setter Property="Margin" Value="0"/>

<Setter Property="Background" Value="{StaticResource hintBrushText}"/>

</Style>

<Style x:Key="leftFrameStyle" TargetType="Frame">

<Setter Property="Width" Value="250"/>

<Setter Property="Height" Value="450"/>

<Setter Property="Background" Value="{StaticResource hintBrushText}"/>

<Setter Property="NavigationUIVisibility" Value="Hidden"/>

</Style>

<Style x:Key="mainGridStyle" TargetType="Grid">

<Setter Property="Height" Value="450"/>

<Setter Property="Width" Value="650"/>

<Setter Property="Margin" Value="250, 0, 0, 0"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="Background" Value="{StaticResource hintBrushText}"/>

</Style>

<Style x:Key="mainFrameStyle" TargetType="Frame">

<Setter Property="Width" Value="650"/>

<Setter Property="Height" Value="450"/>

<Setter Property="Background" Value="{StaticResource hintBrushText}"/>

<Setter Property="NavigationUIVisibility" Value="Hidden"/>

<Setter Property="Padding" Value="0"/>

</Style>

</Window.Resources>

<Grid x:Name="mainBackgroundGrid"

VerticalAlignment="Top"

HorizontalAlignment="Left"

Width="900"

Height="450">

<Grid x:Name="leftGrid"

Style="{StaticResource leftGridStyle}">

<Frame x:Name="menuFrame"

Style="{StaticResource leftFrameStyle}"/>

</Grid>

<Grid x:Name="mainGrid"

Style="{StaticResource mainGridStyle}">

<Frame x:Name="mainFrame"

Style="{StaticResource mainFrameStyle}"/>

</Grid>

</Grid>

</Window>

using System.Globalization;

using System.Threading;

using System.Windows;

namespace AeroViewer

{

public partial class MainWindow : Window

{

public static MainWindow Window { get; set; }

public MainWindow()

{

InitializeComponent();

Window = this;

Position.UpdateMainWindow(this);

menuFrame.Content = new LeftMenuPage(menuFrame);

mainFrame.Content = new MainPage(mainFrame);

}

#region Size changed methods

private void RepositionElements(object sender, SizeChangedEventArgs e)

{

if (WindowState == WindowState.Maximized)

Position.UpdateMainWindow(this);

Position.MainWindowPosition.RepositionMainWindowElemets();

}

#endregion

}

}

<Window x:Class="AeroViewer.SaveWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:AeroViewer"

mc:Ignorable="d"

Title="Сохранить файл"

Height="200"

MaxHeight="200"

Width="450"

MaxWidth="450">

<Window.Resources>

<Style x:Key="saveFileTextBoxStylePassive" TargetType="TextBox">

<Setter Property="OverridesDefaultStyle" Value="False"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="15"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Width" Value="220"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Padding" Value="0, 7, 0, 0"/>

<Setter Property="CaretBrush" Value="Black"/>

<Setter Property="Margin" Value="10, 145, 0, 0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="TextBox">

<Border BorderThickness="1.5, 1.5, 1.5, 1.5"

BorderBrush="{StaticResource darkMainThemeBrush}"

CornerRadius="5"

SnapsToDevicePixels="True">

<ScrollViewer Name="PART\_ContentHost"

Background="White"

Width="390"

Height="30"

HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="saveFileTextBoxStyleActive" TargetType="TextBox">

<Setter Property="OverridesDefaultStyle" Value="False"/>

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="15"/>

<Setter Property="Foreground" Value="{StaticResource defaultTextBrush}"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Width" Value="220"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Padding" Value="0, 7, 0, 0"/>

<Setter Property="CaretBrush" Value="Black"/>

<Setter Property="Margin" Value="10, 145, 0, 0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="TextBox">

<Border BorderThickness="1.5, 1.5, 1.5, 1.5"

BorderBrush="{StaticResource lightMainThemeBrush}"

CornerRadius="5"

SnapsToDevicePixels="True">

<ScrollViewer Name="PART\_ContentHost"

Background="White"

Width="390"

Height="30"

HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="descriptionTextBlockStyle" TargetType="TextBlock">

<Setter Property="FontFamily" Value="Arial, Verdana"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="Foreground" Value="Gray"/>

<Setter Property="FontWeight" Value="Light"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="TextAlignment" Value="Left"/>

