Частное учреждение образования

«Колледж бизнеса и права»

РАЗРАБОТКА ПРОГРАММОГО СРЕДСТВА ДЛЯ РАСЧЕТА СТОИМОСТИ ПОЕЗДКИ НА АВТОМОБИЛЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по предмету

«Конструирование программ и языки программирования»

КП Т.0910018.401

Руководитель проекта (Е.Н. Коропа)

Обучающийся (А.С. Самойлов)

2023

Оглавление

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Датаа*

*Лист*

*3*

*КП Т.0910018.401*

*Разраб.*

*Самойлов А.С.*

*Провер.*

*Коропа Е.Н.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Разработка программного средства для расчета стоимости поездки на автомобиле*

*Лит.*

*Листов*

*КБП*

92

*у*

[Введение 4](#_Toc136948623)

[1 Описание задачи 6](#_Toc136948624)

[1.1 Анализ предметной области 6](#_Toc136948625)

[1.2 Постановка задачи 7](#_Toc136948626)

[2 Проектирование системы 8](#_Toc136948627)

[2.1 Требования к приложению 8](#_Toc136948628)

[2.2 Проектирование модели 8](#_Toc136948629)

[2.3 Организация данных 9](#_Toc136948630)

[2.4 Концептуальный прототип 10](#_Toc136948631)

[3 Описание реализации программного средства 15](#_Toc136948632)

[3.1 Инструменты разработки и применяемые технологии 15](#_Toc136948633)

[3.2 Порядок авторизации пользователей 16](#_Toc136948634)

[3.3 Организация данных 16](#_Toc136948635)

[4 Применение 29](#_Toc136948636)

[4.1 Назначение программного средства 29](#_Toc136948637)

[Заключение 30](#_Toc136948638)

[Список информационных источников 31](#_Toc136948639)

[ПРИЛОЖЕНИЕ А 32](#_Toc136948640)

[ПРИЛОЖЕНИЕ В 90](#_Toc136948641)

**Введение**

В настоящее время все больше людей используют различные виды транспорта для перемещения по городу или между городами. В связи с этим возникает потребность в удобных и быстрых инструментах для определения стоимости поездки. Разработка приложения поможет упростить процесс расчета стоимости поездки для конечных пользователей.

Актуальностью поставленной задачи обусловлена ростом спроса на необходимость расчета стоимости поездки. Также автоматизация этого процесса, который позволит сократить время, затрачиваемое на ручной расчет, и уменьшить вероятность ошибок в расчетах.

При разработке программного средства составлена записка, в которой будут описаны следующие разделы.

В пояснительной записке содержатся четыре раздела.

Первый раздел «Описание задачи» содержит в себе анализ предметной области, в которой автор проекта исследует предметную область, в которой описывает бизнес-процессы происходящие в предметной области, описывает алгоритмы получения промежуточных и итоговых показателей, обосновывает необходимость компьютерной обработки информации или процессов, определяет потенциальную аудиторию, а так же содержит постановку задачи в котором определяет функции которые должны быть автоматизированы на основе проанализированных бизнес-процессов и бизнес-задач, описаны существующих аналогов с указание отличий которые будут реализованы в разрабатываемом программном средстве.

Второй раздел «Проектирование системы» содержит в себе требования к аппаратным и операционным ресурсам, средства защиты будущего приложения, накладываемые ограничения на интерфейс. Также стоит отобразить в разделе диаграммы: вариантов использования с указанием типов связей между вариантами использования, диаграмма классов с выделением необходимых атрибутов и/или методов, диаграмма последовательности или деятельности. Необходимо отразить описание внешнего пользовательского интерфейса и также должны приведены графические изображения макетов и прототипов диалоговых окон.

Третий «Описание реализации программного средства» содержит в себе инструменты разработки и применяемые технологии, порядок авторизации пользователей, организация данных, функции логические и физическая организация, входные и выходные данные, функциональное тестирование, описание справочной системы.

Четвертый раздел «Применение» содержит в себе назначение программного средства, условия применения программного средства.

В заключении будет проанализировано созданное программное средства, определена степень соответствия поставленной задачи и выполненной работы.

Приложения содержат содержит текст программных модулей, результаты работы.

Графическая часть содержит в себе диаграмму вариантов использования, диаграмму классов, диаграмму деятельности.

# **Описание задачи**

## **Анализ предметной области**

Целью является разработка программного средства на языке C# для автоматизации расчета стоимости поездки.

Создание такого приложения имеет актуальность в настоящее время в связи с увеличивающейся потребностью в экономии времени и ресурсов, а также повышенной необходимостью контроля расходов на транспорт. Большинство людей понимают, что иметь возможность автоматически рассчитывать стоимость поездки в режиме реального времени может значительно облегчить их повседневную жизнь и повысить эффективность использования транспортных средств.

Алгоритм для расчета стоимости поездки алгоритм будет включать в себя такие параметры, как расстояние между местом отправления и местом назначения, вид потребления топлива, октановое число потребляемого топлива, расход потребления топлива средняя скорость, характеристики выбранного автомобиля. Расчетное время прибытия рассчитывается по формуле (1).

*(1)*

где t – время, часы;

S– расстояние, км;

V – скорость, км/ч.

Формула стоимости поездки рассчитывается по формуле (2).

*(2)*

где С – стоимость поездки, руб;

R – расстояние, км;

P – потребление топлива на 100 км, литр;

Pp – цена топлива, руб.

Формула затраченного топлива на поездку (3)

*(3)*

где Q – затраченное топливо, литр;

S – расстояние, км;

P – потребление топлива на 100 км, литр.

Такая программа будет максимально облегчать процесс расчета и анализа данных за определенный срок времени.

Это программное средство может быть полезным для широкого круга пользователей, как для частных лиц, так и для предприятий и туристических агентств. Оно может помочь им в быстром и точном расчете стоимости поездок. Безусловно, в наше время все больше людей уделяют внимание здоровому образу жизни и правильному питанию, и это вполне понятно. Также, многие компании и предприятия стараются контролировать свои расходы, включая расходы на поездки.

## **Постановка задачи**

Необходимо разработать программное средство для автоматизации расчета стоимости поездки, в который будут реализованы следующие функции:

* ведение справочников типов топлива, цен на выбранный тип топлива, списка транспортных средств, списком маршрутов;
* автоматический расчет стоимости поездки по введенным критериям пользователя;
* сохранения результатов расчетов в выходной файл, а также сформировать выходной файл в виде отчета или чека;
* формирование и просмотр статистики поездок с расчетом средней стоимости поездок и общего кол-во потребляемого топлива и предоставить пользователю диаграмму расходов пользователя за разные промежутки времени (за выбранный день, месяц, год);
* предоставить возможность разграничение прав доступа к приложению для каждого пользователя.

В результате исследования предметной области были рассмотрены такие аналоги как: «Avtonov», «Transportdepo», «SV-auto», «Fuelio», «TripLog» были обнаружены такие отличия как:

* наличие обширных баз данных с автомобилями и маршрутами, что обеспечивает более точный и полный расчет стоимости поездки;
* кроссплатформенность, что позволяет использовать программное средство на различных операционных системах и устройствах;
* большое количество параметров для подсчета результата, что увеличивает точность расчета и гибкость программы;
* лаконичный и простой интерфейс, что делает использование программы более удобным для пользователя.

Разработанное программное средство будет обладать такими функциями как:

* возможность добавления транспортного средства и маршрута;
* наличии истории поездок, что делает возможным отслеживание статистики и анализ прошлых поездок;
* наличие выходных данных, таких как графики, диаграммы и отчеты, что может предоставить возможность анализа данных.

# **Проектирование системы**

## **Требования к приложению**

Для защиты программного средства будет реализована функция регистрации и авторизации в системе для обеспечения и удобства работы пользователя.

Для работы программного средства необходимо устройство, которое будет иметь следующие характеристики:

* процессор тактовой частотой не менее 2 ГГц, архитектурой x64;
* оперативную память не менее 4 ГБ;
* место на диске от 512 МБ;
* клавиатура проводная;
* мышь проводная;
* видеопамять от 256 МБ;
* операционная система Windows 10 или Windows 11.

Для корректной работы программного средства подключение к интернету не требуется.

Ограничения на интерфейс программы включают в себя:

* размер окна: окно приложения должно быть достаточного размера, чтобы вместить все элементы управления и содержимого.
* разрешение экрана: размер элементов управления и текста должен соответствовать разрешению экрана, для корректного отображения элементов управления на разных устройствах с разным разрешением экрана;
* расположение элементов: необходимо расположить их в интуитивно понятных местах для пользователя и не загромождать окна ненужными элементами;
* шрифт и размер: тест должен быть читаем заметен на форме;

Общие требования к интерфейсу:

* использование стандартных элементов управления, таких как кнопки, текстовые поля;
* понятный и легкий интерфейс для пользователя, чтобы интерфейс был понятен пользователю и не мешал его работе в программе.

## **Проектирование модели**

Проектирование модели начинается с построения UML диаграммы, каждая из которых имеет свою смысловую нагрузки, например диаграмма сущность-связь отображает взаимодействие сущностей между собой. Для реализации функций разработана диаграмма вариантов использования. Диаграмма вариантов использования включает в себя две сущности «Администратор» и «Пользователь». «Пользователь» имеет возможность: рассчитать стоимость поездки, что будет включать в себя ввод данных, выбор данных, формирование отчета; предоставить информацию о типах маршрутах, предоставить информацию о типах топлива, предоставить информацию об автомобилях. «Администратор» имеет возможность: осуществлять работу со справочником топлива, осуществлять работу со справочником автомобилей, осуществлять работу со справочником маршрутов. Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов представлена в графической части на листе 2.

Диаграмма деятельности разработана для функции «Рассчитать стоимость поездки» и представлена в графической части на листе 3.

## **Организация данных**

Исходя из исследования предметной области были выделены следующие сущности: «Автомобиль», «Пользователь», «Поездка», «Маршрут», «Топливо».

Для сущности «Автомобиль» существуют такие атрибуты как:

* Название автомобиля;
* год выпуска;
* максимальная скорость;
* класс автомобиля;
* количество мест;
* вид топлива;
* тип топлива;
* мощность двигателя;
* расход на 100 км смешанный;
* расход на 100 км городской;
* расход на 100 км по трассе;
* объем бензобака.

Для сущности «Пользователь» существуют такие атрибуты как:

* имя;
* фамилия;
* логин;
* пароль.

Для сущности «Поездка» существуют такие атрибуты как:

* имя пользователя;
* автомобиль;
* точка отправления;
* точка прибытия;
* средняя скорость;
* расстояние;
* время в пути;
* сумма денег;
* дата поездки;
* вид топлива
* потраченное топливо.

Для сущности «Маршрут» существуют такие атрибуты как:

* точка отправления;
* точка прибытия;
* расстояние.

Для сущности «Топливо» существуют такие атрибуты как:

* вид топлива;
* октановое число;
* стоимость.

Существует связь между сущностями «Автомобиль» и «Поездка» («используется для»), «Поездка» и «Пользователь» («совершает»), «Маршрут» и «Поездка» («проходит по»), «Автомобиль» и «Топливо» («использует»).

Разработанная диаграмма сущность-связь в нотации Чена изображена рисунке 2.1.

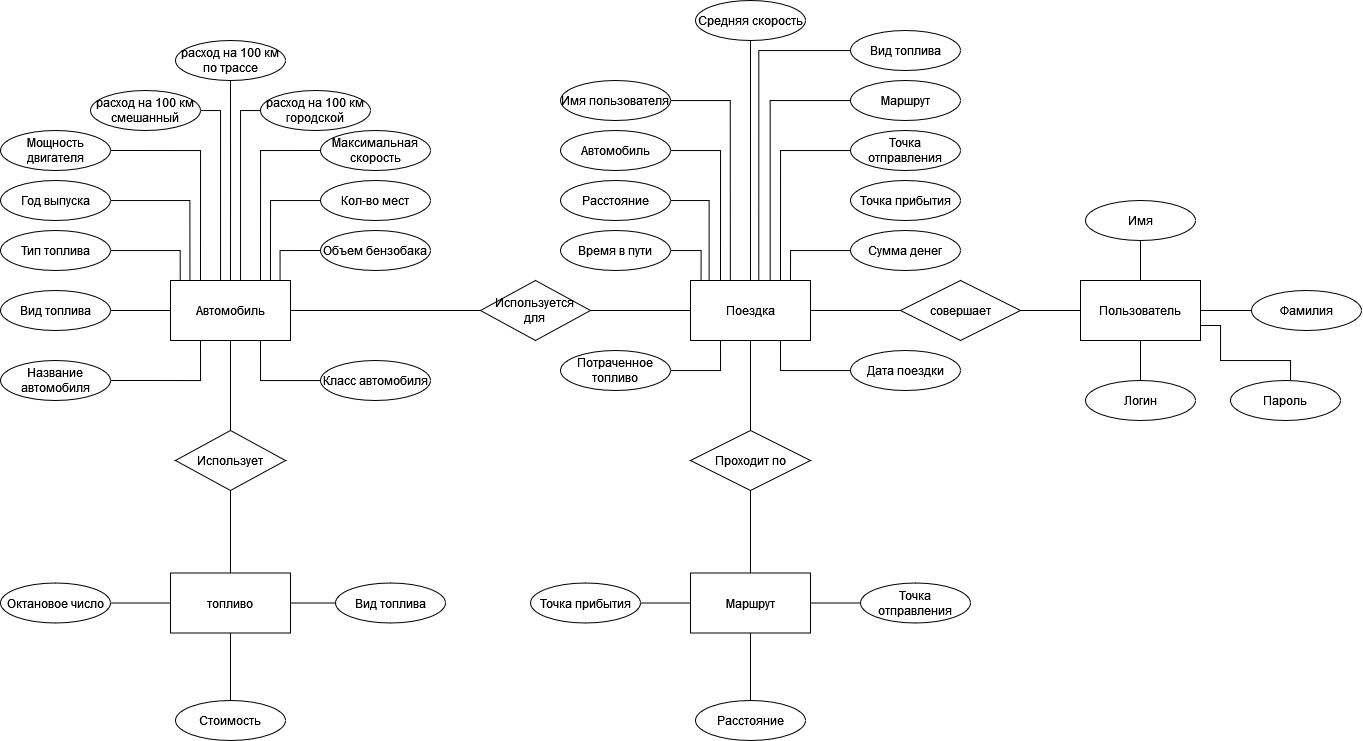


Рисунок 2.1 – Диаграмма сущность-связь

## **Концептуальный прототип**

При запуске программного средства будет отображаться окно аутентификации, в которое будет состоять из текстовых полей и кнопок управления. Прототип окна аутентификации изображен на рисунке 2.2.

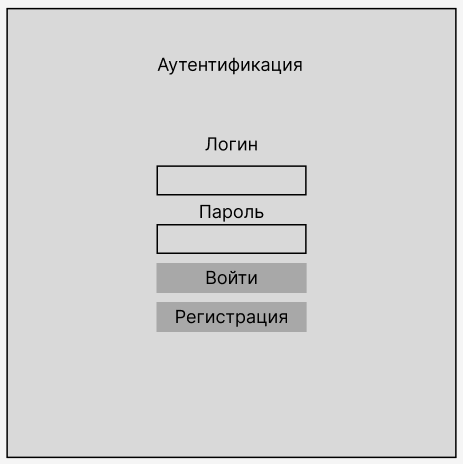


Рисунок 2.2 – Окно аутентификации

В данном окне будет находится два поля для ввода данных и одна кнопка. В поля вводятся логин и пароль пользователя, который используется программой. При нажатии на кнопку «Регистрация» будет открыто окно регистрации, изображенное на рисунке 2.3.

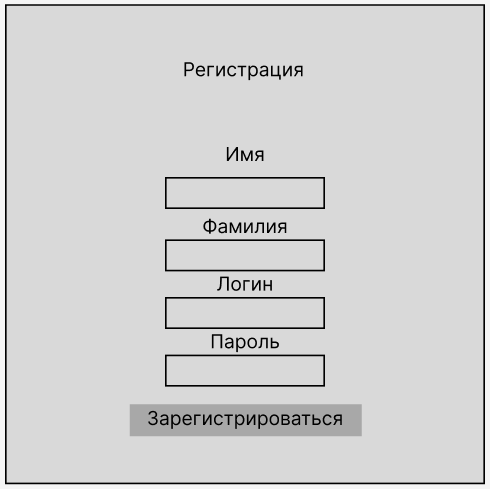


Рисунок 2.3 – Окно регистрации

Окно регистрации будет иметь четыре поля для ввода данных о пользователе и кнопка. При нажатии «Зарегистрироваться» происходит создание нового пользователя и запись его данных в систему для последующего входа в аккаунт.

Вернемся к окну аутентификации и рассмотрим кнопку «Войти», при нажатии на которую произойдет открытие окна главной формы, изображенного на рисунке 2.4.

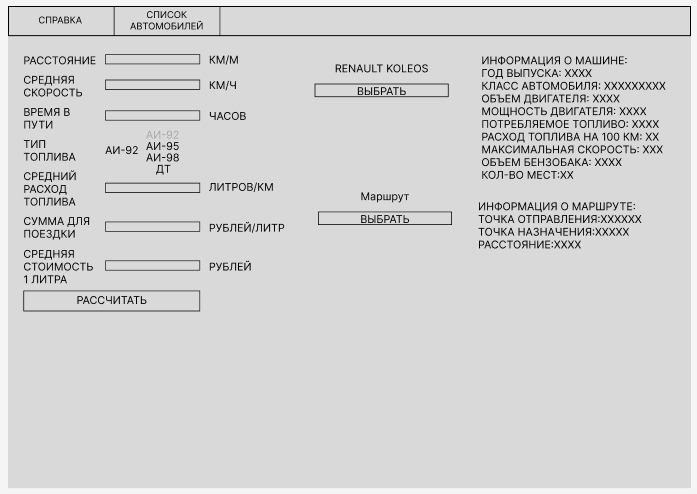


Рисунок 2.4 – Главное окно

При нажатии на кнопку «Справка» программа отобразит новое окно с информацией о программе и алгоритме работы с ним. Окно «Справка» изображено на рисунке 2.5.

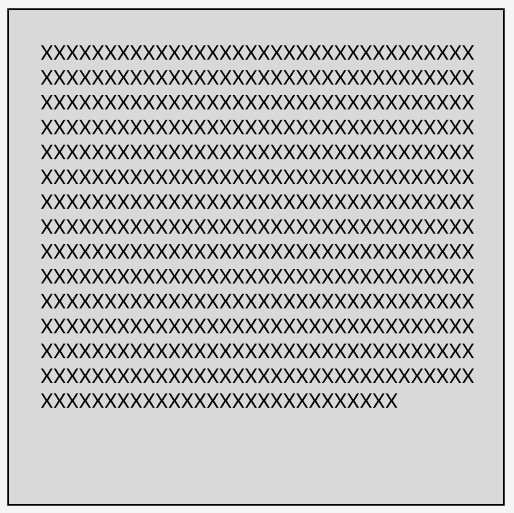


Рисунок 2.5 – Окно справки

Также программе будет предусмотрены окна для работы администратора. Администратор будет иметь одно главное окно для открытия других окон. Главное окно администратора будет иметь в себе несколько кнопок управления. Окно администратора изображено на рисунке 2.6.

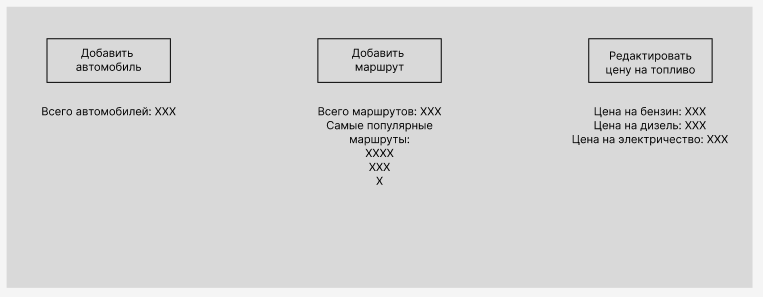


Рисунок 2.6 – Окно администратора

При нажатии на кнопку «Добавить автомобиль» будет открыто окно добавления автомобиля в систему. Окно добавления автомобилей включает в себя множество текстовых полей для ввода и кнопку для добавления автомобиля. Окно добавления изображено на рисунке 2.7.

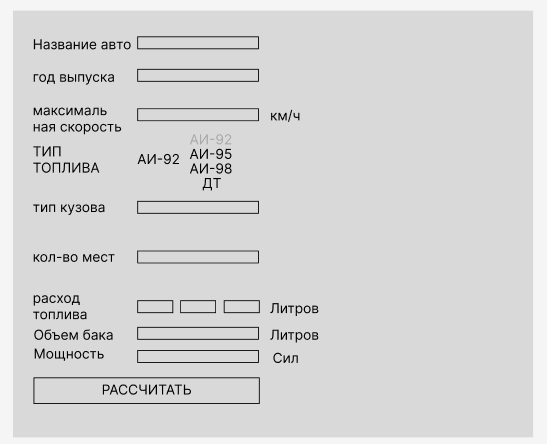


Рисунок 2.7 – Окно добавления автомобиля

При нажатии на кнопку «Добавить маршрут» на главной форме администратора, то будет открыто окно добавления маршрута. Окно добавления маршрута будет включать в себя текстовые поля и кнопку для добавления маршрута. Окно добавления маршрута изображено на рисунке 2.8.

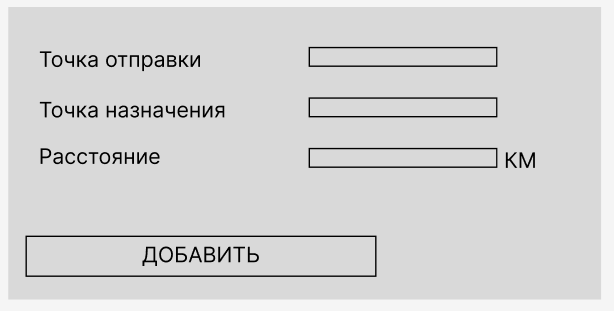


Рисунок 2.8 – Окно добавления маршрута

При нажатии на кнопку «Редактировать цену на топливо» на главной форме администратора, то будет открыто окно редактирования цен на топливо. Окно будет включать в себя текстовые поля и кнопку для изменения цен на топливо. Окно изменения цен изображено на рисунке 2.9.

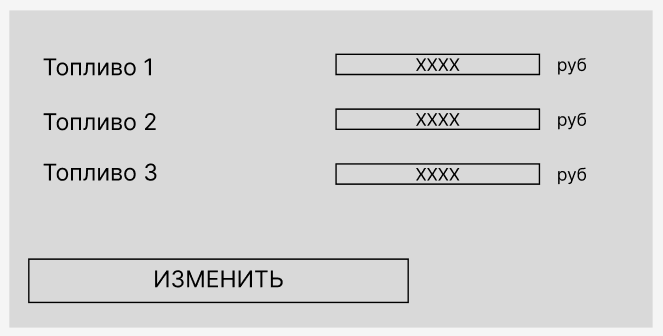


Рисунок 2.9 – Окно изменения цен на топливо

# **Описание реализации программного средства**

## **Инструменты разработки и применяемые технологии**

В ходе реализации программного средства были использованы следующие инструменты:

* операционная система Windows 10;
* среда разработки MS Visual Studio 2022;
* язык программирования C#;
* технология WPF;
* Microsoft Word;
* Dr.Explain.

Windows – группа семейств коммерческих операционных систем корпорации Microsoft. Данная операционная система является универсальной и может использовать для разных целей. Данная операционная система позволяет пользователю компьютера работать с файлами, пользоваться Интернетом, просматривать видеозаписи, прослушивать музыку и многое другое. В данное время операционные системы Windows являются самыми распространенными операционными системами в мире.

Для разработки приложения использовались среда разработки Visual Studio Community. Так как данное программное обеспечение очень удобно и практично для разработки приложения на WPF.

Программное средство было разработано языке программирования C# с использованием WPF для создания окон программы. C# - является компилируемым, статистически типизированным языком программирования общего назначения.

Для выбора языка C# стало его быстродействие в выполнении поставленных задач и желание улучшить навыки владения данным языком программирования.

Для реализации нашего приложения была выбрана технология Windows Presentation Foundation (WPF) в сочетании с инструментами разработки из экосистемы Microsoft. WPF представляет собой фреймворк для создания графического интерфейса пользователя (GUI) в операционной системе Windows. Его основное назначение состоит в разработке интерактивных и эстетически привлекательных приложений с использованием мощного набора инструментов и функциональности. В проектируемом приложении WPF используется для создания интерфейса пользователя. Благодаря богатым возможностям WPF, разработчики могут создавать элементы управления, анимацию, стили и многое другое, что позволяет создать привлекательный и интуитивно понятный пользовательский интерфейс, соответствующий требованиям нашего проекта. В проектируемом приложении WPF используется для создания интерфейса пользователя.

Microsoft Word — [текстовый процессор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных [документов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), с локальным применением простейших форм [таблично](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0)-[матричных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) алгоритмов. Выпускается [корпорацией Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в составе [пакета](https://ru.wikipedia.org/wiki/%D0%9E%D1%84%D0%B8%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) [Microsoft Office](https://ru.wikipedia.org/wiki/Microsoft_Office) [10].

Dr.Explain - программа для быстрого создания файлов справки (help-файлов), справочных систем, on-line руководств пользователя, пособий и документации к программному обеспечению, изделиям, техническим и бизнес-системам [9].

## **Порядок авторизации пользователей**

Программное средство имеет роль администратора и пользователя. Для данного приложения необходима аутентификация и авторизация пользователей.

Роль пользователя имеет такие функции как:

* расчет стоимости поездки, потребления топлива, времени поездки;
* формирование статистики поездок;
* формирование диаграммы расходов пользователя за разные периоды времени;
* формирование отчета за определенную поездку или определенный период.

Роль администратора имеет такие функции как:

* расчет стоимости поездки, потребления топлива, времени поездки;
* формирование статистики поездок;
* формирование диаграммы расходов пользователя за разные периоды времени;
* формирование отчета за определенную поездку или определенный период;
* ведение справочников типов топлива, цен топлива, списка транспортных средств.

## **Организация данных**

Все описанные в разделе «Проектирование модели» сущности необходимо реализовать в виде соответствующих классов. Для сущности «Пользователь» был разработан класс «User», код которого представлен ниже.

public class User

{

public string login { get; set; } // Логин пользователя

public string password { get; set; } // Пароль пользователя

public string name { get; set; } // Имя пользователя

public string surname { get; set; } // Фамилия пользователя

}

Для сущности «Автомобиль» был разработан класс «Сar», код которого

представлен ниже.

public class Car

{

public string Name { get; set; } // Название автомобиля

public int Year { get; set; } // Год выпуска автомобиля

public string TypeCar { get; set; } // Тип автомобиля

public int MaxSpeed { get; set; } // Максимальная скорость автомобиля

public int SeatingCapacity { get; set; } // количество мест в автомобиля

public string Fuel { get; set; } // Тип используемого топлива

public string FuelOctan { get; set; } // Тип топлива

public double FuelConsumptionGeneral { get; set; } // Потребление топлива смешанное

public double FuelConsumptionCity { get; set; } // Потребление топлива в городе

public double FuelConsumptionHighway { get; set; } // Потребление топлива по трассе

public double EnginePower { get; set; } // Мощность двигателя

public double TankSize { get; set; } // Объем бензобака

}

Для сущности «Маршрут» был разработан класс «Route», код которого

представлен ниже.

public class Route

{

public string PointA { get; set; } // Точка отправления

public string PointB { get; set; } // Точка прибытия

public double Distance { get; set; } // Дистанция поездки

}

Для сущности «Топливо» был разработан класс «Fuel», код которого

представлен ниже.

public class Fuel

{

public string name { get; set; } //Название топлива

public string octaneNumber { get; set; } // Тип топлива

public double price { get; set; } // Цена топлива

}

Для сущности «Поездка» был разработан класс «UsersRoutes», код которого представлен ниже.

public class UsersRoutes

{

public string User { get; set; } // Имя пользователя

public string Car { get; set; } // Автомобиль

public string PointA { get; set; } // Точка отправления

public string PointB { get; set; } // Точка прибытия

public double Distance { get; set; } // Расстояние

public double Price { get; set; } // Цена поездки

public string Date { get; set; } // Дата

public string FuelType { get; set; } // Вид топлива

public double UsedFuel { get; set; } // Использовано топлива

public double AverageSpeed { get; set; } // Средняя скорость

}

Для хранения данных классов используется XML – файл, в котором хранятся данные классы, пример структуры файла которого представлен ниже для сущности «Автомобиль»

<?xml version="1.0" encoding="utf-8"?>

<Cars>

<Car>

<name>Toyota Camry</name><!--Название автомобиля-->

<year>2022</year>><!--Год выпуска -->

<typeCar>Седан</typeCar>><!--Класс автомобиля-->

<maxSpeed>130</maxSpeed><!--Максимальная скорость-->

<seatingCapacity>5</seatingCapacity><!--Количество мест -->

<fuel>Бензин</fuel><!--Вид топлива-->

<fuelOctan>АИ-92</fuelOctan><!--Тип топлива-->

<fuelConsumptionGeneral>8</fuelConsumptionGeneral><!--расход на 100 км смешанный-->

<fuelConsumptionCity>10,0</fuelConsumptionCity><!--расход на 100 км городской -->

<fuelConsumptionHighway>5,8</fuelConsumptionHighway><!--расход на 100 км по трассе-->

<enginePower>203</enginePower>

<tankSize>60</tankSize>><!--Объем бензобака-->

</Car>

</Cars>

В данном программном средстве каждый класс проходит сериализацию и записывает данные в XML-файл для последующего использования в программе.

* 1. **Функции: логическая и физическая организация**

Рассмотрим несколько основных функций программного средства.

Функция авторизации пользователя, которая получает данные из текстовых полей и при успешном входе отображает главное окно программы, при неудачном входе отобразит уведомление о ошибке.

Код функции представлен ниже.

private void AuthButtonClick(object sender, RoutedEventArgs e)

{

AllUsers = User.ReadUserInXML(); // инициализация списка AllUser данными из XML файла, путем вызова метода ReadUserInXML

if(AllUsers.Count == 0) // Проверка на наличие пользователей в системе

{

MessageBox.Show("Зарегистрируйтесь в систему.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); // вывод сообщения

return;

}

string login = textBoxLogin.Text; // декларация и инициализация переменной

string password = textBoxPassword.Text;// декларация и инициализация переменной

string userName = "";// декларация и инициализация переменной

if (AdminAccess()) // проверка на вход администратора

{

AdminForm adminForm = new AdminForm(); // создание экземпляра класса

adminForm.ShowDialog(); // отображение окна

return;

}

foreach (var user in AllUsers) {

if (user.login == login && user.password == password) // проверка логина и пароля в списке

{

userName = user.name;

surname = user.surname;

Access = true;

break;

}

}

if (Access)

{

MainWindow mainForm = new MainWindow(userName,surname); // создайние экземпляра класса

mainForm.ShowDialog();// открытие окна mainForm

}

else

{

MessageBox.Show("Ошибка ввода данных. Проверьте логин или пароль.", "Ошибка", MessageBoxButton.OK,MessageBoxImage.Error);// вывод сообщения

}}

Код функции AdminAccess, которая проверяет введенные данные, на ввод данных администратора. Код функции отображен ниже.

public bool AdminAccess()

{

if (textBoxLogin.Text == AdminLog && textBoxPassword.Text == AdminPass)

{

return true;

}

else return false;

}

Функция регистрации пользователя получает данные из текстовых полей окна и регистрирует нового пользователя в систему при успешной регистрации отображает уведомлении о регистрации, при неуспешном вводе отобразит уведомление о ошибке.

Код функции представлен ниже.

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

{

// Проверка валидности текстовых полей textBoxLogin, textBoxPassword, textBoxName, textBoxSurname

if (TextBoxValid(textBoxLogin) & TextBoxValid(textBoxPassword) & TextBoxValid(textBoxName) & TextBoxValid(textBoxSurname))

{

// Создание нового объекта User с данными из текстовых полей

User user = new User(textBoxLogin.Text, textBoxPassword.Text, textBoxName.Text, textBoxSurname.Text);

// Добавление пользователя в XML файл и проверка успешности операции

if (user.AddUserInXML())

{

// Если пользователь успешно добавлен, выводим сообщение об успешном добавлении

MessageBox.Show("Пользователь добавлен", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close(); // Закрытие текущего окна

}

}

else

{

// Если хотя бы одно из текстовых полей не прошло валидацию,

// выводим сообщение об ошибке с указанием возможных причин

MessageBox.Show("Пользователь не добавлен. Возможно ошибка в том что:\n· Пустое поле\n· Длина поля меньше 3 символов\n· Использование запрещенных символов", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Код функции AddUserInXML, которая проверяет введенные данные и добавляет в XML. Код функции представлен ниже.

public bool AddUserInXML()

{

try

{

// Читаем список пользователей из XML файла

List<User> list = ReadUserInXML();

// Проверяем, если уже существует пользователь с таким же логином

foreach (var item in list)

{

if (login == item.login)

{

MessageBox.Show("Данный логин уже занят. Повторите попытку.", "ОшИбка", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

}

// Загружаем корневой элемент из XML файла

XElement root = XElement.Load("User.xml");

// Создаем новый элемент "user" с данными о новом пользователе

XElement userElement = new XElement("user",

new XElement("login", login),

new XElement("password", password),

new XElement("name", name),

new XElement("surname", surname)

);

// Добавляем новый элемент "user" в корневой элемент

root.Add(userElement);

// Сохраняем изменения в XML файле

root.Save("User.xml");

return true;

}

catch (FileNotFoundException ex)

{

// Если файл User.xml не найден, создаем новый XML файл и сохраняем в него данные о пользователе

XDocument xdoc = new XDocument();

XElement user = new XElement("user",

new XElement("login", login),

new XElement("password", password),

new XElement("name", name),

new XElement("surname", surname)

);

XElement users = new XElement("users", user);

xdoc.Add(users);

xdoc.Save("User.xml");

return true;

}

catch (Exception ex)

{

// В случае возникновения других исключений, возвращаем false

return false;

}

}

Функция расчета стоимости поездки, которая принимает введенные в текстовые поля данные, для расчета стоимости поездки. После успешного ввода данных, результат будет отображен в текстовых полях. Если данные введены некорректно, будет выведено сообщение об ошибке. Код функции представлен ниже.

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

{

if (ValidValue())

{

// Получаем значения из комбо-боксов

string car = SetValue(comboBoxCar);

string PointA = SetValue(comboBoxPointOne);

string PointB = SetValue(comboBoxPointTwo);

// Получаем выбранный тип топлива

string fuelType = comboBoxFuelType.SelectedItem.ToString();

// Получаем числовые значения из текстовых полей

double dictance = Convert.ToDouble(this.textBoxDistance.Text);

double fuelPrice = Convert.ToDouble(this.textBoxFuelPrice.Text);

double consimption = Convert.ToDouble(this.textBoxConsumption.Text);

double averageSpeed = Convert.ToDouble(this.textBoxAverSpeed.Text);

// Проверяем условие и увеличиваем расход топлива, если средняя скорость превышает 140 км/ч

if (averageSpeed > 140 && car == "Неизвестно")

{

consimption += 1;

}

// Выполняем необходимые расчеты

double usedFuel = Math.Round(dictance / consimption);

double result = dictance / averageSpeed;

double fullPrice = Math.Round((dictance / 100) \* fuelPrice \* consimption, 2);

// Получаем выбранную дату

DateTime dateOne = DataPickerFirstData.SelectedDate.Value;

string date = dateOne.ToString("dd.MM.yyyy");

// Выводим результаты в соответствующие текстовые поля

this.textBoxUsedFuel.Text = usedFuel.ToString();

this.textBoxTime.Text = Math.Round(result, 2).ToString();

this.textBoxPrice.Text = fullPrice.ToString();

// Создаем объект UsersRoutes и добавляем маршрут в XML

UsersRoutes route = new UsersRoutes(UserName, car, PointA, PointB, dictance, fullPrice, fuelType, date, usedFuel, averageSpeed);

if (route.AddRoutesInXML())

{

// Предлагаем сохранить маршрут в docx файле

MessageBoxResult message = MessageBox.Show($"Поездка оформлена. Желаете сохранить данную поезку в docx файле?", "Подтверждение", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (message == MessageBoxResult.OK)

{

CreateDocxFile(dateOne, car, PointA, PointB, dictance, fullPrice, fuelType, usedFuel, averageSpeed);

}

}

else

{

MessageBox.Show("Поездка не оформлена", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Заполните все поля корректными значениями", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Функция добавления нового маршрута в справочник, которая принимает введенные в текстовые поля данные, для добавления маршрута. После успешного ввода данных и добавленного маршрута в XML, отобразится уведомлении. Если данные введены некорректно, будет выведено сообщение об ошибке.

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

{

// Получаем значения из текстовых полей и удаляем возможные пробелы в конце строки

string pointA = textBoxPointA.Text.TrimEnd();

string pointB = textBoxPointB.Text.TrimEnd();

// Инициализируем переменную для расстояния и проверяем валидность введенного значения

double distance = 0;

bool IsValidValue = IsValidDoubleInput(textBoxDistance, 0, 40000, out distance);

// Проверяем условие валидности введенных данных

if ((TextBoxValid(textBoxPointA) & TextBoxValid(textBoxPointB) & IsValidValue) & (pointA != pointB))

{

// Создаем объект маршрута и добавляем его в XML

Route route = new Route(pointA, pointB, distance);

if (route.AddRoutesInXML())

{

MessageBox.Show($"Маршрут {pointA} - {pointB} добавлен в систему.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show($"Маршрут {pointA} - {pointB} не добавлен, так как уже есть в системе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Ошибка в маршруте, проверьте маршрут и повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Полный код функций представлен в приложении А.

* 1. **Входные и выходные данные**

Для роли администратора определены следующие входные данные:

* логин;
* пароль;
* добавление данных о маршрутах;
* добавление данных о автомобилях;
* добавление данных о типах топлива;
* данные для расчета о поездках.

Для роли пользователя определены следующие входные данные:

* логин;
* пароль;
* имя;
* фамилия;
* данные для расчета о поездках.

Для пользователя и администратора выходные данные будут:

* отчет за текущую поездку;
* отчет расходов пользователя за определенный срок.
  1. **Функциональное тестирование**

Функциональное тестирование – это тестирование программного обеспечения (ПО) в целях проверки реализации функциональных требований в определенных условиях решать задачи, поставленные пользователем.

В данное программное средстве реализованная система ситуаций и ошибок, которые обрабатывается в программное средство (ПС), информация о ошибке выводится в выпадающем окне.

Для функции регистрации, аутентификации, добавления маршрута, расчета стоимости поездки составлены тест-кейсы, отображенные в таблице 3.1.

Таблица 3.1 – Тест-кейсы для функций

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 1 | Регистрация пользователя | 1. Осуществить переход по кнопке «Зарегистрироваться»  2.Заполнить требуемые поля валидными данными  - «Логин» – Samsik  - «Пароль» – 123123  - «Имя» – Саша  - «Фамилия» – Самойлов  3.Нажать на кнопку «Зарегистрироваться» | Ожидаемый: отображается уведомление о успешной регистрации |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены Результаты тестирования приведены на Рисунке Б.1 |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 2 | Авторизация в приложении | 1. Осуществить переход в окно аутентификации  2. Заполнить требуемые поля необходимыми данными  - «Логин» – Samsik  - «Пароль» – Не заполнен  3. Нажать кнопку "войти" | Ожидаемый: отображается уведомление о некорректных данных |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены. Результаты тестирования приведены на рисунке Б.2 |
| 3 | Добавление маршрута | 1. Выбрать кнопку «Добавить маршрут» в окне «Окно администратора»  2. Заполнить требуемые поля необходимыми данными  - «Точка отправления» – Минск  - «Точка прибытия» – Сочи  - «Расстояние» - 2660  3. Нажать кнопку "Добавить маршрут" | Ожидаемый: отображается уведомление о добавлении нового маршрута |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены. Результаты тестирования приведены на рисунке Б.3 |
| 4 | Расчет стоимости поездки | 1. Осуществить переход в окно «Главная».  2. Заполнить все поля необходимыми корректными данными  3. Нажать кнопку "Рассчитать" | Ожидаемый: отображается уведомление о расчете стоимости поездки и вывод рассчитанных данных на окно |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены. Результаты тестирования приведены на рисунке Б.4 |
| 5 | Расчет стоимости поездки | 1. Осуществить переход в окно «Главная».  2. Заполнить некоторые поля некорректными и некоторые корректными данными  3. Нажать кнопку "Рассчитать" | Ожидаемый:  отображается уведомление о некорректно введенных данных и отображение некорректно введённых данных |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены. Результаты тестирования приведены на рисунке Б.5 |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 6 | Добавление автомобиля | 1. Выбрать кнопку «Добавить автомобиль» в окне «Окно администратора»  2. Заполнить требуемые поля необходимыми корректными данными  3. Нажать кнопку "Добавить автомобиль" | Ожидаемый: отображается уведомление о добавлении нового автомобиля |
| Фактический: фактический результат соответствует ожидаемому, все условия соблюдены. Результаты тестирования приведены на рисунке Б.6 |

* 1. **Описание справочной системы**

Справочная система реализована в формате .chm разработанная при помощи инструмента Dr.Explain.

Dr.Explain – это мощный инструмент для создания справочных систем и документации. Он обладает широкими возможностями по автоматизации процесса создания и обновления справочных материалов, обеспечивая высокую производительность и эффективность в работе. Система на базе Dr.Explain позволяет авторам создавать справочники, инструкции, помощь пользователям и другую документацию в удобном и привлекательном виде.

В данной справочной системе имеются 8 разделов.

В разделе «Введение» описано назначение справочной системы.

В разделе «Цель программного средства» описывается назначение данного программного средства

В разделе «Аудитория и предполагаемые пользователи» описывается аудитория и предполагаемые пользователи данного программного средства. Так же описывается назначение программного средства для каждого типа пользователя.

В разделе «Системные требования программного средства» описаны системные требования для установки и работы с данным программным средством.

В разделе «Основные функции» описывается основные функции и возможности программного средства.

В разделе «Обзор пользовательского интерфейса» описывается основные окна и рекомендации к работе с приложением.

В разделе «Выходные данные» описывается все виды выходных файлов. Отображаются примеры данных документов для просмотра их.

В разделе «Заключение» описывается разработчик и цель программного средства.

Для вызова справочной системы используется MenuItem, который находится на главной странице программы, после нажатия на пункт меню «Справка» откроется справочная система в отдельном окне. Справка изображена на рисунке 3.1.