<Setter Property="TextWrapping" Value="Wrap"/>

</Style>

</Window.Resources>

<Grid VerticalAlignment="Top"

HorizontalAlignment="Left"

Width="400"

Height="300">

<TextBox x:Name="filePathTextBox"

Style="{StaticResource saveFileTextBoxStylePassive}"

Margin="25,70,-30.667,0"

Width="405"/>

<Button Style="{StaticResource appDefaultButtonStyle}"

Click="SaveFile"

Margin="320,121,-30.667,0">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="23"

Width="23"

Margin="8, 0, 0, 2"

Source="../Resources/saveChangesIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="20, 0, 0, 2"

Text="Сохранить"/>

</Grid>

</Button.Content>

</Button>

<Button Style="{StaticResource appDangerousBtnStyle}"

Click="CancelSavingProcess"

Width="100"

Margin="205,121,0,0">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="20"

Width="20"

Margin="13, 0, 0, 2"

Source="../Resources/cancelActionIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="17, 0, 0, 2"

Text="Отмена"/>

</Grid>

</Button.Content>

</Button>

<Button Style="{StaticResource appDefaultButtonStyle}"

Click="ChooseFilePath"

Width="100"

Margin="25,121,0,0">

<Button.Content>

<Grid>

<Image VerticalAlignment="Center"

HorizontalAlignment="Left"

Height="20"

Width="20"

Margin="15, 0, 0, 2"

Source="../Resources/uploadCSVButtonIcon.png"/>

<TextBlock VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="10, 0, 0, 2"

Text="Обзор"/>

</Grid>

</Button.Content>

</Button>

<RadioButton x:Name="rewriteFileRadioBtn"

Content="Перезаписать файл"

HorizontalAlignment="Left"

Margin="31,13,0,0"

IsChecked="True"

VerticalAlignment="Top"/>

<RadioButton x:Name="appendFileRadioBtn"

Content="Добавить в конец файла"

HorizontalAlignment="Left"

Margin="31,33,0,0"

IsChecked="False"

VerticalAlignment="Top"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Collections.ObjectModel;

using AeroViewer.Models;

using AeroViewer.ViewModels;

using System.IO;

using AeroViewer.Services;

using AeroViewer.Interfaces;

using Microsoft.Win32;

namespace AeroViewer

{

/// <summary>

/// Логика взаимодействия для SaveWindow.xaml

/// </summary>

public partial class SaveWindow : Window

{

#region Properties

private List<TunnelExit> TunnelExits { get; set; }

#endregion

#region Constrcutors

public SaveWindow() { InitializeComponent(); }

public SaveWindow(ObservableCollection<TunnelExitModel> tunnelExitModels)

{

if (tunnelExitModels == null || tunnelExitModels.Count == 0)

throw new ArgumentException("Нет данных для сохранения");

InitializeComponent();

filePathTextBox.Text = CSVService.CSVServiceObject.Database.FilePath;

TunnelExits = CreateTunnelExitsList(tunnelExitModels);

}

#endregion

private List<TunnelExit> CreateTunnelExitsList(

ObservableCollection<TunnelExitModel> tunnelExitModels)

{

List<TunnelExit> tunnelExits = new List<TunnelExit>();

foreach (TunnelExitModel tunnelExitModel in tunnelExitModels)

tunnelExits.Add(new TunnelExit(tunnelExitModel));

return tunnelExits;

}

#region Save file methods

/// <summary>

/// Saves the changes in CSV file in a mode which was selected by user

/// (Rewrite, append w/o header string, append w header string)

/// </summary>

private async void SaveFile(object sender, EventArgs e)

{

string filePath = filePathTextBox.Text;

try

{

Close();

Process.SetInitialStatus("Открываем файл");

CSVService.UpdateFilePath(filePathTextBox.Text);

IFileService<TunnelExit> csvService = CSVService.CSVServiceObject;

Process.UpdateStatus("Записываем в файл");

if (rewriteFileRadioBtn.IsChecked == true)

await csvService.RewriteAsync(TunnelExits);

else

await csvService.AppendAsync(TunnelExits, false);

Process.SetFinalStatus("Файл перезаписан", true);

}

catch (UnauthorizedAccessException ex)

{

HandleSaveFileException(ex);

}

catch (IOException ex)

{

HandleSaveFileException(ex);

}

catch (Exception ex)

{

HandleSaveFileException(ex);

}

}

private void HandleSaveFileException(Exception ex)

{

Process.SetFinalStatus("Ошибка записи", false);

ExceptionHandler.Handler.HandleExceptionWithMessageBox(

ex, "Ошибка при сохранении файла");

}

private void ChooseFilePath(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog()

{

Title = "Выберете файл или папку",

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

};

if (saveFileDialog.ShowDialog() == false)

return;

filePathTextBox.Text = saveFileDialog.FileName;

if (filePathTextBox.Text.Substring(filePathTextBox.Text.Length - 4) != ".csv")

filePathTextBox.Text += ".csv";

}

private void CancelSavingProcess(object sender, EventArgs e) => Close();

#endregion

}

}

using System.Collections.Generic;

using System.Threading.Tasks;

using AeroViewer.Data;

using AeroViewer.Interfaces;

using AeroViewer.Models;

namespace AeroViewer.Services

{

/// <summary>

/// Service which works with CSV files

/// </summary>

public class CSVService : IFileService<TunnelExit>

{

#region Properties

public Database Database { get; } = new Database();

private static CSVService csvService;

public static CSVService CSVServiceObject

{

get

{

if (csvService == null)

csvService = new CSVService();

return csvService;

}

}

#endregion

#region Constructors

private CSVService() { }

#endregion

public static void UpdateFilePath(string filePath) =>

CSVServiceObject.Database.FilePath = filePath;

#region IFileService implementation

public async Task<List<TunnelExit>> ReadAsync() =>

await Database.ReadFileDataAsync();

public async Task RewriteAsync(List<TunnelExit> tunnelExits) =>

await Database.RewriteDataAsync(tunnelExits);

public async Task AppendAsync(List<TunnelExit> tunnelExits, bool? addHeaderString) =>

await Database.AppendDataAsync(tunnelExits, addHeaderString);

#endregion

}

}

using System.Collections;

using System.ComponentModel;

namespace AeroViewer.ViewModels

{

public class AdmAreaSort<T> : IComparer

where T : TunnelExitModel

{

private ListSortDirection ListSortDirection { get; }

public AdmAreaSort(ListSortDirection listSortDirection) =>

ListSortDirection = listSortDirection;

public int Compare(object x, object y)

{

T xT = x as T;

T yT = y as T;

if (!(MainPageModel.PageModel.DistrictCountDictionary.ContainsKey(xT.AdmArea) &&

MainPageModel.PageModel.DistrictCountDictionary.ContainsKey(yT.AdmArea)))

return 0;

if (MainPageModel.PageModel.DistrictCountDictionary[xT.AdmArea].Count >

MainPageModel.PageModel.DistrictCountDictionary[yT.AdmArea].Count)

{

return -1 \* ((ListSortDirection == ListSortDirection.Descending) ? -1 : 1);

}

if (MainPageModel.PageModel.DistrictCountDictionary[xT.AdmArea].Count <

MainPageModel.PageModel.DistrictCountDictionary[yT.AdmArea].Count)

{

return 1 \* ((ListSortDirection == ListSortDirection.Descending) ? -1 : 1);

}

return 0;

}

}

}

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Collections.ObjectModel;

using AeroViewer.Models;

using AeroViewer.Services;

namespace AeroViewer.ViewModels

{

public delegate Task UploadDelegate();

public class MainPageModel

{

#region View Model Properties

public string ShortDocumentName { get; set; } = "Название документа";

public string FullDocumentName { get; set; } = "Название документа";

public ObservableCollection<TunnelExitModel> TunnelsData { get; set; }

#endregion

#region Usual properties

/// <summary>

/// Delegate which runs after creating a list of TunnelExitModels,

/// basicly the function of this delegate is to upload the data to the UI

/// </summary>

public static UploadDelegate UploadDelegate { get; set; }

public int NumberOfDamagedRecords { get; set; } = 0;

public Dictionary<string, List<string>> DistrictCountDictionary { get; set; }

#endregion

#region Constructors

private MainPageModel() =>

TunnelsData = new ObservableCollection<TunnelExitModel>();

#endregion

/// <summary>

/// Creates a list of TunnelExitModel objets based on the TunnelExit list

/// </summary>

/// <returns>The number of damaged records</returns>

public void CreateNewTunnelData(List<TunnelExit> tunnelExitsList)

{

InitializePropeties();

foreach (TunnelExit tunnelExit in tunnelExitsList)

{

if (tunnelExit.IsDamaged == "Damaged")