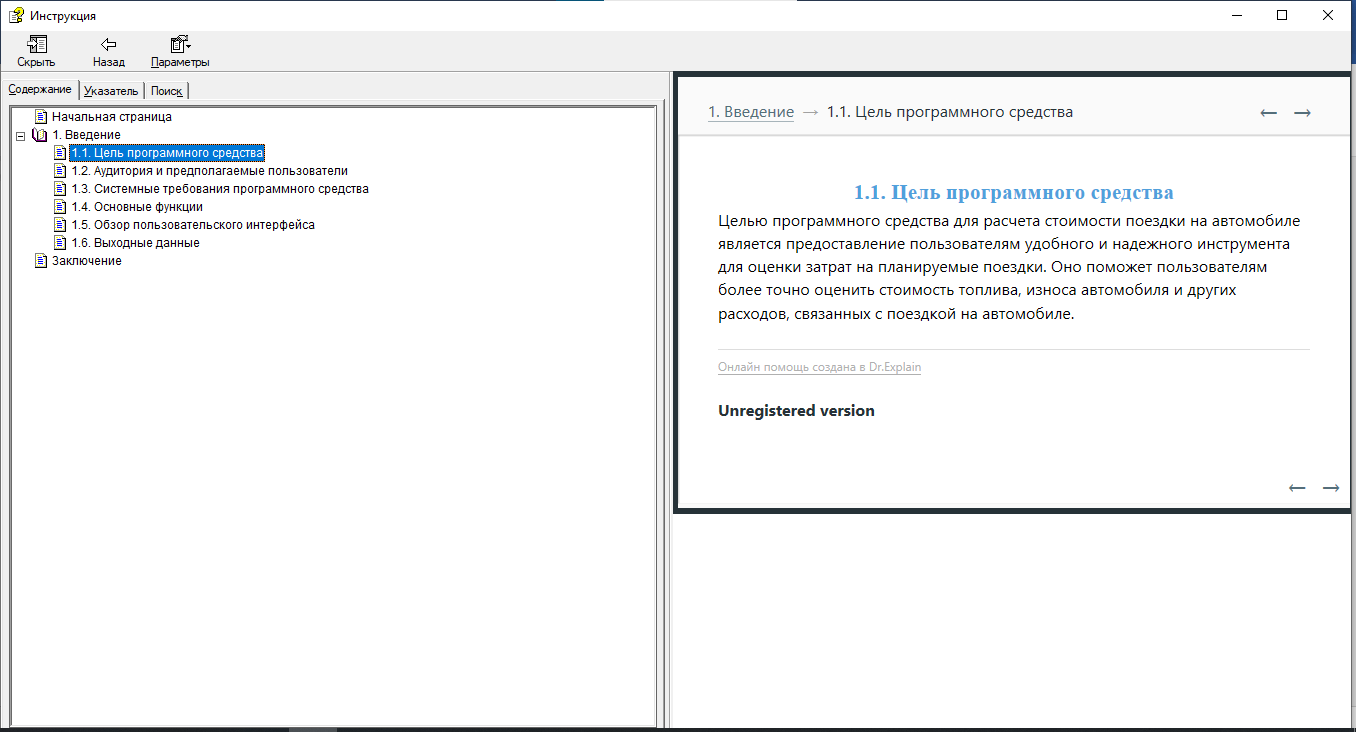


Рисунок 3.1 – Содержание справки

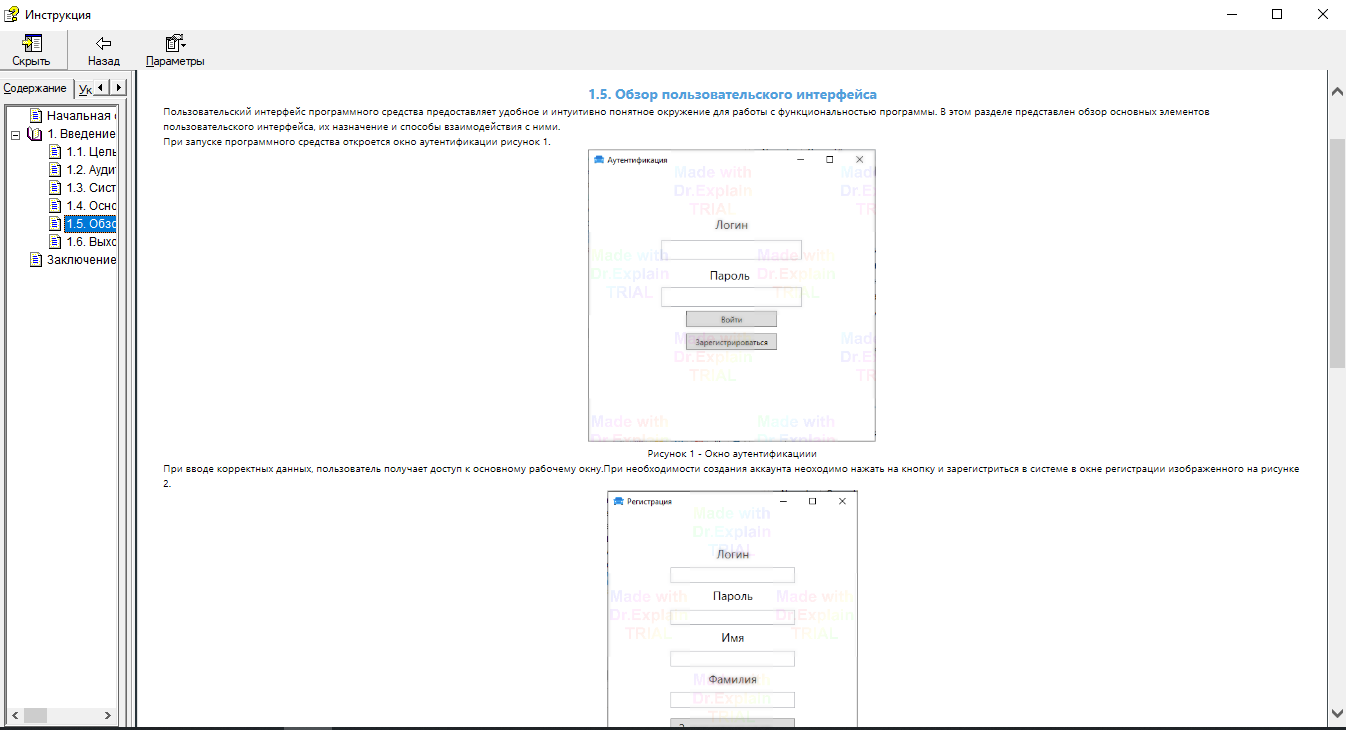


Рисунок 3.2 – Пример раздела справочной системы

# **Применение**

## **Назначение программного средства**

Программное средство разработано для автоматизации расчета стоимости поездки. Данное программное средства называется «JourneyExpense». Для запуска программного средства установка каких-то дополнительных библиотек не требуется, все необходимые библиотеки устанавливаются автоматически.

* 1. **Условия применения**

Средства, ограничивающие работу приложения, не были выявлена. Для корректной работы программного средства требуется соблюдать минимальные системные требования программы, а именно:

* процессор тактовой частотой не менее 2 ГГц, архитектурой x64;
* оперативную память не менее 4 ГБ;
* место на диске от 512 МБ;
* клавиатура проводная;
* мышь проводная;
* видеопамять от 256 МБ;
* операционная система Windows 10 или Windows 11.

Для работы с программным средством необходимо установить Microsoft

.Net Framework 4.8. Данный компонент будет предложено установить в ходе установки программы.

# **Заключение**

В результате работы над курсовым проектом было разработано и создано программное средство, реализующее программу для автоматизации стоимости поездки.

Для разработки программного средства было изучено и сформировано значение о WPF, а также значение о основах разработки приложений на C#. В ходе тестирования приходилось корректировать уже созданный алгоритм для более точного расчета.

В данном программном средства реализованы такие задачи как: регистрация и авторизация в системе, расчет затрат на поездку, ведение справочников системы, формирование статистика поездок, формирование графика расходов пользователя за разные промежутки времени.

Преимуществом данного программного средства является легкость в освоении, понятный интерфейс и низкие системные требования.

Недостатками данного ПС является такие свойства как: ограниченный функционал, отсутствие кроссплатформенности, отсутствие обновление, отсутствие обновления цен на топливо в реальном времени.

Разработанное программное средство является готовым продуктом, готовым к использованию.

# **Список информационных источников**

1. Агуров, Павел. C#. Сборник рецептов / Павел Агуров. - М.: «БХВ-Петербург», **2012**. - 432 c
2. Альфред, В. Ахо. Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015.- **266** c.
3. Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, **2013**. - 472 c.
4. Джозеф, Албхари. Справочник. Полное описание языка/ Джозеф Албхари. – СПБ: ООО «Диалектика», 2021. – 1056 c.
5. Подбельский В.В. Программирование. Базовый курс С#: учебник для вузов/ В.В. Подбельский. – Москва: Издательство Юрайт, 2020. – 369 c.
6. Прайс Марк. С# и .NET Core Разработка и оптимизация/ Прайс Марк. – СПБ, 2020. – 832 c.
7. Подбельский, В. В. Язык С#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 c
8. Прайс, Джейсон Visual C# 2.0. Полное руководство / Джейсон Прайс, Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, **2010**. - 736 c.
9. Drexplain.ru [Электронный ресурс]. – drexplain, 2023. – Режим доступа: https://www.drexplain.ru/. – Дата доступа 05.06.2023.
10. Wikipedia [Электронный ресурс]. – Microsoft Word, 2023. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Word. – Дата доступа 05.06.2023.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программных модулей**

<Window x:Class="JourneyExpense.AuthForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Аутентификация"

Height="456" MinHeight="456" MaxHeight="456"

Width="456" MinWidth="456" MaxWidth="456"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="104\*"/>

<RowDefinition Height="55\*"/>

<RowDefinition Height="61\*"/>

<RowDefinition Height="61\*"/>

<RowDefinition Height="55\*"/>

<RowDefinition Height="111\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="23\*"/>

<ColumnDefinition Width="22\*"/>

<ColumnDefinition Width="22\*"/>

<ColumnDefinition Width="23\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="AuthButton" Content="Войти" Margin="38,12,38,0" VerticalAlignment="Top" Grid.ColumnSpan="2" Height="25" Grid.Row="3" Grid.Column="1" Click="AuthButtonClick"/>

<TextBox x:Name="textBoxLogin" Grid.ColumnSpan="2" Grid.Row="1" VerticalAlignment="Center" Height="30" Grid.Column="1" FontSize="20"/>

<Button x:Name="RegButton" Grid.Column="1" Content="Зарегистрироваться" Margin="38,47,38,0" Grid.Row="3" VerticalAlignment="Top" Grid.ColumnSpan="2" Grid.RowSpan="2" Click="RegButtonClick" Height="25"/>

<PasswordBox x:Name="textBoxPassword" Grid.ColumnSpan="2" Grid.Row="2" VerticalAlignment="Top" Height="30" Grid.Column="1" Margin="0,33,0,0" Grid.RowSpan="2" FontSize="20"/>

<Label Grid.Column="1" Content="Логин" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" FontSize="18" Margin="55,67,55,0" Height="35" VerticalAlignment="Top"/>

<Label Grid.Column="1" Content="Пароль" Grid.Row="1" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" Height="32" VerticalAlignment="Top" FontSize="18" Margin="52,48,58,0" Grid.RowSpan="2"/>

</Grid>

</Window>

using System.Collections.Generic;

using System.Windows;

using JourneyExpense.Classes;

namespace JourneyExpense

{

public partial class AuthForm : Window

{

private List<User> AllUsers = new List<User>();

private string surname = "";

private bool Access = false;

private string AdminLog = "Admin";

private string AdminPass = "Admin";

public AuthForm()

{

InitializeComponent();

}

private void AuthButtonClick(object sender, RoutedEventArgs e)

{

AllUsers = User.ReadUserInXML(); // инициализация списка AllUser данными из XML файла, путем вызова метода ReadUserInXML

if(AllUsers.Count == 0) // Проверка на наличие пользователей в системе

{

MessageBox.Show("Зарегистрируйтесь в систему.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); // вывод сообщения

return;

}

string login = textBoxLogin.Text; // декларация и инициализация переменной

string password = textBoxPassword.Password;// декларация и инициализация переменной

string userName = "";// декларация и инициализация переменной

if (AdminAccess()) // проверка на вход администратора

{

AdminForm adminForm = new AdminForm(); // создание экземпляра класса

adminForm.ShowDialog(); // отображение окна

return;

}

foreach (var user in AllUsers)

{

if (user.login == login && user.password == password) // проверка логина и пароля в списке

{

userName = user.name;

surname = user.surname;

Access = true;

break;

}

}

if (Access)

{

MainWindow mainForm = new MainWindow(userName,surname); // создайние экземпляра класса

mainForm.ShowDialog();// открытие окна mainForm

}

else

{

MessageBox.Show("Ошибка ввода данных. Проверьте логин или пароль.", "Ошибка", MessageBoxButton.OK,MessageBoxImage.Error);// вывод сообщения

}

}

public bool AdminAccess() // проверка на вход администратора

{

if (textBoxLogin.Text == AdminLog && textBoxPassword.Password == AdminPass)

{

return true;

}

else return false;

}

private void RegButtonClick(object sender, RoutedEventArgs e)//обработчик нажатия на кнопку для открытия формы регисрации

{

RegForm RegForm = new RegForm();

RegForm.ShowDialog();

}

}

}

<Window x:Class="JourneyExpense.RegForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Регистрация"

Height="450" MinHeight="450" MaxHeight="450"

Width="400" MinWidth="400" MaxWidth="400"

Icon="/Icons/Icons.ico" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="49\*"/>

<RowDefinition Height="68\*"/>

<RowDefinition Height="68\*"/>

<RowDefinition Height="68\*"/>

<RowDefinition Height="68\*"/>

<RowDefinition Height="58\*"/>

<RowDefinition Height="56\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="20\*"/>

<ColumnDefinition Width="9\*"/>

<ColumnDefinition Width="11\*"/>

<ColumnDefinition Width="11\*"/>

<ColumnDefinition Width="9\*"/>

<ColumnDefinition Width="20\*"/>

</Grid.ColumnDefinitions>

<Label Grid.Column="2" Content="Логин" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" FontSize="18" Grid.Row="1" Margin="0,1,0,0" Height="35" VerticalAlignment="Top"/>

<Label Grid.Column="2" Content="Пароль" Grid.Row="2" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" Height="32" VerticalAlignment="Top" FontSize="18" Margin="0,1,0,0"/>

<Label Grid.Column="2" Content="Имя" Grid.Row="3" VerticalAlignment="Top" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" Height="32" FontSize="18" Margin="0,1,0,0"/>

<Label Grid.Column="2" Content="Фамилия" Grid.Row="4" Grid.ColumnSpan="2" HorizontalContentAlignment="Center" FontSize="18" Margin="0,1,0,0" Height="31" VerticalAlignment="Top"/>

<Button Grid.Column="1" Content="Зарегистрироваться" Grid.Row="5" Grid.ColumnSpan="4" Height="28" VerticalAlignment="Top" Margin="0,15,0,0" FontSize="18" Click="Button\_Click"/>

<TextBox Grid.Column="1" Grid.ColumnSpan="4" Grid.Row="1" TextWrapping="Wrap" Margin="0,40,0,1" FontSize="20"/>

<TextBox x:Name="textBoxSurname" Grid.Column="1" Grid.ColumnSpan="4" Grid.Row="4" Margin="0,38,0,1" FontSize="15"/>

<PasswordBox x:Name="textBoxPassword" Grid.Column="1" Grid.ColumnSpan="4" Grid.Row="2" Margin="0,40,0,1" FontSize="15"/>

<TextBox x:Name="textBoxLogin" Grid.Column="1" Grid.ColumnSpan="4" Grid.Row="1" Margin="0,40,0,1" FontSize="15"/>

<TextBox x:Name="textBoxName" Grid.Column="1" Grid.ColumnSpan="4" Grid.Row="3" Margin="0,39,0,1" FontSize="15"/>

</Grid>

</Window>

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using JourneyExpense.Classes;

namespace JourneyExpense

{

public partial class RegForm : Window

{

public RegForm()

{

InitializeComponent();

}

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

{

// Проверка валидности текстовых полей textBoxLogin, textBoxPassword, textBoxName, textBoxSurname

if (TextBoxValid(textBoxLogin) & PasswordValid(textBoxPassword) & TextBoxValid(textBoxName) & TextBoxValid(textBoxSurname))

{

// Создание нового объекта User с данными из текстовых полей

User user = new User(textBoxLogin.Text, textBoxPassword.Password, textBoxName.Text, textBoxSurname.Text);

// Добавление пользователя в XML файл и проверка успешности операции

if (user.AddUserInXML())

{

// Если пользователь успешно добавлен, выводим сообщение об успешном добавлении

MessageBox.Show("Пользователь добавлен", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close(); // Закрытие текущего окна

}

}

else

{

// Если хотя бы одно из текстовых полей не прошло валидацию,

// выводим сообщение об ошибке с указанием возможных причин

MessageBox.Show("Пользователь не добавлен. Возможно ошибка в том что:\n· Пустое поле\n· Длина поля меньше 3 символов\n· Использование запрещенных символов", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private bool TextBoxValid(TextBox text)

{

if (text.Text != "" & Regex.IsMatch(text.Text, @"^[a-zA-Zа-яА-Я\s\d\-\_]+$") & text.Text.Length > 3 & text.Text != "Admin")

{

text.BorderBrush = Brushes.Gray;

return true;

}

else

{

text.BorderBrush = Brushes.Red;

return false;

}

}

private bool PasswordValid(PasswordBox text)

{

if (text.Password != "" & Regex.IsMatch(text.Password, @"^[a-zA-Zа-яА-Я\s\d\-\_]+$") & text.Password.Length > 3 & text.Password != "Admin")

{

text.BorderBrush = Brushes.Gray;

return true;

}

else

{

text.BorderBrush = Brushes.Red;

return false;

}

}

}

}

<Window x:Class="JourneyExpense.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:JourneyExpense"

mc:Ignorable="d"

Title="Главная страница"

Height="450" MinHeight="450" MaxHeight="450"

Width="855" MinWidth="855" MaxWidth="855"

WindowStartupLocation="CenterScreen"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="47\*"/>

<RowDefinition Height="47\*"/>

<RowDefinition Height="16\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="210"/>

<ColumnDefinition Width="145"/>

<ColumnDefinition Width="95"/>

<ColumnDefinition Width="165"/>

<ColumnDefinition Width="160"/>

<ColumnDefinition Width="80"/>

<ColumnDefinition Width="0\*"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="textBoxDistance" FontSize="15" Grid.Column="2" Grid.Row="1" Margin="10,11,10,0" Height="27" VerticalAlignment="Top"/>

<Label Content="Расстояние" Grid.Column="1" Grid.Row="1" Margin="10,0,10,0" Height="27" VerticalAlignment="Center" FontSize="15"/>

<TextBox x:Name="textBoxAverSpeed" FontSize="15" Height="27" VerticalAlignment="Top" Grid.Column="2" Grid.Row="2" Margin="10,11,10,0"/>

<Label Content="Средняя скорость" FontSize="15" Grid.Column="1" Grid.Row="2" Margin="10,11,10,0" Height="34" VerticalAlignment="Top"/>

<Label Content="Примерное время в пути" Grid.Column="1" Grid.Row="3" Margin="10,8,10,0" Height="33" VerticalAlignment="Top" FontSize="15"/>

<TextBox x:Name="textBoxTime" FontSize="15" Grid.Column="2" Grid.Row="3" Margin="11,11,10,0" Height="27" VerticalAlignment="Top" IsReadOnly="True"/>

<Label Content="Тип топлива" VerticalAlignment="Center" FontSize="15" Grid.Column="1" Grid.Row="4" Height="27" Margin="10,0,10,0"/>

<ComboBox x:Name="comboBoxFuelType" Grid.Column="2" Grid.Row="4" VerticalAlignment="Center" Margin="10,0,10,0" Height="27" SelectionChanged="comboBoxFuelType\_SelectionChanged\_1" MouseDoubleClick="comboBoxFuelType\_MouseDoubleClick"/>

<Label Content="Расход топлива на 100 км" FontSize="15" Grid.Column="1" Grid.Row="6" Margin="10,10,10,0" Height="27" VerticalAlignment="Top"/>

<TextBox x:Name="textBoxConsumption" FontSize="15" Grid.Column="2" Grid.Row="6" Margin="10,10,10,0" Height="27" VerticalAlignment="Top"/>

<Label Grid.Column="3" Content="Км" Margin="28,0,22,0" Grid.Row="1" VerticalAlignment="Center" FontSize="15" Height="27" HorizontalContentAlignment="Center"/>

<Label Grid.Column="3" Content="Км/Ч" Grid.Row="2" VerticalAlignment="Center" FontSize="15" Height="27" Margin="19,0,18,0" HorizontalContentAlignment="Center"/>

<Label Grid.Column="3" Content="Часов" Grid.Row="3" VerticalAlignment="Center" FontSize="15" Height="27" Margin="17,0,10,0" HorizontalContentAlignment="Center"/>

<Label x:Name="LabelConsumption" Grid.Column="3" Content="Литров" Grid.Row="6" VerticalAlignment="Top" FontSize="15" Height="29" Margin="5,9,5,0" HorizontalContentAlignment="Center"/>

<Label Content="Тип потребления топлива" FontSize="15" Grid.Column="1" Grid.Row="5" Margin="10,0,10,0" Height="29" VerticalAlignment="Center"/>

<ComboBox x:Name="comboBoxConsumption" Grid.Column="2" Grid.Row="5" VerticalAlignment="Top" Margin="10,11,10,0" Height="27" SelectionChanged="comboBoxConsumption\_SelectionChanged"/>

<Label Content="Итоговая сумма" FontSize="15" Grid.Column="1" Grid.Row="7" Margin="10,10,10,0" Height="30" VerticalAlignment="Top"/>

<TextBox x:Name="textBoxPrice" FontSize="15" Grid.Column="2" Grid.Row="7" Margin="10,10,10,0" Height="27" VerticalAlignment="Top" IsReadOnly="True"/>

<Label x:Name="LabelConsumption\_Copy" Grid.Column="3" Content="Рублей" Grid.Row="7" VerticalAlignment="Center" FontSize="15" Height="29" Margin="5,0,5,0" HorizontalContentAlignment="Center"/>

<Button x:Name="ButtonCalculate" Grid.Column="1" Content="Рассчитать" Grid.Row="8" Margin="10,10,10,10" Click="ButtonCalculate\_Click" />

<ComboBox x:Name="comboBoxFuelOctan" Grid.Column="3" Grid.Row="4" VerticalAlignment="Center" Margin="10,0,10,0" Height="27" SelectionChanged="comboBoxFuelOctan\_SelectionChanged"/>

<Label Content="Автомобиль" Grid.Column="4" Grid.Row="1" Margin="15,0,15,0" Height="27" VerticalAlignment="Center" FontSize="15"/>

<Label Content="Точка отправления" Grid.Column="4" Grid.Row="2" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<ComboBox x:Name="comboBoxCar" Grid.Column="5" Grid.Row="1" VerticalAlignment="Center" Margin="18,0,18,0" Height="27" SelectionChanged="comboBoxCar\_SelectionChanged" MouseDoubleClick="comboBoxCar\_MouseDoubleClick"/>

<ComboBox x:Name="comboBoxPointOne" Grid.Column="5" Grid.Row="2" VerticalAlignment="Top" Margin="18,10,18,0" Height="27" SelectionChanged="comboBoxPointOne\_SelectionChanged" MouseDoubleClick="comboBoxPointOne\_MouseDoubleClick"/>

<Label Content="Точка прибытия" Grid.Column="4" Grid.Row="3" Margin="10,8,10,0" Height="29" VerticalAlignment="Top" FontSize="15"/>

<ComboBox x:Name="comboBoxPointTwo" Grid.Column="5" Grid.Row="3" VerticalAlignment="Top" Margin="18,10,18,0" Height="27" SelectionChanged="comboBoxPointTwo\_SelectionChanged" MouseDoubleClick="comboBoxPointTwo\_MouseDoubleClick"/>

<Label Content="Цена топлива" Grid.Column="4" Grid.Row="4" Margin="10,10,10,0" Height="29" VerticalAlignment="Top" FontSize="15"/>

<TextBox x:Name="textBoxFuelPrice" FontSize="15" Grid.Column="5" Grid.Row="4" Margin="18,10,18,0" Height="27" VerticalAlignment="Top"/>

<Menu Grid.ColumnSpan="7">

<MenuItem Header="Очистить" Height="29" Width="70" Margin="10,0,0,0" Click="MenuItem\_Click"></MenuItem>

<MenuItem Header="Статистика" Height="29" Width="80" Click="MenuItem\_Click\_1"></MenuItem>

<MenuItem Header="Справка" Height="29" Width="80" Click="MenuItem\_Click\_2"></MenuItem>

</Menu>

<Label Content="Использовано топлива" Grid.Column="4" Grid.Row="5" Height="29" VerticalAlignment="Top" FontSize="15" Grid.ColumnSpan="2" Margin="10,10,140,0"/>

<TextBox x:Name="textBoxUsedFuel" FontSize="15" Grid.Column="5" Grid.Row="5" Margin="18,10,18,0" Height="27" VerticalAlignment="Top" IsReadOnly="True"/>

<Label Content="Дата отправки" Grid.Column="4" Grid.Row="6" Height="29" VerticalAlignment="Top" FontSize="15" Grid.ColumnSpan="2" Margin="10,9,140,0"/>

<DatePicker x:Name="DataPickerFirstData" HorizontalAlignment="Center" Grid.Row="6" VerticalAlignment="Center" Width="124" Grid.Column="5"/>