NumberOfDamagedRecords++;

else

TunnelsData.Add(new TunnelExitModel(tunnelExit));

if (DistrictCountDictionary.ContainsKey(tunnelExit.AdmArea))

{

if (DistrictCountDictionary[tunnelExit.AdmArea].FindIndex(d => d == tunnelExit.District) == -1)

DistrictCountDictionary[tunnelExit.AdmArea].Add(tunnelExit.District);

}

else

{

DistrictCountDictionary[tunnelExit.AdmArea] = new List<string>();

DistrictCountDictionary[tunnelExit.AdmArea].Add(tunnelExit.District);

}

}

}

private void InitializePropeties()

{

PageModel.TunnelsData = new ObservableCollection<TunnelExitModel>();

NumberOfDamagedRecords = 0;

DistrictCountDictionary = new Dictionary<string, List<string>>();

PageModel.FullDocumentName = CSVService.CSVServiceObject.Database.FilePath;

PageModel.ShortDocumentName = PageModel.FullDocumentName.Substring(

PageModel.FullDocumentName.LastIndexOf("\\") + 1);

}

#region Singleton

private static MainPageModel mainPageModel;

public static MainPageModel PageModel

{

get

{

if (mainPageModel == null)

mainPageModel = new MainPageModel();

return mainPageModel;

}

}

#endregion

}

}

using AeroViewer.Attributes;

using AeroViewer.Models;

namespace AeroViewer.ViewModels

{

public class TunnelExitModel

{

[HeaderName("")]

[CustomProperty]

public bool IsSelected { get; set; }

[HeaderName("Номер")]

public int RowNum { get; set; }

[HeaderName("ID")]

public long ID { get; set; }

[HeaderName("Имя")]

public string Name { get; set; } = string.Empty;

[HeaderName("GlobalID туннеля")]

public long TunnelGlobalID { get; set; }

[HeaderName("Описание туннеля")]

public string TunnelDescription { get; set; } = string.Empty;

[HeaderName("Округ")]

public string AdmArea { get; set; } = string.Empty;

[HeaderName("Район")]

public string District { get; set; } = string.Empty;

[HeaderName("Широта")]

public double Longitude { get; set; }

[HeaderName("Долгота")]

public double Latitude { get; set; }

[HeaderName("Статус загрузки")]

[CustomProperty]

public string IsDamaged { get; set; } = string.Empty;

public TunnelExitModel() { }

public TunnelExitModel(TunnelExit tunnelExit)

{

IsSelected = false;

RowNum = tunnelExit.RowNum;

ID = tunnelExit.ID;

Name = tunnelExit.Name;

TunnelGlobalID = tunnelExit.Tunnel.GlobalID;

TunnelDescription = tunnelExit.Tunnel.Name;

AdmArea = tunnelExit.AdmArea;

District = tunnelExit.District;

Longitude = tunnelExit.Longitude;

Latitude = tunnelExit.Latitude;

IsDamaged = tunnelExit.IsDamaged;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Threading;

namespace AeroViewer

{

/// <summary>

/// Class which handles all the exceptions which can occur during runtime

/// </summary>

public class ExceptionHandler

{

public Dispatcher Dispatcher { get; private set; }

#region Singleton

private static ExceptionHandler exceptionHandler;

public static ExceptionHandler Handler

{

get

{

if (exceptionHandler == null)

exceptionHandler = new ExceptionHandler();

return exceptionHandler;

}

}

#endregion

#region Constructors

private ExceptionHandler() { }

#endregion

public void HandleExceptionWithMessageBox(Exception ex, string title) =>

AeroViewerMessageBox.ShowMessageBox(title, ex.Message, System.Windows.MessageBoxButton.OK);

public static void UpdateDispatcher(Dispatcher dispatcher) =>

Handler.Dispatcher = dispatcher;

}

}

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace AeroViewer

{

internal class Position

{

#region Constants

private const double LeftGridWidth = 250;

private const double MainGridLeftMargin = 10;

#endregion

#region Singleton

private static Position mainWindowPosition;

public static Position MainWindowPosition

{

get

{

if (mainWindowPosition == null)

mainWindowPosition = new Position();

return mainWindowPosition;

}

}

#endregion

#region Properties

private MainWindow mainWindow;

private MainWindow MainWindow

{

get => mainWindow;

set

{

mainWindow = value;

SetMaximaziedParams();

}

}

private void SetMaximaziedParams()