<Label x:Name="LabelLitr" Grid.Column="6" Content="Литров" Grid.Row="5" VerticalAlignment="Top" FontSize="15" HorizontalAlignment="Left" Width="64" Margin="0,9,0,0"/>

<Label Grid.Column="6" Content="Рублей" Grid.Row="4" VerticalAlignment="Top" FontSize="15" HorizontalAlignment="Left" Width="64" Margin="0,7,0,0"/>

</Grid>

</Window>

using JourneyExpense.Classes;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using Xceed.Document.NET;

using Xceed.Words.NET;

namespace JourneyExpense

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow(string userName, string surname)

{

UserName = userName;

Surname = surname;

InitializeComponent();

InitList();

InitComboBox();

}

private string UserName;

private string Surname;

private bool isMessageBoxShown = false;

private bool isMessageBoxShowPoint = false;

private List<string> FuelList = new List<string>() { "Бензин", "Дизельное топливо", "Электричество" };

private List<string> TypeConsuption = new List<string> { "Городской", "По трассе", "Смешанный" };

private List<Car> CarList = new List<Car>();

private List<Fuel> PriceList = new List<Fuel>();

private List<Route> AllRoutes = new List<Route>();

private List<string> PointA = new List<string>();

private List<string> PointB = new List<string>();

public void InitList()//инициализация списков

{

CarList.Clear();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Car.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

Car car = new Car();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "name")

{

car.Name = childnode.InnerText;

}

if (childnode.Name == "year")

{

car.Year = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "typeCar")

{

car.TypeCar = childnode.InnerText;

}

if (childnode.Name == "maxSpeed")

{

car.MaxSpeed = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "seatingCapacity")

{

car.SeatingCapacity = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "fuel")

{

car.Fuel = childnode.InnerText;

}

if (childnode.Name == "fuelOctan")

{

car.FuelOctan = childnode.InnerText;

}

if (childnode.Name == "fuelConsumptionGeneral")

{

car.FuelConsumptionGeneral = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionCity")

{

car.FuelConsumptionCity = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionHighway")

{

car.FuelConsumptionHighway = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "enginePower")

{

car.EnginePower = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "tankSize")

{

car.TankSize = Convert.ToDouble(childnode.InnerText);

}

}

CarList.Add(car);

}

CarList = CarList.OrderBy(car => car.Name).ToList();

PriceList.Clear();

XmlDocument xDocTwo = new XmlDocument();

xDocTwo.Load("Fuel.xml");

XmlElement xRootTwo = xDocTwo.DocumentElement;

foreach (XmlNode xnode in xRootTwo)

{

Fuel fuel = new Fuel();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "name")

{

fuel.name = childnode.InnerText;

}

if (childnode.Name == "octaneNumber")

{

fuel.octaneNumber = childnode.InnerText;

}

if (childnode.Name == "price")

{

fuel.price = Convert.ToDouble(childnode.InnerText);

}

}

PriceList.Add(fuel);

}

AllRoutes = Route.ReadRousteInXML();

foreach (var item in AllRoutes)

{

if (!PointA.Contains(item.PointA))

{

PointA.Add(item.PointA);

}

if (!PointA.Contains(item.PointB))

{

PointA.Add(item.PointB);

}

if (!PointB.Contains(item.PointA))

{

PointB.Add(item.PointA);

}

if (!PointB.Contains(item.PointB))

{

PointB.Add(item.PointB);

}

}

PointA = PointA.OrderBy(item => item).ToList();

PointB = PointB.OrderBy(item => item).ToList();

}

public void InitComboBox() // инициализация comboBox

{

foreach (var item in FuelList)

{

comboBoxFuelType.Items.Add(item);

}

foreach (var item in TypeConsuption)

{

comboBoxConsumption.Items.Add(item);

}

foreach (var item in CarList)

{

comboBoxCar.Items.Add(item.Name);

}

comboBoxConsumption.IsEnabled = false;

foreach (var item in PointA)

{

comboBoxPointOne.Items.Add(item);

}

foreach (var item in PointB)

{

comboBoxPointTwo.Items.Add(item);

}

}

private void comboBoxFuelType\_SelectionChanged\_1(object sender, SelectionChangedEventArgs e)// проверка значений comboBoxFuelType

{

if (comboBoxFuelType.SelectedIndex != -1)

{

LabelLitr.Content = "Литров";

string selectedItem = comboBoxFuelType.SelectedItem.ToString();

if (selectedItem == "Бензин")

{

comboBoxFuelOctan.ToolTip = "Октановое число";

comboBoxFuelOctan.ItemsSource = GetFuelList("Бензин");

LabelConsumption.Content = "Литров";

}

else if (selectedItem == "Дизельное топливо")

{

comboBoxFuelOctan.ToolTip = "Октановое число";

comboBoxFuelOctan.ItemsSource = GetFuelList("Дизельное топливо");

LabelConsumption.Content = "Литров";

}

else if (selectedItem == "Электричество")

{

comboBoxFuelOctan.ToolTip = "Вид зарядки";

comboBoxFuelOctan.ItemsSource = GetFuelList("Электричество");

LabelConsumption.Content = "Вт-ч";

LabelLitr.Content = "Вт-ч";

}

comboBoxFuelOctan.SelectedIndex = 1;

}

}

private string SetValue(ComboBox comboBox)//установка значений

{

if (comboBox.SelectedIndex != -1)

{

return comboBox.SelectedItem.ToString();

}

else { return "Неизвестно"; }

}

private void ButtonCalculate\_Click(object sender, RoutedEventArgs e)//обработчик нажатия на клавишу для начала расчета

{

if (ValidValue())

{

// Получаем значения из комбо-боксов

string car = SetValue(comboBoxCar);

string PointA = SetValue(comboBoxPointOne);

string PointB = SetValue(comboBoxPointTwo);

// Получаем выбранный тип топлива

string fuelType = comboBoxFuelType.SelectedItem.ToString();

// Получаем числовые значения из текстовых полей

double dictance = Convert.ToDouble(this.textBoxDistance.Text);

double fuelPrice = Convert.ToDouble(this.textBoxFuelPrice.Text);

double consimption = Convert.ToDouble(this.textBoxConsumption.Text);

double averageSpeed = Convert.ToDouble(this.textBoxAverSpeed.Text);

// Проверяем условие и увеличиваем расход топлива, если средняя скорость превышает 140 км/ч

if (averageSpeed > 140 && car == "Неизвестно")

{

consimption += 1;

}

// Выполняем необходимые расчеты

double usedFuel = Math.Round(dictance / consimption);

double result = dictance / averageSpeed;

double fullPrice = Math.Round((dictance / 100) \* fuelPrice \* consimption, 2);

// Получаем выбранную дату

DateTime dateOne = DataPickerFirstData.SelectedDate.Value;

string date = dateOne.ToString("dd.MM.yyyy");

// Выводим результаты в соответствующие текстовые поля

textBoxUsedFuel.Text = usedFuel.ToString();

textBoxUsedFuel.BorderBrush = Brushes.Green;

textBoxTime.Text = Math.Round(result, 2).ToString();

textBoxTime.BorderBrush = Brushes.Green;

textBoxPrice.Text = fullPrice.ToString();

textBoxPrice.BorderBrush = Brushes.Green;

// Создаем объект UsersRoutes и добавляем маршрут в XML

UsersRoutes route = new UsersRoutes(UserName, car, PointA, PointB, dictance, fullPrice, fuelType, date, usedFuel, averageSpeed);

if (route.AddRoutesInXML())

{

// Предлагаем сохранить маршрут в docx файле

MessageBoxResult message = MessageBox.Show($"Поездка оформлена. Желаете сохранить данную поезку в docx файле?", "Подтверждение", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (message == MessageBoxResult.OK)

{

CreateDocxFile(dateOne, car, PointA, PointB, dictance, fullPrice, fuelType, usedFuel, averageSpeed);

}

}

else

{

MessageBox.Show("Поездка не оформлена", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Заполните все поля корректными значениями", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void CreateDocxFile(DateTime reportDate, string car, string PointA, string PointB, double dictance, double fullPrice, string fuelType, double usedFuel, double AverageSpeed)// создание Docx файла

{

try

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Документ Word (\*.docx)|\*.docx";

saveFileDialog.FileName = $"Отчет о поездке на {reportDate:dd.MM.yyyy}.docx";

if (saveFileDialog.ShowDialog() == true)

{

string fileName = saveFileDialog.FileName;

// Создание документа Word

using (DocX document = DocX.Create(fileName))

{

string reportTitle = $"Отчет о поездке на {reportDate:dd.MM.yyyy}";

document.InsertParagraph(reportTitle).FontSize(12d).Bold().Alignment = Alignment.center;

// Добавление информации о средней стоимости, жидком топливе и электричестве

string tripDescription = $"Пользователь {UserName} {Surname} совершил поездку";

if (car != "Неизвестно")

{

tripDescription += $" на {car}";

}

if (PointA != "Неизвестно" && PointB != "Неизвестно")

{

tripDescription += $" из {PointA} - {PointB}";

}

tripDescription += $", дистанция маршрута составила {dictance}.";

document.InsertParagraph(tripDescription).FontSize(12d);

document.InsertParagraph($"Поездка обошлась в {fullPrice:0.00} рублей.").FontSize(12d);

if (fuelType == "Электричесво")

{

document.InsertParagraph($"Во время поездки было потрачено {fuelType} {usedFuel} Киловатт-часов.").FontSize(12d);

}

else

{

document.InsertParagraph($"Во время поездки было потрачено {fuelType} {usedFuel} Литров.").FontSize(12d);

}

document.InsertParagraph($"Средняя скорость поездки {AverageSpeed} Км/ч.").FontSize(12d);

document.InsertParagraph($"Дата формирования отчета {DateTime.Now:dd.MM.yyyy}").FontSize(12d).Alignment = Alignment.right;

// Сохранение документа

document.Save();

}

MessageBox.Show("Отчет успешно создан!", "Создание отчета", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (System.IO.IOException ex)

{

MessageBox.Show("Отчет не создан. Закройте открытый отчет и повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void comboBoxFuelOctan\_SelectionChanged(object sender, SelectionChangedEventArgs e)// проверка выбора в comboBoxFuelOctan

{

if (comboBoxFuelOctan.SelectedIndex != -1)

{

string selectedItem = comboBoxFuelOctan.SelectedItem.ToString();

FuelPrice(selectedItem);

textBoxFuelPrice.IsReadOnly = true;

}

else

{

//textBoxFuelPrice.BorderBrush = Brushes.Gray;

textBoxFuelPrice.IsReadOnly = false;

}

}

public void FuelPrice(string name)//вывод цены на топливо

{

foreach (var item in PriceList)

{

if (item.octaneNumber == name)

{

textBoxFuelPrice.Text = item.price.ToString();

textBoxFuelPrice.BorderBrush = Brushes.Green;

break;

}

}

}

public void ConsumptionCar(string text, string type)//выбор типа потребления топлива

{

foreach (var item in CarList)

{

if (item.Name == text)

{

if (type == "Городской")

{

textBoxConsumption.Text = item.FuelConsumptionCity.ToString();

break;

}

else if (type == "По трассе")

{

textBoxConsumption.Text = item.FuelConsumptionHighway.ToString();

break;

}

else

{

textBoxConsumption.Text = item.FuelConsumptionGeneral.ToString();

break;

}

}

}

}

private void comboBoxConsumption\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxConsumption.SelectedIndex != -1 && comboBoxCar.SelectedIndex != -1)

{

ConsumptionCar(comboBoxCar.SelectedItem.ToString(), comboBoxConsumption.SelectedItem.ToString());

isMessageBoxShown = false;

}

else

{

if (!isMessageBoxShown)

{

MessageBox.Show("Выберите автомобиль или не используйте данное поле", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Information);

isMessageBoxShown = true;

}

comboBoxConsumption.SelectedIndex = -1;

}

}

private void comboBoxCar\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxCar.SelectedIndex != -1)

{

foreach (var item in CarList)

{

if (item.Name == comboBoxCar.SelectedItem.ToString())

{

textBoxConsumption.Text = item.FuelConsumptionGeneral.ToString();

textBoxConsumption.BorderBrush = Brushes.Green;

comboBoxConsumption.SelectedIndex = 2;

comboBoxFuelType.SelectedIndex = FuelList.IndexOf(item.Fuel);

comboBoxFuelOctan.SelectedIndex = comboBoxFuelOctan.Items.IndexOf(item.FuelOctan);

comboBoxFuelType.IsEnabled = false;

}

}

comboBoxConsumption.IsEnabled = true;

}

else

{

comboBoxFuelType.IsEnabled = true;

comboBoxConsumption.IsEnabled = false;

textBoxConsumption.Clear();

textBoxFuelPrice.Clear();

textBoxFuelPrice.BorderBrush = Brushes.Gray;

textBoxConsumption.BorderBrush = Brushes.Gray;

comboBoxConsumption.SelectedIndex = -1;

comboBoxFuelType.SelectedIndex = -1;

comboBoxFuelOctan.SelectedIndex = -1;

}

}

private void MenuItem\_Click(object sender, RoutedEventArgs e)//очистка всех полей

{

isMessageBoxShown = true;

textBoxDistance.Clear();

textBoxAverSpeed.Clear();

textBoxTime.Clear();

textBoxConsumption.Clear();

textBoxPrice.Clear();

textBoxFuelPrice.Clear();

textBoxUsedFuel.Clear();

textBoxDistance.BorderBrush = Brushes.Gray;

textBoxFuelPrice.BorderBrush = Brushes.Gray;

textBoxConsumption.BorderBrush = Brushes.Gray;

comboBoxFuelType.SelectedIndex = -1;

comboBoxFuelOctan.SelectedIndex = -1;

comboBoxCar.SelectedIndex = -1;

comboBoxConsumption.SelectedIndex = -1;

comboBoxPointOne.SelectedIndex = -1;

comboBoxPointTwo.SelectedIndex = -1;

}

private static List<string> GetFuelList(string type) // получение названий топлива

{

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

XmlDocument doc = new XmlDocument();

doc.Load("Fuel.xml");

XmlNodeList fuelNodes = doc.GetElementsByTagName("fuel");

foreach (XmlNode fuelNode in fuelNodes)

{

XmlNode nameNode = fuelNode.SelectSingleNode("name");

if (nameNode.InnerText == type)

{

XmlNode octaneNode = fuelNode.SelectSingleNode("octaneNumber");

Fuel.Add(octaneNode.InnerText);

}

}

return Fuel;

}

private void comboBoxPointOne\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1 && comboBoxPointOne.SelectedItem != comboBoxPointTwo.SelectedItem)

{

string A = comboBoxPointOne.SelectedItem.ToString();

string B = comboBoxPointTwo.SelectedItem.ToString();

isMessageBoxShowPoint = false;

if (A != B)

{

foreach (var item in AllRoutes)

{

if ((item.PointA == A && item.PointB == B) || (item.PointA == B && item.PointB == A))

{

this.textBoxDistance.Text = item.Distance.ToString();

textBoxDistance.BorderBrush = Brushes.Green;//

isMessageBoxShowPoint = true;

textBoxDistance.IsReadOnly = true;////

break;

}

}

if (!isMessageBoxShowPoint)

{

MessageBox.Show("Выбранный маршрут не найден. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointOne.SelectedIndex = -1;

textBoxDistance.Clear();

textBoxDistance.IsReadOnly = false;////

}

}

}

else if (comboBoxPointOne.SelectedIndex == -1 || comboBoxPointTwo.SelectedIndex == -1)

{

textBoxDistance.Clear();

textBoxDistance.BorderBrush = Brushes.Gray;

textBoxDistance.IsReadOnly = false;////

}

else if (comboBoxPointOne.SelectedIndex != -1 || comboBoxPointTwo.SelectedIndex != -1)

{

MessageBox.Show("Выберите корректное место назначения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointOne.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

private void comboBoxPointTwo\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1 && comboBoxPointOne.SelectedItem != comboBoxPointTwo.SelectedItem)

{

string A = comboBoxPointOne.SelectedItem.ToString();

string B = comboBoxPointTwo.SelectedItem.ToString();

isMessageBoxShowPoint = false;

if (A != B)

{

foreach (var item in AllRoutes)

{

if ((item.PointA == A && item.PointB == B) || (item.PointA == B && item.PointB == A))

{

textBoxDistance.Text = item.Distance.ToString();

textBoxDistance.BorderBrush = Brushes.Green;

isMessageBoxShowPoint = true;

textBoxDistance.IsReadOnly = true;///

break;

}

}

if (!isMessageBoxShowPoint)

{

MessageBox.Show("Выбранный маршрут не найден. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointTwo.SelectedIndex = -1;

textBoxDistance.Clear();

textBoxDistance.IsReadOnly = false;////

}

}

}

else if (comboBoxPointOne.SelectedIndex == -1 || comboBoxPointTwo.SelectedIndex == -1)

{

textBoxDistance.Clear();

textBoxDistance.BorderBrush = Brushes.Gray;

textBoxDistance.IsReadOnly = false;////

}

else if (comboBoxPointOne.SelectedIndex != -1 || comboBoxPointTwo.SelectedIndex != -1)

{

MessageBox.Show("Выберите корректное место назначения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointTwo.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

private bool ValidValue()//проверка значений

{

double distance, averageSpeed, consumption, fuelPrice;

if (IsValidDoubleInput(textBoxDistance, 0, 10000, out distance) &

IsValidDoubleInput(textBoxAverSpeed, 0, 400, out averageSpeed) &

IsValidDoubleInput(textBoxConsumption, 0, 100, out consumption) &

IsValidDoubleInput(textBoxFuelPrice, 0, 1000, out fuelPrice) &

IsValidDataInput(DataPickerFirstData))

{

return true;

}

else

{

return false;

}

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)//проверка корректности вещественного значения в textBox

{

if (box.Text == "")

{

value = 0;

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private bool IsValidDataInput(DatePicker picker) // проверка корректности даты

{

if (picker.SelectedDate.HasValue & picker.SelectedDate < DateTime.Now)

{

picker.BorderBrush = Brushes.Gray;

return true;

}

else

{

picker.BorderBrush = Brushes.Red;

return false;

}

}

private string FixStr(string input)//замена . на ,

{

return input.Replace('.', ',');

}

private void MenuItem\_Click\_1(object sender, RoutedEventArgs e)//открытие окна с графиком

{

GraphForm graphForm = new GraphForm(UserName, Surname);

graphForm.ShowDialog();

}

private void comboBoxPointTwo\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка comboBox двойным нажатием

{

comboBoxPointTwo.SelectedIndex = -1;

}

private void comboBoxPointOne\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка comboBox двойным нажатием

{

comboBoxPointOne.SelectedIndex = -1;

}

private void comboBoxCar\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка comboBox двойным нажатием

{

comboBoxCar.SelectedIndex = -1;

}

private void MenuItem\_Click\_2(object sender, RoutedEventArgs e)//открытие справки

{

Process.Start("hh.exe", "HelpBook.chm");

}

private void comboBoxFuelType\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка comboBox двойным нажатием

{

if (comboBoxCar.SelectedIndex == -1)

{

comboBoxFuelType.SelectedIndex = -1;

comboBoxFuelOctan.SelectedIndex = -1;

textBoxFuelPrice.Clear();

textBoxFuelPrice.BorderBrush = Brushes.Gray;

LabelLitr.Content = "Литров";

LabelConsumption.Content = "Литров";

}

}

}

}

<Window x:Class="JourneyExpense.AdminForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Окно администратора"

Height="450" MinHeight="450" MaxHeight="450"

Width="800" MinWidth="800" MaxWidth="800"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="32\*"/>

<RowDefinition Height="145\*"/>

<RowDefinition Height="40\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button x:Name="AddCarButton" Content="Добавить автомобиль" Margin="20,10,20,0" Click="AddCarButton\_Click" FontSize="11" Height="39" VerticalAlignment="Top"/>

<Button x:Name="AddRoutesButton" Content="Добавить маршрут" Margin="20,10,20,0" VerticalAlignment="Top" Height="39" Click="AddRoutesButton\_Click" Grid.Column="1" FontSize="11"/>

<Button x:Name="EditFuelButton" Content="Редактировать топливо" Margin="20,10,20,0" VerticalAlignment="Top" Height="39" Click="EditFuelButton\_Click" Grid.Column="2" FontSize="12" Grid.Row="1"/>

<Button x:Name="AddFuelButton" Content="Добавить топливо" Margin="20,0,20,0" VerticalAlignment="Center" Height="39" Grid.Column="2" Click="AddFuelButton\_Click" FontSize="11"/>

<Button x:Name="CalculateButton" Grid.Column="3" Content="Рассчитать потребление" Click="CalculateButton\_Click" Height="39" VerticalAlignment="Top" FontSize="11" Margin="20,10,20,0"/>

<Button x:Name="EditCarButton" Content="Редактировать автомобиль" Margin="20,10,20,0" Height="39" VerticalAlignment="Top" Grid.Row="1" Click="EditCarButton\_Click"/>

<Button x:Name="EditRoutesButton" Content="Редактировать Маршрут" Margin="20,10,20,0" FontSize="12" Height="39" VerticalAlignment="Top" Grid.Row="1" Grid.Column="1" Click="EditRoutesButton\_Click"/>

</Grid>

</Window>

using JourneyExpense.EditWindows;

using System.Windows;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для AdminForm.xaml

/// </summary>

public partial class AdminForm : Window

{

public AdminForm()

{

InitializeComponent();

}

private void AddCarButton\_Click(object sender, RoutedEventArgs e) // обработчик нажатия на кнопку для октрытия формы для добавления автомобиля

{

AddCarForm addCarForm = new AddCarForm();

addCarForm.Show();

}

private void EditFuelButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку откртытия формы для редактирования топлива

{

UpdatePriceForm updatePriceForm = new UpdatePriceForm();

updatePriceForm.Show();

}

private void AddRoutesButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку для октрытия формы для добавления маршрута

{

AddRoutesForm routesForm = new AddRoutesForm();

routesForm.Show();

}

private void AddFuelButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку для октрытия формы для добавления топлива

{

AddFuelForm fuelForm = new AddFuelForm();

fuelForm.Show();

}

private void CalculateButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку для октрытия формы для открытия главного окна

{

MainWindow adminMain = new MainWindow("Admin","Admin");

adminMain.Show();

}

private void EditCarButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку откртытия формы для редактирования автомобиля

{

EditCarForm editCar = new EditCarForm();

editCar.Show();

}

private void EditRoutesButton\_Click(object sender, RoutedEventArgs e)// обработчик нажатия на кнопку откртытия формы для редактирования маршрута

{

EditRouteForm editRoute = new EditRouteForm();

editRoute.Show();

}

}

}

<Window x:Class="JourneyExpense.AddCarForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Добавление автомобиля"

Height="450" MinHeight="450" MaxHeight="450"

Width="800" MinWidth="800" MaxWidth="800"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="49\*"/>

<RowDefinition Height="47\*"/>

<RowDefinition Height="47\*"/>

<RowDefinition Height="16\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="25\*"/>

<ColumnDefinition Width="180\*"/>

<ColumnDefinition Width="227\*"/>

<ColumnDefinition Width="187\*"/>

<ColumnDefinition Width="181"/>

</Grid.ColumnDefinitions>

<TextBox x:Name="textBoxName" FontSize="15" Grid.Column="2" Grid.Row="1" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

<Label Content="Марка автомобиля" Grid.Column="1" Grid.Row="1" Margin="10,10,10,0" Height="29" VerticalAlignment="Top" FontSize="15"/>

<TextBox x:Name="textBoxYear" FontSize="15" Grid.Column="2" Grid.Row="2" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

<Label Content="Год выпуска" Grid.Column="1" Grid.Row="2" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<Label x:Name="LabelTypeCar" Content="Тип кузова автомобиля" Grid.Column="1" Grid.Row="4" Margin="5,9,0,0" Height="29" VerticalAlignment="Top" FontSize="15" HorizontalContentAlignment="Center" HorizontalAlignment="Left" Width="175"/>

<TextBox x:Name="textBoxPlace" FontSize="15" Grid.Column="2" Grid.Row="5" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

<Label Content="Кол-во мест" Grid.Column="1" Grid.Row="5" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<Label x:Name="LabelTypeFuel" Content="Вид топлива" Grid.Column="1" Grid.Row="6" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<Label x:Name="OctanLabel" Content="Октановое число" Grid.Column="1" Grid.Row="7" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<Label Content="Расход топлива" Grid.Column="1" Grid.Row="8" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<TextBox x:Name="textBoxFuelConsumptionGeneral" Grid.Column="2" Margin="10,10,0,10" Grid.Row="8" FontSize="15" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Смешанный расход

</TextBox.ToolTip>

</TextBox>

<TextBox x:Name="textBoxFuelConsumptionCity" Grid.Column="2" Grid.Row="8" FontSize="15" Margin="85,10,0,10" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Расход в городе

</TextBox.ToolTip>

</TextBox>

<TextBox x:Name="textBoxFuelConsumptionHighway" Grid.Column="2" Grid.Row="8" FontSize="15" Margin="160,10,0,10" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Расход за городом

</TextBox.ToolTip>

</TextBox>

<ComboBox x:Name="comboBoxFuelType" Grid.Column="2" Grid.Row="6" VerticalAlignment="Center" Height="27" Margin="10,0,10,0" SelectionChanged="comboBoxFuelType\_SelectionChanged"/>

<ComboBox x:Name="comboBoxOctan" Grid.Column="2" Grid.Row="7" VerticalAlignment="Center" Height="27" Margin="10,0,10,0"/>

<ComboBox x:Name="comboBoxCarType" Grid.Column="2" Grid.Row="4" VerticalAlignment="Center" Height="27" Margin="10,0,10,0"/>

<Button Grid.Column="3" Content="Добавить автомобиль" HorizontalAlignment="Center" Grid.Row="3" VerticalAlignment="Bottom" Click="Button\_Click" Height="27" Width="165" Margin="0,0,0,11"/>

<Label Content="Мощность двигателя" Grid.Column="3" Grid.Row="1" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<TextBox x:Name="textBoxPower" FontSize="15" Grid.Column="4" Grid.Row="1" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

<Label x:Name="tank" Content="Объем бака" Grid.Column="3" Grid.Row="2" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<TextBox x:Name="textBoxSizeTank" FontSize="15" Grid.Column="4" Grid.Row="2" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

<Label Content="Максимальная скорость" Grid.Column="1" Grid.Row="3" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="13"/>

<TextBox x:Name="textBoxMaxSpeed" FontSize="15" Grid.Column="2" Grid.Row="3" Margin="10,0,10,0" Height="27" VerticalAlignment="Center"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using JourneyExpense.Classes;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для AddCarForm.xaml

/// </summary>

public partial class AddCarForm : Window

{

public AddCarForm()

{

InitializeComponent();

InitComboBox();

}

private List<string> Fuel = new List<string>() { "Бензин", "Дизельное топливо", "Электричество" };

private List<string> CarType = new List<string>() { "Седан", "Купе", "Хэтчбек", "Универсал", "Внедорожник", "Кроссовер", "Кабриолет", "Лифтбек", "Фургон", "Минивэн", "Пикап" };

public void InitComboBox()// иницилизцаии comboBox

{

comboBoxFuelType.ItemsSource = Fuel;

comboBoxCarType.ItemsSource = CarType;

}

private static List<string> GetFuelList(string type)//функция для получения списка топлива

{

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

XmlDocument doc = new XmlDocument();

doc.Load("Fuel.xml");

XmlNodeList fuelNodes = doc.GetElementsByTagName("fuel");

foreach (XmlNode fuelNode in fuelNodes)

{

XmlNode nameNode = fuelNode.SelectSingleNode("name");

if (nameNode.InnerText == type)

{

XmlNode octaneNode = fuelNode.SelectSingleNode("octaneNumber");

Fuel.Add(octaneNode.InnerText);

}

}

return Fuel;

}

private void comboBoxFuelType\_SelectionChanged(object sender, SelectionChangedEventArgs e) // обработчик изменение значения в comboBox

{

comboBoxOctan.Visibility = Visibility.Visible;

OctanLabel.Visibility = Visibility.Visible;

textBoxFuelConsumptionCity.Visibility = Visibility.Visible;

textBoxFuelConsumptionHighway.Visibility = Visibility.Visible;

comboBoxFuelType.IsReadOnly = false;

string selectedItem = comboBoxFuelType.SelectedItem.ToString();

if (selectedItem == "Бензин")

{

comboBoxOctan.ItemsSource = GetFuelList("Бензин");

tank.Content = "Объем бака";

}

else if (selectedItem == "Дизельное топливо")

{

comboBoxOctan.ItemsSource = GetFuelList("Дизельное топливо");

comboBoxFuelType.IsReadOnly = true;

tank.Content = "Объем бака";

}

else if (selectedItem == "Электричество")

{

comboBoxOctan.ItemsSource = null;

comboBoxOctan.SelectedItem = null;

comboBoxOctan.Visibility = Visibility.Hidden;

OctanLabel.Visibility = Visibility.Hidden;

textBoxFuelConsumptionCity.Visibility = Visibility.Hidden;

textBoxFuelConsumptionHighway.Visibility = Visibility.Hidden;

textBoxFuelConsumptionCity.Clear();

textBoxFuelConsumptionHighway.Clear();

tank.Content = "Объем батареи";

}

comboBoxOctan.SelectedIndex = 1;

}

private bool ValidValue() // проверка валидности значений

{

if (TextBoxValid(textBoxName) & ValidFuel() & СonsumptionСheck() & LogicCheck())

{

return true;

}

return false;

}

private bool LogicCheck() // логическая проверка значений в полях

{

string selectedItem = IsValidComboBox(comboBoxCarType);

int year, speed, place, minSeats = 2, maxSeats = 5;

bool IsYear, IsSpeed, IsSeats;

IsYear = IsValidIntInput(textBoxYear, 1980, 2024, out year);

IsSpeed = IsValidIntInput(textBoxMaxSpeed, 0, 500, out speed);

IsSeats = IsValidIntInput(textBoxPlace, 0, 11, out place);

if (selectedItem == "Фургон")

{

minSeats = 2;

maxSeats = 10;

}

else if (selectedItem == "Универсал" || selectedItem == "Внедорожник" || selectedItem == "Кроссовер" || selectedItem == "Минивэн")

{

minSeats = 5;

maxSeats = 8;

}

if (place >= minSeats && place <= maxSeats)

{

textBoxPlace.BorderBrush = Brushes.Gray;

if (IsYear && IsSpeed && IsSeats)

{

comboBoxCarType.BorderBrush = Brushes.Gray;

return true;

}

return false;

}

else

{

if (selectedItem == "Ошибка")

{

textBoxPlace.BorderBrush = Brushes.Red;

return false;

}

MessageBox.Show($"Ошибка количества мест. Требуется от {minSeats} до {maxSeats} мест для автомобиля типа {selectedItem}");

textBoxPlace.BorderBrush = Brushes.Red;

return false;

}

}

private bool СonsumptionСheck() // проверка потребления

{

string selectedItem = IsValidComboBox(comboBoxFuelType);

double ConsumptionGeneral, ConsumptionCity, ConsumptionHighway, enginePower, engineSize, tankSize;

bool isNumericOne = IsValidDoubleInput(textBoxFuelConsumptionGeneral, 0, 100, out ConsumptionGeneral);

bool isNumericTwo = IsValidDoubleInput(textBoxFuelConsumptionCity, 0, 100, out ConsumptionCity);

bool isNumericThree = IsValidDoubleInput(textBoxFuelConsumptionHighway, 0, 100, out ConsumptionHighway);

bool isNumericFour = IsValidDoubleInput(textBoxPower, 0, 2000, out enginePower);

bool isNumericSix = IsValidDoubleInput(textBoxSizeTank, 0, 600, out tankSize);

if (selectedItem == "Бензин" || selectedItem == "Дизельное топливо")

{

if (isNumericOne && isNumericTwo && isNumericThree && isNumericFour && isNumericSix)

{

textBoxFuelConsumptionGeneral.BorderBrush = Brushes.Gray;

return true;

}

}

else if (selectedItem == "Электричество")

{

if (isNumericOne && isNumericFour && isNumericSix)

{

textBoxFuelConsumptionGeneral.BorderBrush = Brushes.Gray;

return true;

}

textBoxFuelConsumptionGeneral.BorderBrush = Brushes.Red;

}

return false;

}

private bool ValidFuel() // проверка корректности выбора топлива

{

string selectedItem = IsValidComboBox(comboBoxFuelType);

if (selectedItem == "Бензин" || selectedItem == "Дизельное топливо")

{

if (comboBoxFuelType.SelectedIndex != -1 && textBoxFuelConsumptionGeneral.Text != "" && textBoxFuelConsumptionCity.Text != "" && textBoxFuelConsumptionHighway.Text != "")

{

return true;

}

}

else

{

if (comboBoxFuelType.SelectedIndex != -1 && textBoxFuelConsumptionGeneral.Text != "")

{

return true;

}

}

return false;

}

private void Button\_Click(object sender, RoutedEventArgs e) // обработчик нажатия на кнопку

{

if (ValidValue())

{

string name = this.textBoxName.Text;

int year = Convert.ToInt32(this.textBoxYear.Text);

string typeCar = this.comboBoxCarType.SelectedItem.ToString();

int place = Convert.ToInt32(this.textBoxPlace.Text);

int maxSpeed = Convert.ToInt32(this.textBoxMaxSpeed.Text);

string typeFuel = this.comboBoxFuelType.SelectedItem.ToString();

double fuelConsumptionGeneral = DoubleNull(textBoxFuelConsumptionGeneral);

double fuelConsumptionCity = DoubleNull(textBoxFuelConsumptionCity);

double fuelConsumptionHighway = DoubleNull(textBoxFuelConsumptionHighway);

double enginePower = Convert.ToDouble(FixStr(this.textBoxPower.Text));

double tankSize = Convert.ToDouble(FixStr(this.textBoxSizeTank.Text));

Car car;

if (typeFuel == "Электричество")

{

car = new Car(name, year, typeCar, maxSpeed, place, typeFuel, "Медл.", fuelConsumptionGeneral, fuelConsumptionGeneral, fuelConsumptionGeneral, enginePower, tankSize);

}

else

{

string typeOctan = comboBoxOctan.SelectedItem.ToString();

car = new Car(name, year, typeCar, maxSpeed, place, typeFuel, typeOctan, fuelConsumptionGeneral, fuelConsumptionCity, fuelConsumptionHighway, enginePower, tankSize);

}

if (car.AddCarInXML())

{

MessageBox.Show($"Автомобиль: {name} был добавлен в систему.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show($"Автомобиль: {name} не был добавлен в систему. Автомобиль уже находится в системе.","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Некорректные данные","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private double DoubleNull(TextBox text) // проверка пустоты текстовых полей

{

if (text.Text != "")

{

return Convert.ToDouble(FixStr(text.Text));

}

else

{

return 0;

}

}

private string FixStr(string input) // замена точки на запятую

{

return input.Replace('.', ',');

}

private bool IsValidIntInput(TextBox text, int min, int max, out int value) // проверка корректности введенных целочисленных значений

{

if (text.Text == "")

{

value = 0;

text.BorderBrush = Brushes.Red; // смена цвета границы элемента

return false;

}

bool isNumeric = int.TryParse(text.Text, out value);

if (isNumeric)

{

if (value > min && value < max)

{

text.BorderBrush = Brushes.Gray;

return true;

}

else

{

text.BorderBrush = Brushes.Red;

return false;

}

}

else

{

text.BorderBrush = Brushes.Red;

return false;

}

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)// проверка корректности введенных вещественное значений

{

if (box.Text == "")

{

value = 0;

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private string IsValidComboBox(ComboBox box)// проверка корректности выбора значения в comboBox

{

if (box.SelectedIndex == -1)

{

return "Ошибка";

}

else

{

return box.SelectedItem.ToString();

}

}

private bool TextBoxValid(TextBox text)// проверка значения в textBox

{

if (text.Text == "")

{

text.BorderBrush = Brushes.Red;

return false;

}

else

{

text.BorderBrush = Brushes.Gray;

return true;

}

}

}

}<Window x:Class="JourneyExpense.AddFuelForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Добавление топлива"

Height="260" MinHeight="260" MaxHeight="260"

Width="400" MinWidth="400" MaxWidth="400"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="13\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="51\*"/>

<RowDefinition Height="35\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="Вид топилива" Margin="18,0,0,0" FontSize="15" Grid.Row="1" Height="29" VerticalAlignment="Center" HorizontalAlignment="Left" Width="164"/>

<ComboBox x:Name="comboBoxFuelType" Grid.Row="1" VerticalAlignment="Center" Height="29" Grid.Column="1" Margin="18,0,18,0"/>

<Label Content="Тип топилива" Margin="18,0,0,0" FontSize="15" Grid.Row="2" Height="29" VerticalAlignment="Center" HorizontalAlignment="Left" Width="164"/>

<Label Content="Цена на литр топлива" FontSize="15" Grid.Row="3" Height="29" VerticalAlignment="Center" HorizontalAlignment="Center" Width="164"/>

<TextBox x:Name="textBoxPrice" Grid.Column="1" Grid.Row="3" VerticalAlignment="Center" Height="28" Margin="18,0,18,0" FontSize="20"/>

<Button x:Name="UpdatePriceButton" Content="Добавить топливо" HorizontalAlignment="Left" Grid.Row="4" VerticalAlignment="Center" Height="28" Width="164" Click="UpdatePriceButton\_Click" Margin="18,0,0,0"/>

<TextBox x:Name="textBoxFuelOctan" Grid.Column="1" Grid.Row="2" VerticalAlignment="Center" Height="28" Margin="18,0,18,0" FontSize="20"/>

</Grid>

</Window>

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using JourneyExpense.Classes;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для AddFuelForm.xaml

/// </summary>

public partial class AddFuelForm : Window

{

public AddFuelForm()

{

InitializeComponent();

InitComboBox();

}

private List<string> FuelType = new List<string>() { "Бензин", "Дизельное топливо", "Электричество" };

public void InitComboBox()// инициализацтя comboBox

{

foreach (var item in FuelType)

{

comboBoxFuelType.Items.Add(item);

}

}

private void UpdatePriceButton\_Click(object sender, RoutedEventArgs e)//обновление цена на топливо

{

double price = 0;

if (comboBoxFuelType.SelectedIndex != -1 & TextBoxValid(textBoxFuelOctan) & IsValidDoubleInput(textBoxPrice, 0, 100, out price))

{

string name = comboBoxFuelType.SelectedItem.ToString();

string octaneNumber = textBoxFuelOctan.Text.TrimEnd();

Fuel fuel = new Fuel(name, octaneNumber.ToUpper(), price);

if (fuel.AddFuelInXML())

{

MessageBox.Show($"Топливо добавлено в систему.","Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show($"Топливо уже находится в системе.","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Введите корректные данные","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)//проверка корректности ввода вещественного значения

{

if (box.Text == "")

{

value = 0;

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private string FixStr(string input)// замена . на ,

{

return input.Replace('.', ',');

}

private bool TextBoxValid(TextBox text)// проверка корректности textBox

{

if (text.Text == "")

{

text.BorderBrush = Brushes.Red;

return false;

}

else

{

text.BorderBrush = Brushes.Gray;

return true;

}

}

}

}

<Window x:Class="JourneyExpense.AddRoutesForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Добавление маршрута"

Height="260" MinHeight="260" MaxHeight="280"

Width="400" MinWidth="400" MaxWidth="280"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="13\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="51\*"/>

<RowDefinition Height="36\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBox x:Name="textBoxPointA" FontSize="15" Margin="10,12,10,0" Height="27" VerticalAlignment="Top" Grid.Column="1" Grid.Row="1"/>

<Label Content="Точка отправления" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="1"/>

<TextBox x:Name="textBoxPointB" FontSize="15" Margin="10,0,10,0" Height="27" VerticalAlignment="Center" Grid.Row="2" Grid.Column="1"/>

<Label Content="Точка назначения" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="2"/>

<TextBox x:Name="textBoxDistance" FontSize="15" Margin="10,0,0,0" Height="27" VerticalAlignment="Center" Grid.Row="3" Grid.Column="1" Width="145" HorizontalAlignment="Left"/>

<Label Content="Расстояние" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="3"/>

<Button x:Name="ButtonAddRoute" Content="Добавить маршрут" Margin="18,0,18,0" Click="ButtonAddRoute\_Click" Grid.Row="4" Height="28" VerticalAlignment="Center" />

<Label Grid.Column="1" Content="Км" HorizontalAlignment="Left" Margin="160,0,0,0" Grid.Row="3" VerticalAlignment="Center"/>

</Grid>

</Window>

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using JourneyExpense.Classes;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для AddRoutesForm.xaml

/// </summary>

public partial class AddRoutesForm : Window

{

public AddRoutesForm()

{

InitializeComponent();

}

private void ButtonAddRoute\_Click(object sender, RoutedEventArgs e)//добавление маршурта

{

string pointA = textBoxPointA.Text.TrimEnd();

string pointB = textBoxPointB.Text.TrimEnd();

double distance = 0;

bool IsValidValue = IsValidDoubleInput(textBoxDistance, 0, 40000, out distance);

if ((TextBoxValid(textBoxPointA) & TextBoxValid(textBoxPointB) & IsValidValue) & (pointA != pointB))

{

Route route = new Route(pointA, pointB, distance);

if (route.AddRoutesInXML())

{

MessageBox.Show($"Маршрут {pointA} - {pointB} добавлен в систему.","Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show($"Маршрут {pointA} - {pointB} не добавлен, так как уже есть в системе.","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Ошибка в маршруте, проверьте маршрут и повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)//проверка вещественных значений

{

if (box.Text == "")

{

value = 0;

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private bool TextBoxValid(TextBox text)//проверка кооректносоти данных в textBox

{

if (text.Text == "")

{

text.BorderBrush = Brushes.Red;

return false;

}

else

{

text.BorderBrush = Brushes.Gray;

return true;

}

}

private string FixStr(string input)//замена . на ,

{

return input.Replace('.', ',');

}

}

}

<Window x:Class="JourneyExpense.EditCarForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Редактирование атвомобиля"

Height="280" MinHeight="280" MaxHeight="280"

Width="435" MinWidth="435" MaxWidth="435"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="10\*"/>

<RowDefinition Height="44\*"/>

<RowDefinition Height="44\*"/>

<RowDefinition Height="44\*"/>

<RowDefinition Height="44\*"/>

<RowDefinition Height="79\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="5\*"/>

<ColumnDefinition Width="37\*"/>

<ColumnDefinition Width="45\*"/>

</Grid.ColumnDefinitions>

<Label Content="Марка автомобиля" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Column="1" Grid.Row="1"/>

<ComboBox x:Name="comboBoxCar" VerticalAlignment="Center" Height="25" Margin="10,0,0,0" SelectionChanged="comboBoxCar\_SelectionChanged" Grid.Column="2" Grid.Row="1" HorizontalAlignment="Left" Width="205"/>

<TextBox x:Name="textBoxFuelConsumptionGeneral" Grid.Column="2" Margin="10,0,0,0" Grid.Row="3" FontSize="15" Height="27" VerticalAlignment="Center" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Смешанный расход

</TextBox.ToolTip>

</TextBox>

<TextBox x:Name="textBoxFuelConsumptionCity" Grid.Column="2" Grid.Row="3" FontSize="15" Margin="85,0,0,0" Height="27" VerticalAlignment="Center" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Расход в городе

</TextBox.ToolTip>

</TextBox>

<TextBox x:Name="textBoxFuelConsumptionHighway" Grid.Column="2" Grid.Row="3" FontSize="15" Margin="160,0,0,0" Height="27" VerticalAlignment="Center" HorizontalAlignment="Left" Width="55">

<TextBox.ToolTip>

Расход за городом

</TextBox.ToolTip>

</TextBox>

<Label Content="Расход топлива" Grid.Column="1" Grid.Row="3" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<Label x:Name="OctanLabel" Content="Октановое число" Grid.Column="1" Grid.Row="2" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<ComboBox x:Name="comboBoxOctan" Grid.Column="2" Grid.Row="2" VerticalAlignment="Center" Height="27" Margin="10,0,0,0" HorizontalAlignment="Left" Width="205"/>

<Label Content="Мощность двигателя" Grid.Column="1" Grid.Row="4" Margin="10,0,10,0" Height="29" VerticalAlignment="Center" FontSize="15"/>

<TextBox x:Name="textBoxPower" FontSize="15" Grid.Column="2" Grid.Row="4" Margin="10,0,70,0" Height="27" VerticalAlignment="Center"/>

<Button x:Name="UpdateCarButton" Content="Обновить данные" Grid.Row="5" VerticalAlignment="Top" Height="28" Margin="30,10,31,0" Grid.Column="2" Click="UpdateCarButton\_Click"/>

<Button x:Name="DeleteCarButton" Content="Удалить авто" Grid.Row="5" VerticalAlignment="Top" Height="21" Grid.Column="2" Margin="57,48,58,0" Click="DeleteCarButton\_Click"/>

<Label Grid.Column="2" Content="Л.С." HorizontalAlignment="Left" Margin="179,0,0,0" Grid.Row="4" VerticalAlignment="Center">

<Label.ToolTip>

Лошадиные силы

</Label.ToolTip>

</Label>

</Grid>

</Window>

using JourneyExpense.Classes;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для EditCarForm.xaml

/// </summary>

public partial class EditCarForm : Window

{

private List<Car> AllCar = new List<Car>();

private List<Fuel> AllFuel = new List<Fuel>();

private bool Type = false;

public EditCarForm()

{

InitializeComponent();

ReadData();