{

MainWindow.Width = SystemParameters.MaximizedPrimaryScreenWidth;

MainWindow.Height = SystemParameters.MaximizedPrimaryScreenHeight;

MainWindow.Top = 0;

MainWindow.Left = 0;

}

#endregion

#region Constructors

private Position() { }

#endregion

public static void UpdateMainWindow(MainWindow mainWindow) =>

MainWindowPosition.MainWindow = mainWindow;

#region Size changed methods

public void RepositionMainWindowElemets()

{

RepositionGrids();

RepositionFrames();

}

private void RepositionGrids()

{

Grid mainGrid = GetMainGrid();

Grid backgroundGrid = GetBackGroundGrid();

Grid leftGrid = GetLeftGrid();

backgroundGrid.Width = MainWindow.Width -

(SystemParameters.MaximizedPrimaryScreenWidth - SystemParameters.FullPrimaryScreenWidth);

backgroundGrid.Height = MainWindow.Height -

(SystemParameters.MaximizedPrimaryScreenHeight - SystemParameters.FullPrimaryScreenHeight);

leftGrid.Height = backgroundGrid.Height;

mainGrid.Width = backgroundGrid.Width - LeftGridWidth - MainGridLeftMargin;

mainGrid.Height = backgroundGrid.Height;

mainGrid.Margin = new Thickness(LeftGridWidth + MainGridLeftMargin, 0, 0, 0);

}

private Grid GetLeftGrid()

{

Grid backgroundGrid = MainWindow.Content as Grid;

List<Grid> mainWindowGrids = backgroundGrid.Children.OfType<Grid>().ToList();

return mainWindowGrids.Find(g => g.Name == "leftGrid");

}

private Grid GetBackGroundGrid() => MainWindow.Content as Grid;

private Grid GetMainGrid()

{

Grid backgroundGrid = MainWindow.Content as Grid;

List<Grid> mainWindowGrids = backgroundGrid.Children.OfType<Grid>().ToList();

return mainWindowGrids.Find(g => g.Name == "mainGrid");

}

private void RepositionFrames()

{

Grid leftGrid, mainGrid;

Frame menuFrame = GetInnerFrame(leftGrid = GetLeftGrid());

Frame mainFrame = GetInnerFrame(mainGrid = GetMainGrid());

menuFrame.Width = leftGrid.Width;

menuFrame.Height = leftGrid.Height;

mainFrame.Width = mainGrid.Width;

mainFrame.Height = mainGrid.Height;

}

private Frame GetInnerFrame(Grid grid) => grid.Children[0] as Frame;

#endregion

}

}

using System.Windows.Controls;

using System.Windows.Threading;

namespace AeroViewer

{

public static class Process

{

#region Properties

public static Loader Loader { get; set; }

public static TextBlock StatusTextBox { get; set; }

public static Image SuccessIcon { get; set; }

public static Image FailIcon { get; set; }

public static Dispatcher Dispatcher { get; set; }

#endregion

/// <summary>

/// Sets initial status and makes all icons collapsed, only loader and the status

/// text block are visible

/// </summary>

public static void SetInitialStatus(string initialStatus)

{

Dispatcher.Invoke(() =>

{

StatusTextBox.Text = initialStatus;

Loader.Visibility = System.Windows.Visibility.Visible;

SuccessIcon.Visibility = System.Windows.Visibility.Collapsed;

FailIcon.Visibility = System.Windows.Visibility.Collapsed;

});

}

public static void UpdateStatus(string newStatus) =>

Dispatcher.Invoke(() => StatusTextBox.Text = newStatus);

/// <summary>

/// Sets the final status and displays the result status depends on the isSuccess value

/// </summary>

public static void SetFinalStatus(string finalStatus, bool isSuccess)

{

Dispatcher.Invoke(() =>

{

StatusTextBox.Text = finalStatus;

Loader.Visibility = System.Windows.Visibility.Collapsed;

if (isSuccess)

{

SuccessIcon.Visibility = System.Windows.Visibility.Visible;

FailIcon.Visibility = System.Windows.Visibility.Collapsed;

return;

}

SuccessIcon.Visibility = System.Windows.Visibility.Collapsed;

FailIcon.Visibility = System.Windows.Visibility.Visible;

});

}

}

}

**Список использованной литературы**

1. <https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.datagrid.columnheaderstyle?view=netframework-4.0>, Дата обращения – 18.03.2019