InitComboBox();

}

private void ReadData()//инициализация списков

{

AllCar = Car.ReadCarInXML();

AllCar = AllCar.OrderBy(car => car.Name).ToList();

AllFuel = Fuel.ReadFuelInXML();

}

private void InitComboBox()//инициализация comboBox

{

foreach (var item in AllCar)

{

comboBoxCar.Items.Add(item.Name);

}

}

private void comboBoxCar\_SelectionChanged(object sender, SelectionChangedEventArgs e)//обработчик изменения элементов в comboBox

{

if (comboBoxCar.SelectedIndex != -1)

{

foreach (var item in AllCar)

{

if (item.Name == comboBoxCar.SelectedItem.ToString())

{

textBoxFuelConsumptionCity.Text = item.FuelConsumptionCity.ToString();

textBoxFuelConsumptionGeneral.Text = item.FuelConsumptionCity.ToString();

textBoxFuelConsumptionHighway.Text = item.FuelConsumptionHighway.ToString();

textBoxPower.Text = item.EnginePower.ToString();

comboBoxOctan.Items.Clear();

comboBoxOctan.IsEnabled = true;

textBoxFuelConsumptionCity.Visibility = Visibility.Visible;

textBoxFuelConsumptionHighway.Visibility = Visibility.Visible;

if (item.Fuel == "Электричество")

{

Type = true;

comboBoxOctan.IsEnabled = false;

textBoxFuelConsumptionCity.Visibility = Visibility.Hidden;

textBoxFuelConsumptionHighway.Visibility = Visibility.Hidden;

}

else if (item.Fuel == "Дизельное топливо")

{

Type = false;

comboBoxOctan.IsEnabled = true;

List<Fuel> filteredList = AllFuel.Where(fuel => fuel.name == "Дизельное топливо").ToList();

foreach (var fuel in filteredList)

{

comboBoxOctan.Items.Add(fuel.octaneNumber);

}

}

else if (item.Fuel == "Бензин")

{

Type = false;

comboBoxOctan.IsEnabled = true;

List<Fuel> filteredList = AllFuel.Where(fuel => fuel.name == "Бензин").ToList();

foreach (var fuel in filteredList)

{

comboBoxOctan.Items.Add(fuel.octaneNumber);

}

}

}

}

}

}

private void UpdateCarButton\_Click(object sender, RoutedEventArgs e)//обновленение характеритик автомобиля

{

if (IsValidComboBox(comboBoxOctan) & СonsumptionСheck())

{

string car = comboBoxCar.SelectedItem.ToString();

string fuelConsumptionGeneral = FixStr(textBoxFuelConsumptionGeneral.Text);

string fuelConsumptionCity = FixStr(textBoxFuelConsumptionCity.Text);

string fuelConsumptionHighway = FixStr(textBoxFuelConsumptionHighway.Text);

string enginePower = FixStr(textBoxPower.Text);

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.Load("Car.xml");

XmlNode carNode = xmlDoc.SelectSingleNode($"//car[name='{car}']");

if (Type)

{

carNode.SelectSingleNode("fuelConsumptionGeneral").InnerText = fuelConsumptionGeneral;

carNode.SelectSingleNode("fuelConsumptionCity").InnerText = fuelConsumptionGeneral;

carNode.SelectSingleNode("fuelConsumptionHighway").InnerText = fuelConsumptionGeneral;

}

else

{

carNode.SelectSingleNode("fuelConsumptionGeneral").InnerText = fuelConsumptionGeneral;

carNode.SelectSingleNode("fuelConsumptionCity").InnerText = fuelConsumptionCity;

carNode.SelectSingleNode("fuelConsumptionHighway").InnerText = fuelConsumptionHighway;

}

carNode.SelectSingleNode("enginePower").InnerText = enginePower;

// Сохранение изменений в XML-документе

xmlDoc.Save("Car.xml");

MessageBox.Show($"Характеристики автомобиля {comboBoxCar.SelectedItem.ToString()} обновлены.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Ошибка ввода данных. Повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private bool СonsumptionСheck()//проверка значений

{

double ConsumptionGeneral, ConsumptionCity, ConsumptionHighway, power;

bool isNumericOne = IsValidDoubleInput(textBoxFuelConsumptionGeneral, 0, 100, out ConsumptionGeneral);

bool isNumericTwo = IsValidDoubleInput(textBoxFuelConsumptionCity, 0, 100, out ConsumptionCity);

bool isNumericThree = IsValidDoubleInput(textBoxFuelConsumptionHighway, 0, 100, out ConsumptionHighway);

bool isNumericFour = IsValidDoubleInput(textBoxFuelConsumptionHighway, 0, 2000, out power);

if (isNumericOne & isNumericTwo & isNumericThree & isNumericFour)

{

return true;

}

else

{

return false;

}

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)//проверка вещественных значений

{

value = 0;

if (box.Text == "")

{

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private bool IsValidComboBox(ComboBox box)//проверка корретконсти знначений в comboBox

{

if (box.SelectedIndex == -1)

{

if (!Type)

{

return false;

}

return true;

}

else

{

return true;

}

}

private string FixStr(string input)//замена . на ,

{

return input.Replace('.', ',');

}

private void DeleteCarButton\_Click(object sender, RoutedEventArgs e)// удаление автомобиля

{

if (comboBoxCar.SelectedIndex != -1)

{

XDocument xdoc = XDocument.Load("Car.xml");

// Определите условия фильтрации

string car = comboBoxCar.SelectedItem.ToString();

MessageBoxResult result = MessageBox.Show($"Автомобиль {car} будет удален из системы. Желаете продолжить?", "Подтверждение", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (result == MessageBoxResult.OK)

{

IEnumerable<XElement> fuelElements = xdoc.Descendants("car")

.Where(e => e.Element("name")?.Value == car);

// Удаляем найденные элементы

foreach (XElement fuelElement in fuelElements.ToList())

{

fuelElement.Remove();

}

// Сохраняем изменения обратно в XML-файл

xdoc.Save("Car.xml");

MessageBox.Show($"Автомобиль {car} был успешно удален из системы.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else if (result == MessageBoxResult.Cancel)

{

// Код для отмены выполнения

MessageBox.Show("Удаление отменено.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

MessageBox.Show("Автомобиль не выбран. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

<Window x:Class="JourneyExpense.EditWindows.EditRouteForm"

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:JourneyExpense.EditWindows"

mc:Ignorable="d"

Title="Редактирование маршрута"

Height="260" MinHeight="260" MaxHeight="280"

Width="400" MinWidth="400" MaxWidth="280"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="13\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="51\*"/>

<RowDefinition Height="36\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="Точка отправления" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="1"/>

<Label Content="Точка назначения" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="2"/>

<TextBox x:Name="textBoxDistance" FontSize="15" Margin="10,0,0,0" Height="27" VerticalAlignment="Center" Grid.Row="3" Grid.Column="1" Width="145" HorizontalAlignment="Left"/>

<Label Content="Расстояние" Margin="24,0,24,0" Height="29" VerticalAlignment="Center" FontSize="15" Grid.Row="3"/>

<Button x:Name="EditRouteButton" Content="Редактировать маршрут" Margin="10,0,10,0" Grid.Row="4" Height="28" VerticalAlignment="Center" Grid.Column="1" Click="EditRouteButton\_Click" />

<Button x:Name="DeleteRouteButton" Content="Удалить маршрут" HorizontalAlignment="Center" Grid.Row="5" VerticalAlignment="Top" Height="21" Width="110" Grid.Column="1" Margin="0,5,0,0" Click="DeleteRouteButton\_Click"/>

<ComboBox x:Name="comboBoxPointOne" Grid.Column="1" HorizontalAlignment="Center" Grid.Row="1" VerticalAlignment="Center" Width="180" Height="28" SelectionChanged="comboBoxPointOne\_SelectionChanged" MouseDoubleClick="comboBoxPointOne\_MouseDoubleClick"/>

<ComboBox x:Name="comboBoxPointTwo" Grid.Column="1" HorizontalAlignment="Center" Grid.Row="2" VerticalAlignment="Center" Width="180" Height="28" SelectionChanged="comboBoxPointTwo\_SelectionChanged" MouseDoubleClick="comboBoxPointTwo\_MouseDoubleClick"/>

<Label Grid.Column="1" Content="Км" HorizontalAlignment="Left" Margin="160,0,0,0" Grid.Row="3" VerticalAlignment="Center"/>

</Grid>

</Window>

using JourneyExpense.Classes;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.EditWindows

{

/// <summary>

/// Логика взаимодействия для EditRouteForm.xaml

/// </summary>

public partial class EditRouteForm : Window

{

private List<Route> AllRoutes = new List<Route>();

private List<string> PointA = new List<string>();

private List<string> PointB = new List<string>();

private bool isMessageBoxShowPoint = false;

public EditRouteForm()

{

InitializeComponent();

ReadData();

InitComboBox();

}

private void ReadData()//инициализация списка

{

AllRoutes.Clear();

AllRoutes = Route.ReadRousteInXML();

foreach (var item in AllRoutes)

{

if (!PointA.Contains(item.PointA))

{

PointA.Add(item.PointA);

}

if (!PointA.Contains(item.PointB))

{

PointA.Add(item.PointB);

}

if (!PointB.Contains(item.PointA))

{

PointB.Add(item.PointA);

}

if (!PointB.Contains(item.PointB))

{

PointB.Add(item.PointB);

}

}

PointA = PointA.OrderBy(item => item).ToList();

PointB = PointB.OrderBy(item => item).ToList();

}

private void InitComboBox()//иницилия combBox

{

comboBoxPointOne.Items.Clear();

comboBoxPointTwo.Items.Clear();

foreach (var item in PointA)

{

comboBoxPointOne.Items.Add(item);

}

foreach (var item in PointB)

{

comboBoxPointTwo.Items.Add(item);

}

}

private void EditRouteButton\_Click(object sender, RoutedEventArgs e)//редактировать данные в маршрутах

{

double distance = 0;

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1 && IsValidDoubleInput(textBoxDistance, 0, 10000, out distance))

{

string pointA = comboBoxPointOne.SelectedItem.ToString();

string pointB = comboBoxPointTwo.SelectedItem.ToString();

string distanceStr = FixStr(distance.ToString());

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.Load("Routes.xml");

XmlNode carNode = xmlDoc.SelectSingleNode($"//route[(PointA='{pointA}' and PointB='{pointB}') or (PointA='{pointB}' and PointB='{pointA}')]");

carNode.SelectSingleNode("kilometer").InnerText = distanceStr;

xmlDoc.Save("Routes.xml");

MessageBox.Show($"Дистанция маршрута {pointA} - {pointB} обновлена.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Ошибка ввода данных. Повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void DeleteRouteButton\_Click(object sender, RoutedEventArgs e)//удалить маршрут

{

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1)

{

XDocument xdoc = XDocument.Load("Routes.xml");

// Определите условия фильтрации

string pointA = comboBoxPointOne.SelectedItem.ToString();

string pointB = comboBoxPointTwo.SelectedItem.ToString();

MessageBoxResult result = MessageBox.Show($"Маршрут {pointA} - {pointB} будет удален из системы. Желаете продолжить?", "Подтверждение", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (result == MessageBoxResult.OK)

{

IEnumerable<XElement> routesElements = xdoc.Descendants("route")

.Where(e => (e.Element("PointA")?.Value == pointA && e.Element("PointB")?.Value == pointB) | (e.Element("PointA")?.Value == pointB && e.Element("PointB")?.Value == pointA));

// Удаляем найденные элементы

foreach (XElement routeElement in routesElements.ToList())

{

routeElement.Remove();

}

// Сохраняем изменения обратно в XML-файл

xdoc.Save("Routes.xml");

MessageBox.Show($"Маршрут {pointA} - {pointB} был успешно удален из системы.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else if (result == MessageBoxResult.Cancel)

{

// Код для отмены выполнения

MessageBox.Show("Удаление отменено.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

MessageBox.Show("Маршрут не выбран. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

ReadData();

InitComboBox();

textBoxDistance.Clear();

}

private void comboBoxPointOne\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1 && comboBoxPointOne.SelectedItem != comboBoxPointTwo.SelectedItem)

{

string A = comboBoxPointOne.SelectedItem.ToString();

string B = comboBoxPointTwo.SelectedItem.ToString();

isMessageBoxShowPoint = false;

if (A != B)

{

foreach (var item in AllRoutes)

{

if ((item.PointA == A && item.PointB == B) || (item.PointA == B && item.PointB == A))

{

this.textBoxDistance.Text = item.Distance.ToString();

isMessageBoxShowPoint = true;

break;

}

}

if (!isMessageBoxShowPoint)

{

MessageBox.Show("Выбранный маршрут не найден. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointOne.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

}

else if (comboBoxPointOne.SelectedIndex == -1 || comboBoxPointTwo.SelectedIndex == -1)

{

}

else if (comboBoxPointOne.SelectedIndex != -1 || comboBoxPointTwo.SelectedIndex != -1)

{

MessageBox.Show("Выберите корректное место назначения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointOne.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

private void comboBoxPointTwo\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

if (comboBoxPointOne.SelectedIndex != -1 && comboBoxPointTwo.SelectedIndex != -1 && comboBoxPointOne.SelectedItem != comboBoxPointTwo.SelectedItem)

{

string A = comboBoxPointOne.SelectedItem.ToString();

string B = comboBoxPointTwo.SelectedItem.ToString();

isMessageBoxShowPoint = false;

if (A != B)

{

foreach (var item in AllRoutes)

{

if ((item.PointA == A && item.PointB == B) || (item.PointA == B && item.PointB == A))

{

this.textBoxDistance.Text = item.Distance.ToString();

isMessageBoxShowPoint = true;

break;

}

}

if (!isMessageBoxShowPoint)

{

MessageBox.Show("Выбранный маршрут не найден. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointTwo.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

}

else if (comboBoxPointOne.SelectedIndex == -1 || comboBoxPointTwo.SelectedIndex == -1)

{

}

else if (comboBoxPointOne.SelectedIndex != -1 || comboBoxPointTwo.SelectedIndex != -1)

{

MessageBox.Show("Выберите корректное место назначения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

comboBoxPointTwo.SelectedIndex = -1;

textBoxDistance.Clear();

}

}

private void comboBoxPointOne\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка значений из coomboBox

{

comboBoxPointOne.SelectedIndex = -1;

}

private void comboBoxPointTwo\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)//очистка значений из coomboBox

{

comboBoxPointTwo.SelectedIndex = -1;

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)//проверка корректности вещественных значений

{

value = 0;

if (box.Text == "")

{

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

private string FixStr(string input)//замена . на ,

{

return input.Replace('.', ',');

}

}

}<Window x:Class="JourneyExpense.UpdatePriceForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Обновление цены на топливо"

Height="260" MinHeight="260" MaxHeight="260"

Width="400" MinWidth="400" MaxWidth="400"

Icon="/Icons/Icons.ico">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="13\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="48\*"/>

<RowDefinition Height="51\*"/>

<RowDefinition Height="35\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="Вид топилива" Margin="18,0,0,0" FontSize="15" Grid.Row="1" Height="29" VerticalAlignment="Center" HorizontalAlignment="Left" Width="164"/>

<ComboBox x:Name="comboBoxFuelType" Grid.Row="1" VerticalAlignment="Center" Height="29" Grid.Column="1" Margin="18,0,18,0" SelectionChanged="comboBoxFuelType\_SelectionChanged"/>

<Label Content="Тип топилива" Margin="18,0,0,0" FontSize="15" Grid.Row="2" Height="29" VerticalAlignment="Center" HorizontalAlignment="Left" Width="164"/>

<ComboBox x:Name="comboBoxFuelOctan" Grid.Row="2" VerticalAlignment="Center" Height="29" Grid.Column="1" Margin="18,0,18,0" SelectionChanged="comboBoxFuelOctan\_SelectionChanged"/>

<Label Content="Цена на литр топлива" Margin="0,10,0,0" FontSize="15" Grid.Row="3" Height="29" VerticalAlignment="Top" HorizontalAlignment="Center" Width="164"/>

<TextBox x:Name="textBoxPrice" Grid.Column="1" Grid.Row="3" TextWrapping="Wrap" VerticalAlignment="Top" Height="28" Margin="18,13,0,0" FontSize="20" Width="142" HorizontalAlignment="Left"/>

<Button x:Name="UpdatePriceButton" Content="Обновить цену" HorizontalAlignment="Center" Grid.Row="4" VerticalAlignment="Top" Height="28" Width="164" Click="UpdatePriceButton\_Click" Margin="0,10,0,0" Grid.Column="1"/>

<Button x:Name="DeleteFuelButton" Content="Удалить топливо" HorizontalAlignment="Center" Grid.Row="5" VerticalAlignment="Top" Height="21" Width="110" Grid.Column="1" Click="DeleteFuelButton\_Click" Margin="0,4,0,0"/>

<Label Grid.Column="1" Content="Руб" HorizontalAlignment="Left" Margin="161,14,0,0" Grid.Row="3" VerticalAlignment="Top"/>

</Grid>

</Window>

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using System.Xml.Linq;

using JourneyExpense.Classes;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для UpdatePriceForm.xaml

/// </summary>

public partial class UpdatePriceForm : Window

{

//

private List<string> FuelType = new List<string>() { "Бензин", "Дизельное топливо", "Электричество" };

private List<Fuel> AllFuel = Fuel.ReadFuelInXML();

public UpdatePriceForm()

{

InitializeComponent();

InitComboBox();

}

public void InitComboBox()

{

foreach (var item in FuelType)

{

comboBoxFuelType.Items.Add(item);

}

}

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

{

double value = 0;

if (comboBoxFuelOctan.SelectedIndex != -1 && comboBoxFuelType.SelectedIndex != -1 && IsValidDoubleInput(textBoxPrice, 0, 100, out value))

{

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.Load("Fuel.xml");

XmlNode fuelNode = xmlDoc.SelectSingleNode($"//fuel[name='{comboBoxFuelType.SelectedItem.ToString()}'][octaneNumber='{comboBoxFuelOctan.SelectedItem.ToString()}']");

XmlNode priceNode = fuelNode.SelectSingleNode("price");

priceNode.InnerText = FixStr(textBoxPrice.Text);

xmlDoc.Save("Fuel.xml");

MessageBox.Show($"Цена на {comboBoxFuelOctan.SelectedItem.ToString()} изменена на {textBoxPrice.Text}", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Ошибка ввода данных. Повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void comboBoxFuelType\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxFuelType.SelectedIndex != -1)

{

comboBoxFuelOctan.Items.Clear();

foreach (var item in AllFuel)

{

if (item.name == comboBoxFuelType.SelectedItem.ToString())

{

comboBoxFuelOctan.Items.Add(item.octaneNumber);

}

}

}

}

private void comboBoxFuelOctan\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxFuelOctan.SelectedIndex != -1)

{

foreach (var item in AllFuel)

{

if (comboBoxFuelOctan.SelectedItem.ToString() == item.octaneNumber)

{

textBoxPrice.Text = item.price.ToString();

}

}

}

}

private string FixStr(string input)

{

return input.Replace('.', ',');

}

private bool IsValidDoubleInput(TextBox box, int min, int max, out double value)

{

if (box.Text == "")

{

value = 0;

box.BorderBrush = Brushes.Red;

return false;

}

bool isNumeric = double.TryParse(FixStr(box.Text), out value);

if (isNumeric)

{

if (value > min && value < max)

{

box.BorderBrush = Brushes.Gray;

return true;

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

else

{

box.BorderBrush = Brushes.Red;

return false;

}

}

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

{

if (comboBoxFuelOctan.SelectedIndex != -1 && comboBoxFuelType.SelectedIndex != -1)

{

XDocument xdoc = XDocument.Load("Fuel.xml");

// Определите условия фильтрации

string fuelType = comboBoxFuelType.SelectedItem.ToString();

string octaneNumber = comboBoxFuelOctan.SelectedItem.ToString();

MessageBoxResult result = MessageBox.Show($"Топливо {fuelType} {octaneNumber} будет удалено из системы. Желаете продолжить?", "Подтверждение", MessageBoxButton.OKCancel, MessageBoxImage.Question);

if (result == MessageBoxResult.OK)

{

IEnumerable<XElement> fuelElements = xdoc.Descendants("fuel")

.Where(e => e.Element("name")?.Value == fuelType && e.Element("octaneNumber")?.Value == octaneNumber);

// Удаляем найденные элементы

foreach (XElement fuelElement in fuelElements.ToList())

{

fuelElement.Remove();

}

// Сохраняем изменения обратно в XML-файл

xdoc.Save("Fuel.xml");

MessageBox.Show($"Топливо {fuelType} {octaneNumber} было успешно удалено из системы.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else if (result == MessageBoxResult.Cancel)

{

// Код для отмены выполнения

MessageBox.Show("Удаление отменено.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

MessageBox.Show("Топливо не выбране. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

<Window x:Class="JourneyExpense.GraphForm"

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:JourneyExpense"

mc:Ignorable="d"

Title="Диаграмма расходов пользователя"

Height="480" MinHeight="480" MaxHeight="480"

Width="840" MinWidth="840" MaxWidth="840"

Icon="/Icons/Icons.ico">

<Grid HorizontalAlignment="Center" Width="820">

<Grid.RowDefinitions>

<RowDefinition Height="15\*"/>

<RowDefinition Height="192\*"/>

<RowDefinition Height="10\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="41\*"/>

<ColumnDefinition Width="119\*"/>

</Grid.ColumnDefinitions>

<Canvas x:Name="canvasGraph" Grid.Column="1" Margin="10,30,0,0" Height="380" VerticalAlignment="Top" Width="600" Grid.RowSpan="2"/>

<Label Content="Диаграмма расходов" Margin="30,10,0,0" Grid.Row="1" VerticalAlignment="Top" HorizontalAlignment="Left" Width="127"/>

<Label Content="С" HorizontalAlignment="Left" Margin="6,41,0,0" Grid.Row="1" VerticalAlignment="Top"/>

<DatePicker x:Name="DataPickerFirstData" HorizontalAlignment="Left" Margin="30,43,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150"/>

<Label Content="По" HorizontalAlignment="Left" Margin="6,88,0,0" Grid.Row="1" VerticalAlignment="Top"/>

<DatePicker x:Name="DataPickerSecondData" HorizontalAlignment="Left" Margin="30,90,0,0" Grid.Row="1" VerticalAlignment="Top" Width="150"/>

<Button x:Name="ButtonShow" Content="Показать расход за период" HorizontalAlignment="Left" Margin="10,134,0,0" Grid.Row="1" VerticalAlignment="Top" Height="25" Width="172" Click="ButtonShow\_Click"/>

<Label x:Name="TotalPrice" HorizontalAlignment="Left" Margin="10,224,0,0" Grid.Row="1" VerticalAlignment="Top"/>

<Button x:Name="ButtonCreateView" Content="Создать отчет" Margin="10,361,0,0" Grid.Row="1" Height="25" VerticalAlignment="Top" HorizontalAlignment="Left" Width="172" Click="ButtonCreateView\_Click\_1"/>

<Button x:Name="ButtonClear" Content="Очистить график" HorizontalAlignment="Left" Margin="10,193,0,0" Grid.Row="1" VerticalAlignment="Top" Width="172" Height="25" Click="ButtonClear\_Click"/>

<Button x:Name="ButtonShowAll" Content="Показать все поездки" HorizontalAlignment="Left" Margin="10,163,0,0" Grid.Row="1" VerticalAlignment="Top" Width="172" Height="25" Click="ButtonShowAll\_Click"/>

</Grid>

</Window>

using JourneyExpense.Classes;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Shapes;

using Xceed.Document.NET;

using Xceed.Words.NET;

namespace JourneyExpense

{

/// <summary>

/// Логика взаимодействия для GraphForm.xaml

/// </summary>

public partial class GraphForm : Window

{

private string UserName;

private string Surname;

private List<UsersRoutes> UserListRoutes = new List<UsersRoutes>();

private List<UsersRoutes> AllRoutes = UsersRoutes.ReadUsersRoutesInXML().OrderBy(r => DateTime.ParseExact(r.Date, "dd.MM.yyyy", CultureInfo.InvariantCulture)).ToList();

public GraphForm(string userName, string surname)

{

UserName = userName;

Surname = surname;

InitializeComponent();

ReadData();

CreateGraph();

DrawGraph();

}

private void ButtonShow\_Click(object sender, RoutedEventArgs e)//обработчик нажатия на клавишу для потсрания графика по датам

{

if (DataPickerFirstData.SelectedDate.HasValue && DataPickerSecondData.SelectedDate.HasValue)

{

DateTime dateOne = DataPickerFirstData.SelectedDate.Value;

DateTime dateTwo = DataPickerSecondData.SelectedDate.Value;

if (dateOne < dateTwo)

{

canvasGraph.Children.Clear();

CreateGraph();

string dateOneStr = dateOne.ToString("dd.MM.yyyy");

string dateTwoStr = dateTwo.ToString("dd.MM.yyyy");

DrawGraphInDate(dateOneStr, dateTwoStr);

}

else

{

MessageBox.Show("Введены некорректные даты", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Заполните все поля","Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public void ReadData() // инициализаиця списка все поездок

{

foreach (var route in AllRoutes)

{

if (route.User == UserName)

{

UserListRoutes.Add(route);

}

}

}

private void CreateGraph() // создание графика

{

DrawHorizontalLine();

DrawVerticalLine();

}

private void DrawVerticalLine() // создание вертикальный леиний в графике

{

int left = 0;

int step = 10;

while (left <= 600)

{

Line line = new Line

{

X1 = 0,

Y1 = 0,

X2 = 0,

Y2 = 380,

Stroke = Brushes.Gray,

StrokeThickness = 0.5

};

canvasGraph.Children.Add(line);

Canvas.SetLeft(line, left);

left += step;

}

}

private void DrawHorizontalLine()// создание горизонтальных леиний в графике

{

int top = 380;

int step = 10;

while (top >= 0)

{

Line line = new Line

{

X1 = 0,

Y1 = 0,

X2 = 600,

Y2 = 0,

Stroke = Brushes.Gray,

StrokeThickness = 0.5

};

canvasGraph.Children.Add(line);

Canvas.SetTop(line, top);

top -= step;

}

}

private void DrawGraph() // создание общего графика за весь период

{

if (UserListRoutes.Count != 0)

{

Polyline polyline = new Polyline();

polyline.Stroke = Brushes.Blue;

polyline.StrokeThickness = 2;

int count = UserListRoutes.Count;

double price = UserListRoutes.Max(item => item.Price);

double stepX = 600 / count;//поездки

double stepY = 380 / price;//цена поездки

double totalPrice = 0;

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

{

totalPrice += UserListRoutes[i].Price;

double x = i \* stepX;

double y = 380 - (UserListRoutes[i].Price \* stepY);

polyline.Points.Add(new Point(x, y));

AddPoint(stepX, stepY, i, UserListRoutes);

AddTextBlock(UserListRoutes, i, x, y);

AddAxis(UserListRoutes[i].Date, 0 + (stepX \* i), 380);

}

double step = price / 30;

for (double i = 0; i <= price; i += step)

{

AddAxisY(i, -10, 370 - (i \* stepY));

}

if (polyline.Parent is Panel panel)

{

panel.Children.Remove(polyline);

}

canvasGraph.Children.Add(polyline);

TotalPrice.Content = "Итог за выбранный срок: \n" + Math.Round(totalPrice, 2).ToString() + " Рублей";

}

else

{

MessageBox.Show("Создайте первую поездку и повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void DrawGraphInDate(string dateOne, string dateTwo) // создания графика в диапазоне дат

{

DateTime fromDate = DateTime.ParseExact(dateOne, "dd.MM.yyyy", CultureInfo.InvariantCulture);

DateTime toDate = DateTime.ParseExact(dateTwo, "dd.MM.yyyy", CultureInfo.InvariantCulture);

List<UsersRoutes> filteredList = UserListRoutes

.Where(item => DateTime.ParseExact(item.Date, "dd.MM.yyyy", CultureInfo.InvariantCulture) >= fromDate &&

DateTime.ParseExact(item.Date, "dd.MM.yyyy", CultureInfo.InvariantCulture) <= toDate).ToList();

if (filteredList.Count != 0)

{

Polyline polyline = new Polyline();

polyline.Stroke = Brushes.Blue;

polyline.StrokeThickness = 2;

int count = filteredList.Count;

double price = filteredList.Max(item => item.Price);

double stepX = 600 / count;

double stepY = 380 / price;

double totalPrice = 0;

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

{

totalPrice += filteredList[i].Price;

double x = i \* stepX;

double y = 380 - (filteredList[i].Price \* stepY);

polyline.Points.Add(new Point(x, y));

AddPoint(stepX, stepY, i, filteredList);

AddTextBlock(filteredList, i, x, y);

AddAxis(filteredList[i].Date, 0 + (stepX \* i), 380);

}

double step = price / 30;

for (double i = 0; i <= price; i += step)

{

AddAxisY(i, -10, 370 - (i \* stepY));

}

TotalPrice.Content = "Итог за выбранный срок: \n" + Math.Round(totalPrice, 2).ToString() + " Рублей";

if (polyline.Parent is Panel panel)

{

panel.Children.Remove(polyline);

}

canvasGraph.Children.Add(polyline);

}

else

{

MessageBox.Show("Поезкди в данный период времени не обнаружены.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AddAxis(string i, double x, double y)

{

TextBlock textBlock = new TextBlock();

textBlock.FontSize = 10;

textBlock.Text = i;

textBlock.Margin = new Thickness(x, y, 0, 0);

canvasGraph.Children.Add(textBlock);

}

private void AddAxisY(double i, double x, double y)// добавить размернуб сетку y

{

TextBlock textBlock = new TextBlock();

textBlock.FontSize = 10;

textBlock.Text = Math.Round(i).ToString();

textBlock.Margin = new Thickness(x, y, 0, 0);

canvasGraph.Children.Add(textBlock);

}

private void AddPoint(double stepX, double stepY, int i, List<UsersRoutes> list) // создание вершины маршрута

{

Ellipse redPoint = new Ellipse();

redPoint.Width = 6;

redPoint.Height = 6;

redPoint.Fill = Brushes.DarkBlue;

redPoint.Margin = new Thickness(i \* stepX - 2, 380 - (list[i].Price \* stepY) - 2, 0, 0);

canvasGraph.Children.Add(redPoint);

}

private void AddTextBlock(List<UsersRoutes> list, int i, double x, double y) // создание значенрий у каждой точки

{

TextBlock textBlock = new TextBlock();

textBlock.Text = list[i].Price.ToString();

textBlock.Margin = new Thickness(x + 4, y - 16, 0, 0);

canvasGraph.Children.Add(textBlock);

}

private void ButtonClear\_Click(object sender, RoutedEventArgs e) // очистить значение

{

canvasGraph.Children.Clear();

TotalPrice.Content = "";

CreateGraph();

}

private void ButtonShowAll\_Click(object sender, RoutedEventArgs e) // показать поезкди за все даты

{

canvasGraph.Children.Clear();

CreateGraph();

DrawGraph();

}

private void CalculateFromData(List<UsersRoutes> list, out double dieselFuel, out double petrolFuel, out double electroFuel, out double averagePrice)

{

dieselFuel = 0;

petrolFuel = 0;

electroFuel = 0;

averagePrice = 0;

int count = 0;

foreach (var item in list)

{

if (item.User == UserName)

{

count++;

if (item.FuelType == "Бензин")

{

petrolFuel += item.UsedFuel;

}

else if (item.FuelType == "Дизельное топливо")

{

dieselFuel += item.UsedFuel;

}

else

{

electroFuel += item.UsedFuel;

}

averagePrice += item.Price;

}

}

averagePrice = averagePrice / (double)count;

}

private void ButtonCreateView\_Click\_1(object sender, RoutedEventArgs e) // создание отчета

{

try

{

double dieselFuel = 0;

double petrolFuel = 0;

double electroFuel = 0;

double averagePrice = 0;

DateTime reportDate = DateTime.Now;

// Формирование имени файла и выбор места сохранения

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Документ Word (\*.docx)|\*.docx";

saveFileDialog.FileName = $"Отчет\_{reportDate:dd.MM.yyyy}.docx";

if (saveFileDialog.ShowDialog() == true)

{

string fileName = saveFileDialog.FileName;

int count = 0;

// Создание документа Word

using (DocX document = DocX.Create(fileName))

{

string reportTitle;

if (DataPickerFirstData.SelectedDate.HasValue && DataPickerSecondData.SelectedDate.HasValue)

{

DateTime firstDate = DataPickerFirstData.SelectedDate.Value;

DateTime secondDate = DataPickerSecondData.SelectedDate.Value;

//сортировка поездок по периоду

List<UsersRoutes> filteredList = UserListRoutes

.Where(item => DateTime.ParseExact(item.Date, "dd.MM.yyyy", CultureInfo.InvariantCulture) >= firstDate &&

DateTime.ParseExact(item.Date, "dd.MM.yyyy", CultureInfo.InvariantCulture) <= secondDate).ToList();

count = filteredList.Count;

CalculateFromData(filteredList, out dieselFuel, out petrolFuel, out electroFuel, out averagePrice);

reportTitle = $"Отчет о поездках";

document.InsertParagraph(reportTitle).FontSize(14d).Bold().Alignment = Alignment.center;

reportTitle = $"в период с {firstDate:dd.MM.yy} по {secondDate:dd.MM.yyyy}";

reportTitle += Environment.NewLine;

}

else

{

count = UserListRoutes.Count;

CalculateFromData(UserListRoutes, out dieselFuel, out petrolFuel, out electroFuel, out averagePrice);

reportTitle = $"Отчет о поездках";

document.InsertParagraph(reportTitle).FontSize(14d).Bold().Alignment = Alignment.center;

reportTitle = $"в период по {reportDate:dd.MM.yyyy}";

reportTitle += Environment.NewLine;

}

document.InsertParagraph(reportTitle).FontSize(12d).Bold().Alignment = Alignment.center;

// Добавление информации о средней стоимости, жидком топливе и электричестве

document.InsertParagraph($"Пользователь {UserName} {Surname} совершил {count} поездок за период.").FontSize(12d);

document.InsertParagraph($"Средняя стоимость поездки: {averagePrice:0.00} рублей.").FontSize(12d);

document.InsertParagraph($"Общее кол-во потраченного бензина: {petrolFuel} литров.").FontSize(12d);

document.InsertParagraph($"Общее кол-во потраченного дизельного топлива: {dieselFuel} литров.").FontSize(12d);

document.InsertParagraph($"Общее кол-во потраченного электрического топлива: {electroFuel} Киловатт-часов.").FontSize(12d);

document.InsertParagraph($"Дата формирования отчета {reportDate:dd.MM.yyyy}").FontSize(12d).Alignment = Alignment.right;

// Сохранение документа

document.Save();

}

MessageBox.Show("Отчет успешно создан!", "Создание отчета", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch(System.IO.IOException ex)

{

MessageBox.Show("Отчет не создан. Закройте открытый отчет и повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}using System;

using System.Collections.Generic;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

[Serializable]

public class Car

{

public string Name { get; set; }

public int Year { get; set; }

public string TypeCar { get; set; }

public int MaxSpeed { get; set; }

public int SeatingCapacity { get; set; }

public string Fuel { get; set; }

public string FuelOctan { get; set; }

public double FuelConsumptionGeneral { get; set; }

public double FuelConsumptionCity { get; set; }

public double FuelConsumptionHighway { get; set; }

public double EnginePower { get; set; }

public double TankSize { get; set; }

public Car() { }

public Car(string name, int year, string typeCar, int maxSpeed, int seatingCapacity, string fuel, string fuelOctan, double fuelConsumptionGeneral, double fuelConsumptionCity, double fuelConsumptionHighway, double enginePower, double tankSize)

{

Name = name;

Year = year;

TypeCar = typeCar;

MaxSpeed = maxSpeed;

SeatingCapacity = seatingCapacity;

Fuel = fuel;

FuelOctan = fuelOctan;

FuelConsumptionGeneral = fuelConsumptionGeneral;

FuelConsumptionCity = fuelConsumptionCity;

FuelConsumptionHighway = fuelConsumptionHighway;

EnginePower = enginePower;

TankSize = tankSize;

}

public static List<Car> ReadCarInXML()//чтение автомобилей из XML

{

List<Car> Cars = new List<Car>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Car.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

Car car = new Car();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "name")

{

car.Name = childnode.InnerText;

}

if (childnode.Name == "year")

{

car.Year = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "typeCar")

{

car.TypeCar = childnode.InnerText;

}

if (childnode.Name == "maxSpeed")

{

car.MaxSpeed = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "seatingCapacity")

{

car.SeatingCapacity = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "fuel")

{

car.Fuel = childnode.InnerText;

}

if (childnode.Name == "fuelOctan")

{

car.FuelOctan = childnode.InnerText;

}

if (childnode.Name == "fuelConsumptionGeneral")

{

car.FuelConsumptionGeneral = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionCity")

{

car.FuelConsumptionCity = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionHighway")

{

car.FuelConsumptionHighway = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "enginePower")

{

car.EnginePower = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "tankSize")

{

car.TankSize = Convert.ToDouble(childnode.InnerText);

}

}

Cars.Add(car);

}

return Cars;

}

public bool AddCarInXML()//добавление атвомомбилей в XML

{

bool permission = true;

foreach (var item in ReadCarInXML())

{

if (item.Name.ToLower() == Name.ToLower() && item.Year == Year)

{

permission = false;

}

}

if (permission)

{

XElement root = XElement.Load("Car.xml");

XElement carElement = new XElement("car",

new XElement("name", Name.TrimEnd()),

new XElement("year", Year),

new XElement("typeCar", TypeCar),

new XElement("maxSpeed", MaxSpeed),

new XElement("seatingCapacity", SeatingCapacity),

new XElement("fuel", Fuel),

new XElement("fuelOctan", FuelOctan),

new XElement("fuelConsumptionGeneral", FixStr(FuelConsumptionGeneral)),

new XElement("fuelConsumptionCity", FixStr(FuelConsumptionCity)),

new XElement("fuelConsumptionHighway", FixStr(FuelConsumptionHighway)),

new XElement("enginePower", FixStr(EnginePower)),

new XElement("tankSize", FixStr(TankSize))

);

root.Add(carElement);

root.Save("Car.xml");

return true;

}

else

{

return false;

}

}

public string FixStr(double x)

{

return x.ToString().Replace('.', ',');

}

}

}

using System;

using System.Collections.Generic;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

[Serializable]

public class Car

{

public string Name { get; set; }

public int Year { get; set; }

public string TypeCar { get; set; }

public int MaxSpeed { get; set; }

public int SeatingCapacity { get; set; }

public string Fuel { get; set; }

public string FuelOctan { get; set; }

public double FuelConsumptionGeneral { get; set; }

public double FuelConsumptionCity { get; set; }

public double FuelConsumptionHighway { get; set; }

public double EnginePower { get; set; }

public double TankSize { get; set; }

public Car() { }

public Car(string name, int year, string typeCar, int maxSpeed, int seatingCapacity, string fuel, string fuelOctan, double fuelConsumptionGeneral, double fuelConsumptionCity, double fuelConsumptionHighway, double enginePower, double tankSize)

{

Name = name;

Year = year;

TypeCar = typeCar;

MaxSpeed = maxSpeed;

SeatingCapacity = seatingCapacity;

Fuel = fuel;

FuelOctan = fuelOctan;

FuelConsumptionGeneral = fuelConsumptionGeneral;

FuelConsumptionCity = fuelConsumptionCity;

FuelConsumptionHighway = fuelConsumptionHighway;

EnginePower = enginePower;

TankSize = tankSize;

}

public static List<Car> ReadCarInXML()//чтение автомобилей из XML

{

List<Car> Cars = new List<Car>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Car.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

Car car = new Car();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "name")

{

car.Name = childnode.InnerText;

}

if (childnode.Name == "year")

{

car.Year = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "typeCar")

{

car.TypeCar = childnode.InnerText;

}

if (childnode.Name == "maxSpeed")

{

car.MaxSpeed = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "seatingCapacity")

{

car.SeatingCapacity = Convert.ToInt32(childnode.InnerText);

}

if (childnode.Name == "fuel")

{

car.Fuel = childnode.InnerText;

}

if (childnode.Name == "fuelOctan")

{

car.FuelOctan = childnode.InnerText;

}

if (childnode.Name == "fuelConsumptionGeneral")

{

car.FuelConsumptionGeneral = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionCity")

{

car.FuelConsumptionCity = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "fuelConsumptionHighway")

{

car.FuelConsumptionHighway = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "enginePower")

{

car.EnginePower = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "tankSize")

{

car.TankSize = Convert.ToDouble(childnode.InnerText);

}

}

Cars.Add(car);

}

return Cars;

}

public bool AddCarInXML()//добавление атвомомбилей в XML

{

bool permission = true;

foreach (var item in ReadCarInXML())

{

if (item.Name.ToLower() == Name.ToLower() && item.Year == Year)

{

permission = false;

}

}

if (permission)

{

XElement root = XElement.Load("Car.xml");

XElement carElement = new XElement("car",

new XElement("name", Name.TrimEnd()),

new XElement("year", Year),

new XElement("typeCar", TypeCar),

new XElement("maxSpeed", MaxSpeed),

new XElement("seatingCapacity", SeatingCapacity),

new XElement("fuel", Fuel),

new XElement("fuelOctan", FuelOctan),

new XElement("fuelConsumptionGeneral", FixStr(FuelConsumptionGeneral)),

new XElement("fuelConsumptionCity", FixStr(FuelConsumptionCity)),

new XElement("fuelConsumptionHighway", FixStr(FuelConsumptionHighway)),

new XElement("enginePower", FixStr(EnginePower)),

new XElement("tankSize", FixStr(TankSize))

);

root.Add(carElement);

root.Save("Car.xml");

return true;

}

else

{

return false;

}

}

public string FixStr(double x)

{

return x.ToString().Replace('.', ',');

}

}

}

using System;

using System.Collections.Generic;

using System.Windows.Documents;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

public class Fuel

{

public string name { get; set; }

public string octaneNumber { get; set; }

public double price { get; set; }

public Fuel() { }

public Fuel(string name, string octaneNumber, double price)

{

this.name = name;

this.octaneNumber = octaneNumber;

this.price = price;

}

public static List<Fuel> ReadFuelInXML()//читать топливо из XML

{

List<Fuel> Fuel = new List<Fuel>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Fuel.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

Fuel fuel = new Fuel();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "name")

{

fuel.name = childnode.InnerText;

}

if (childnode.Name == "octaneNumber")

{

fuel.octaneNumber = childnode.InnerText;

}

if (childnode.Name == "price")

{

fuel.price = Convert.ToDouble(childnode.InnerText);

}

}

Fuel.Add(fuel);

}

return Fuel;

}

public bool AddFuelInXML()//добавить топливо в XML

{

bool permission = true;

foreach (var item in ReadFuelInXML())

{

if (item.octaneNumber == octaneNumber)

{

permission = false;

}

}

if (permission)

{

XElement root = XElement.Load("Fuel.xml");

XElement routesElement = new XElement("fuel",

new XElement("name", name.TrimEnd()),

new XElement("octaneNumber", octaneNumber),

new XElement("price", FixStr(price))

);

root.Add(routesElement);

root.Save("Fuel.xml");

return true;

}

else

{

return false;

}

}

public string FixStr(double x)

{

return x.ToString().Replace('.', ',');

}

}

}

using System;

using System.Collections.Generic;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

public class Route

{

public string PointA { get; set; }

public string PointB { get; set; }

public double Distance { get; set; }

public Route() { }

public Route(string pointA, string pointB, double distance)

{

PointA = pointA;

PointB = pointB;

Distance = distance;

}

public static List<Route> ReadRousteInXML()//прочитать данные о поездках из XML

{

List<Route> Route = new List<Route>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("Routes.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

Route route = new Route();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "PointA")

{

route.PointA = childnode.InnerText;

}

if (childnode.Name == "PointB")

{

route.PointB = childnode.InnerText;

}

if (childnode.Name == "kilometer")

{

route.Distance = Convert.ToDouble(childnode.InnerText);

}

}

Route.Add(route);

}

return Route;

}

public bool AddRoutesInXML()//добавить маршрут в XML

{

bool permission = true;

foreach (var item in ReadRousteInXML())

{

if (item.PointA == PointA && item.PointB == PointB || item.PointA == PointB && item.PointB == PointA)

{

permission = false;

}

}

if (permission)

{

XElement root = XElement.Load("Routes.xml");

XElement routesElement = new XElement("route",

new XElement("PointA", PointA),

new XElement("PointB", PointB),

new XElement("kilometer", FixStr(Distance))

);

root.Add(routesElement);

root.Save("Routes.xml");

return true;

}

else

{

return false;

}

}

public string FixStr(double x)

{

return x.ToString().Replace('.', ',');

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

public class User

{

public string login { get; set; }

public string password { get; set; }

public string name { get; set; }

public string surname { get; set; }

public User() { }

public User(string login, string password, string name, string surname)

{

this.login = login;

this.password = password;

this.name = name;

this.surname = surname;

}

public bool AddUserInXML()

{

try

{

// Читаем список пользователей из XML файла

List<User> list = ReadUserInXML();

// Проверяем, если уже существует пользователь с таким же логином

foreach (var item in list)

{

if (login == item.login)

{

MessageBox.Show("Данный логин уже занят. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

}

// Загружаем корневой элемент из XML файла

XElement root = XElement.Load("User.xml");

// Создаем новый элемент "user" с данными о новом пользователе

XElement userElement = new XElement("user",

new XElement("login", login),

new XElement("password", password),

new XElement("name", name),

new XElement("surname", surname)

);

// Добавляем новый элемент "user" в корневой элемент

root.Add(userElement);

// Сохраняем изменения в XML файле

root.Save("User.xml");

return true;

}

catch (FileNotFoundException ex)

{

// Если файл User.xml не найден, создаем новый XML файл и сохраняем в него данные о пользователе

XDocument xdoc = new XDocument();

XElement user = new XElement("user",

new XElement("login", login),

new XElement("password", password),

new XElement("name", name),

new XElement("surname", surname)

);

XElement users = new XElement("users", user);

xdoc.Add(users);

xdoc.Save("User.xml");

return true;

}

catch (Exception ex)

{

// В случае возникновения других исключений, возвращаем false

return false;

}

}

public static List<User> ReadUserInXML()

{

try

{

List<User> User = new List<User>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("User.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

User user = new User();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "login")

{

user.login = childnode.InnerText;

}

if (childnode.Name == "password")

{

user.password = childnode.InnerText;

}

if (childnode.Name == "name")

{

user.name = childnode.InnerText;

}

if (childnode.Name == "surname")

{

user.surname = childnode.InnerText;

}

}

User.Add(user);

}

return User;

}

catch (Exception ex)

{

return new List<User>();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Runtime.ConstrainedExecution;

using System.Transactions;

using System.Windows.Markup;

using System.Xml;

using System.Xml.Linq;

namespace JourneyExpense.Classes

{

public class UsersRoutes

{

public string User { get; set; }

public string Car { get; set; }

public string PointA { get; set; }

public string PointB { get; set; }

public double Distance { get; set; }

public double Price { get; set; }

public string Date { get; set; }

public string FuelType { get; set; }

public double UsedFuel { get; set; }

public double AverageSpeed { get; set; }

public UsersRoutes() { }

public UsersRoutes(string user, string car, string pointA, string pointB, double distance, double price, string fuelType, string date, double usedfuel, double averageSpeed)

{

User = user;

Car = car;

PointA = pointA;

PointB = pointB;

Distance = distance;

Price = price;

FuelType = fuelType;

Date = date;

UsedFuel = usedfuel;

AverageSpeed = averageSpeed;

}

public bool AddRoutesInXML()//добавить поездку в XML

{

XElement root = XElement.Load("UsersRoutes.xml");

XElement carElement = new XElement("route",

new XElement("User", User),

new XElement("Car", Car),

new XElement("PointA", PointA),

new XElement("PointB", PointB),

new XElement("Distance", FixStr(Distance)),

new XElement("Price", FixStr(Price)),

new XElement("FuelType", FuelType),

new XElement("Date", Date),

new XElement("UsedFuel", FixStr(UsedFuel)),

new XElement("AverageSpeed", FixStr(AverageSpeed))

);

root.Add(carElement);

root.Save("UsersRoutes.xml");

return true;

}

public static List<UsersRoutes> ReadUsersRoutesInXML()//добавить поездку в XML

{

List<UsersRoutes> Route = new List<UsersRoutes>();

XmlDocument xDoc = new XmlDocument();

xDoc.Load("UsersRoutes.xml");

XmlElement xRoot = xDoc.DocumentElement;

foreach (XmlNode xnode in xRoot)

{

UsersRoutes route = new UsersRoutes();

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "User")

{

route.User = childnode.InnerText;

}

if (childnode.Name == "Car")

{

route.Car = childnode.InnerText;

}

if (childnode.Name == "PointA")

{

route.PointA = childnode.InnerText;

}

if (childnode.Name == "PointB")

{

route.PointB = childnode.InnerText;

}

if (childnode.Name == "Distance")

{

route.Distance = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "Price")

{

route.Price = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "FuelType")

{

route.FuelType = childnode.InnerText;

}

if (childnode.Name == "Date")

{

route.Date = childnode.InnerText;

}

if (childnode.Name == "UsedFuel")

{

route.UsedFuel = Convert.ToDouble(childnode.InnerText);

}

if (childnode.Name == "AverageSpeed")

{

route.AverageSpeed = Convert.ToDouble(childnode.InnerText);

}

}

Route.Add(route);

}

return Route;

}

public string FixStr(double x)

{

return x.ToString().Replace('.', ',');

}

}

}

# **ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Результат работы программы**

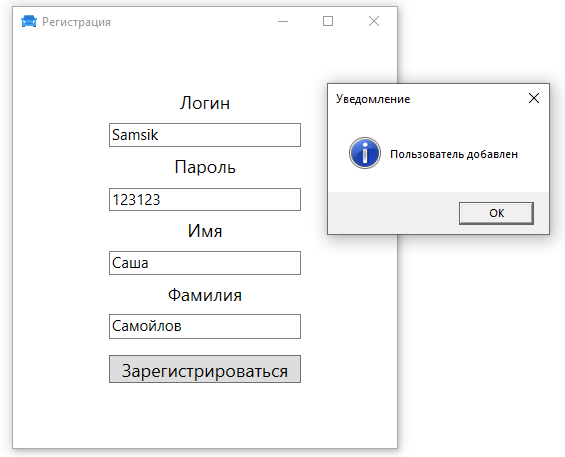


Рисунок Б.1 – Результат действия «Регистрация пользователя»

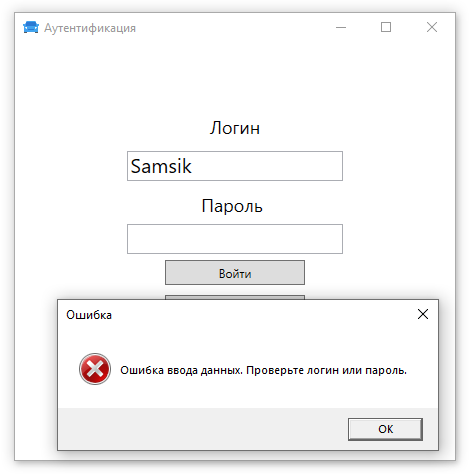


Рисунок Б.2 – Результат действия «Авторизация в приложении»

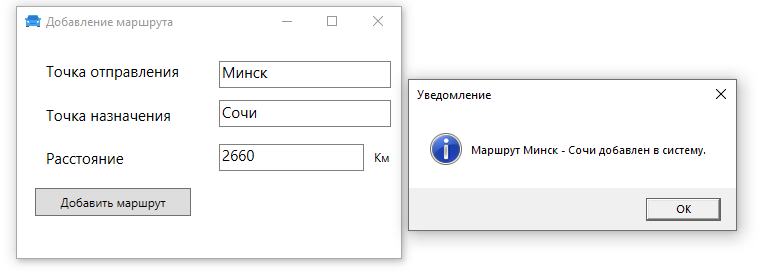


Рисунок Б.3 – Результат действия «Добавление маршрута»

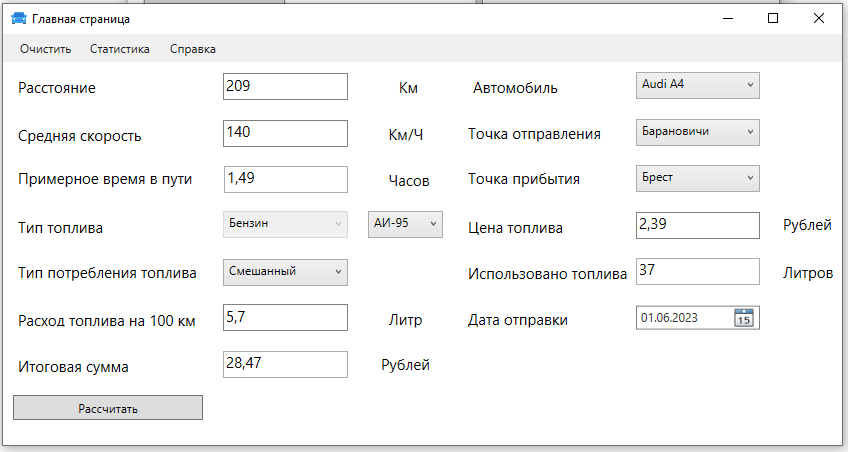


Рисунок Б.4 – Результат действия «Расчет стоимости поездки»

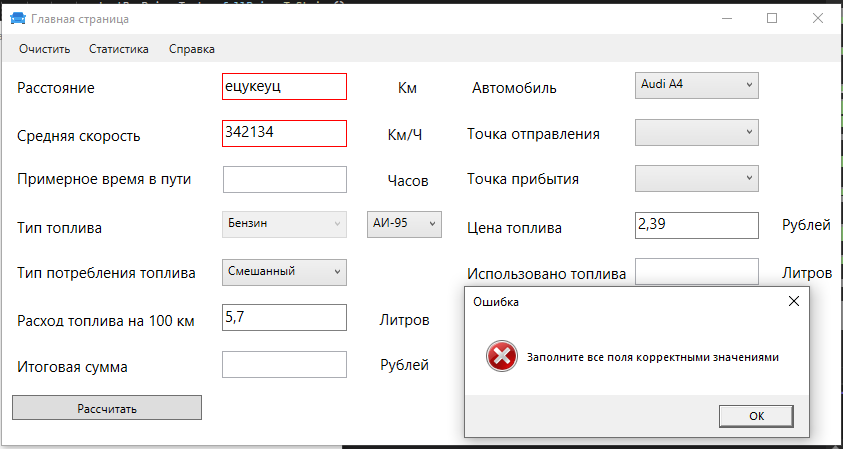


Рисунок Б.5 – Результат действия «Расчет стоимости поездки»

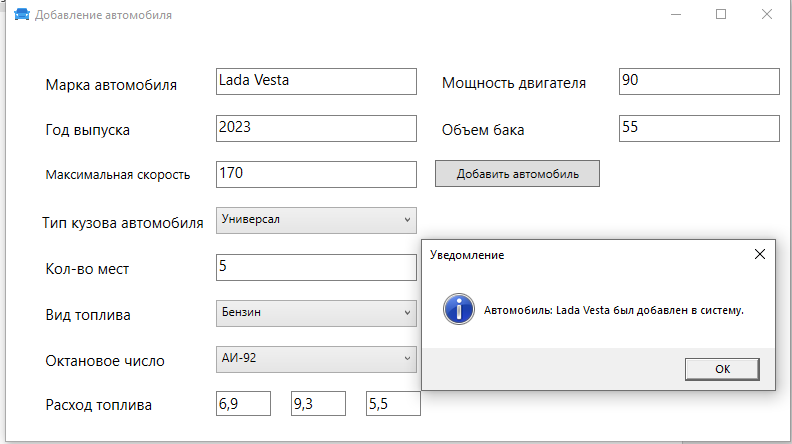
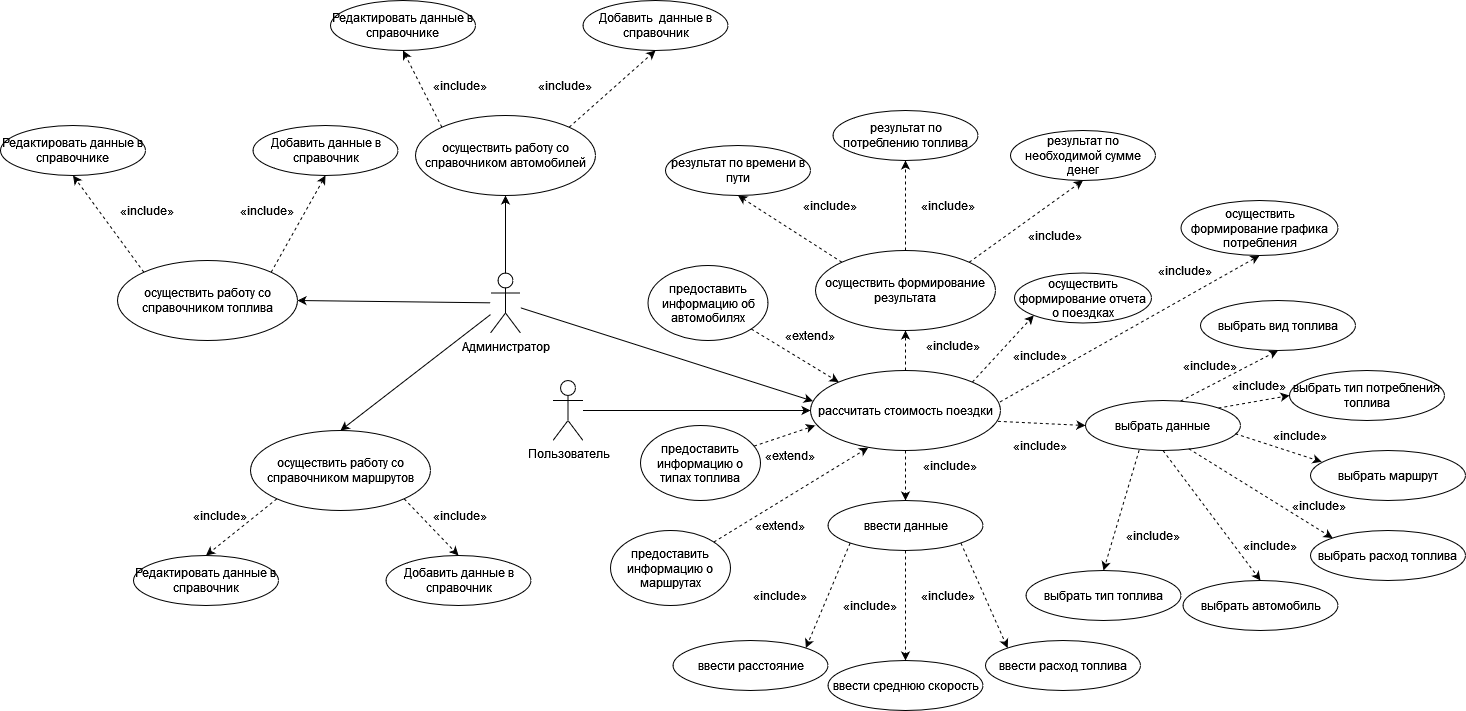


Рисунок Б.6 – Результат действия «Добавление автомобиля»



Провер.

Коропа Е.Н.

Лист 1

№ докум.

Подпись

Дата

Изм.

Лист

Разраб.

Самойлов А.С.

Подп. и дата

Инв.№ дубл.

Инв.№подл.

*КП Т.0910018.401 ГЧ*

*КП Т.0910018.401 ГЧ*

Утверд.

Т. Контр.

Листов 3

*Разработка программного средства для расчета стоимости поездка на автомобиле*

*Диаграмма вариантов использования*

У

Масса

Лиcт.

Масштаб

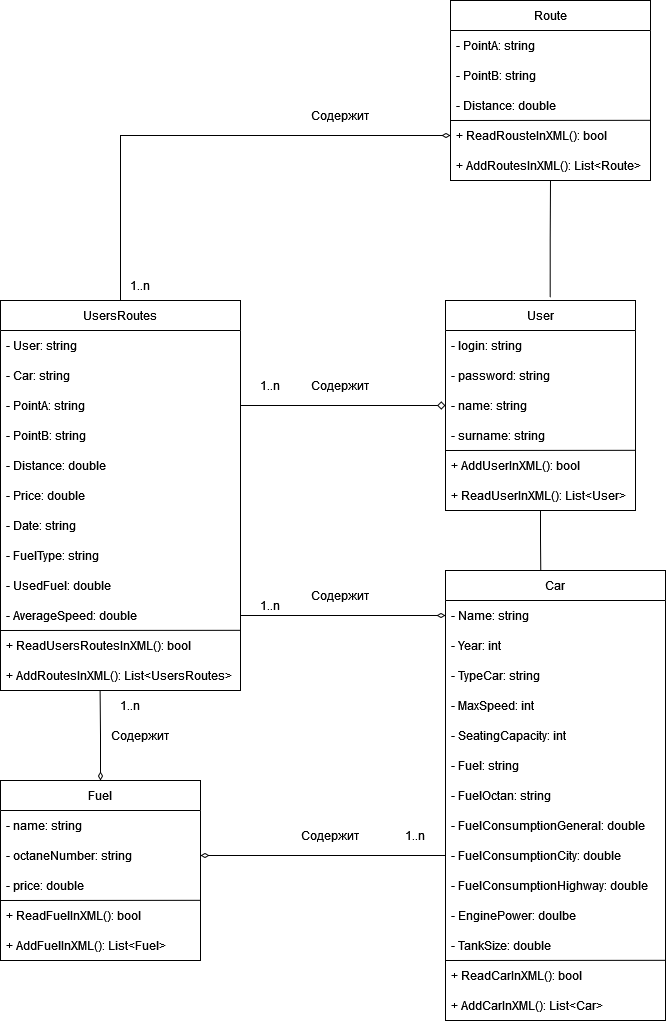
Реценз.

КБП

Н. Контр.

Взам.инв.№

Подп. и дата



Лист 2

№ докум.

Подпись

Дата

Изм.

Лист

Разраб.

Самойлов А.С.

Подп. и дата

Инв.№ дубл.

Инв.№подл.

Провер.

Коропа Е.Н.

*КП Т.0910018.401 ГЧ*

*КП Т.0910018.401 ГЧ*

Утверд.

Т. Контр.

Листов 3

*Разработка программного средства для расчета стоимости поездка на автомобиле*

*Диаграмма классов*

У

Масса

Лиcт.

Масштаб

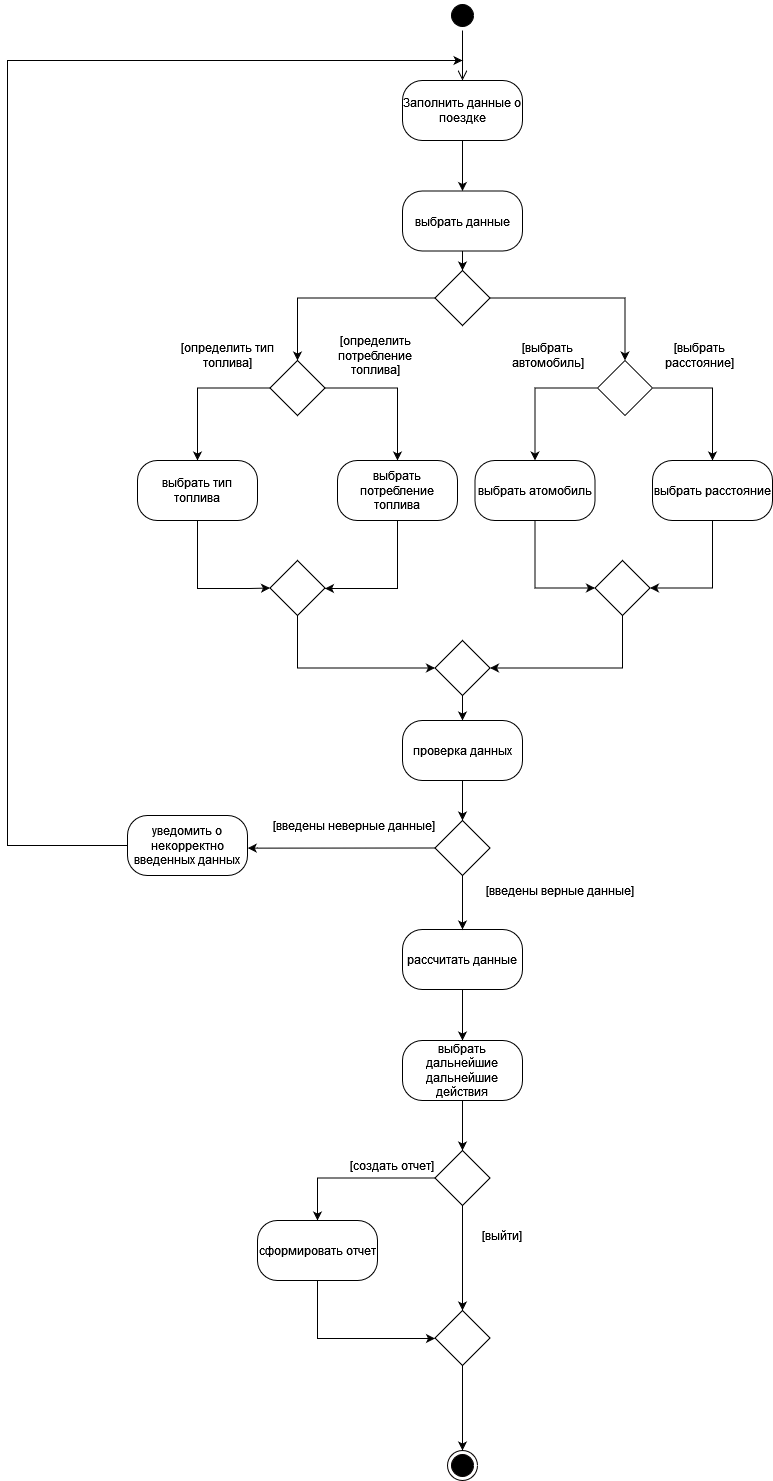
Реценз.

КБП

Н. Контр.

Взам.инв.№

Подп. и дата



Лист 3

Разраб.

Самойлов А.С.

№ докум.

Подпись

Дата

Изм.

Лист

Подп. и дата

Инв.№ дубл.

Инв.№подл.

Провер.

Коропа Е.Н.

*КП Т.0910018.401 ГЧ*

*КП Т.0910018.401 ГЧ*

Утверд.

Т. Контр.

Листов 3

*Разработка программного средства для расчета стоимости поездка на автомобиле*

*Диаграмма деятельности*

У

Масса

Лиcт.

Масштаб

Реценз.

КБП

Н. Контр.

Взам.инв.№

Подп. и дата

**Этикетка**

для курсового проекта

**Курсовой проект**

Тема « Разработка программного средства для расчета стоимости поездки на автомобиле»

КП Т.091018.401

Разработан

Утвержден

Разработчик Самойлов А.С.

Руководитель: Коропа E.H.

Технические средства: Процессор 1.3 ГГц, ОЗУ 6 ГБ, 500 ГБ свободного места

Программные средства: MS Visual Studio 2022 Community, MS Word, Dr.Explain

**Состав документа:**

Пояснительная записка – ПЗ\_СамойловТ091.docх

Графическая часть – Диаграмма вариантов использования.docx, Диаграмма классов.docx, Диаграмма деятельности.docx.

Папка с проектом – JourneyExpense

Установочный пакет программного средства – JourneyExpenseSetup.exe

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема «Разработка программного средства для расчета стоимости поездки на автомобиле»

Обозначение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ КП Т.091018.401\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработчик Самойлов А.С. Руководитель Коропа Е.Н.

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), ПЗ\_СамойловТ091.docх |  |  |
| ГЧ, Диаграмма вариантов использования.docx |  |  |
| ГЧ, Диаграмма классов.docx |  |  |
| ГЧ, Диаграмма деятельности.docx |  |  |
| Папка с проектом «JourneyExpense» |  |  |
| Установочный пакет программного средства «JourneyExpenseSetup.exe» |  |  |
| Тип носителя: оптический компакт-диск |  |  |