|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |
|  |

|  |
| --- |
|  |
| **КУРСОВОЙ ПРОЕКТ** |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и защиты баз данных. |
|  |
| на тему: Разработка информационной системы для проката автомобилей |
|  |
| Студента группы 090203-9о-20/1  специальности 09.02.03 Программирование в компьютерных системах |
| Разухина Максима Николаевича |

|  |  |  |
| --- | --- | --- |
| Студент |  | М.Н. Разухин |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 1 |  | И.А. Миланова |

Москва

2023

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 ОБЩАЯ ЧАСТЬ 5

1.1 Назначение и цели создания системы 5

1.1.1 Объект исследования 5

1.1.2 Предмет исследования 5

1.1.3 Основная цель проекта 5

1.2 Обзор и анализ предметной области 5

1.3 Жизненный цикл базы данных 7

1.3.1 Анализ предметной области 7

1.3.2 Проектирование 8

1.3.3 Реализация 8

1.3.4 Тестирование 9

1.3.5 Эксплуатация и сопровождение 9

1.4 Выбор и характеристика СУБД 9

1.5 Выбор и характеристика среды разработки приложения 10

2 СПЕЦИАЛЬНАЯ ЧАСТЬ 12

2.1 Постановка задачи 12

2.2 Архитектура информационной системы 12

2.3 Логическая модель базы данных 13

2.4 Нормализация таблиц 14

2.5 Описание таблиц 17

2.6 Другие объекты базы данных 21

2.7 Разработка приложения 24

2.7.1 Диаграмма вариантов использования 24

2.7.2 Решение главной задачи проекта 24

2.7.3 Тестирование приложения 48

2.7.4 Защита информационной системы 49

2.8 Инструкция пользователю 49

2.8.1 Общие сведения об информационной системе 50

2.8.2 Требования к техническим средствам 51

2.8.3 Требования к программным средствам 51

2.8.4 Формы ввода 51

2.8.5 Прокат автомобиля 51

ЗАКЛЮЧЕНИЕ 52

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 53

ПРИЛОЖЕНИЕ А. SQL скрипты на создание и заполнение базы данных 54

ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ 72

Окно «Входа» 72

Окно «Выбора авто» 78

Окно «Аренда автомобиля» 85

Окно «Поездка» 88

# ВВЕДЕНИЕ

Курсовой проект посвящен разработке информационной системы для проката автомобилей.

Основная цель проекта – создание приложения средствами языка программирования C#, которое позволяет клиенту арендовать автомобиль, администраторам следить за актуальностью данных автомобилей и аренд. Также данное приложение должно поддерживать подключение базы данных (MS SQL).

База данных создаётся для информационного обслуживания клиентов, желающих взять в прокат автомобиль, а также администраторов для поддержки актуальности базы данных. БД должна содержать данные о автомобилях, пользователях, водительских удостоверениях пользователей, арендах и позволять пользователям выбирать и арендовать автомобиль на определённое количество часов.

Информационная система для проката автомобилей предназначена для пользователей и администраторов. Пользователи имеют доступ к выбору автомобиля, аренде и окну поездки. Администраторы имеют доступ к просмотру и изменению данных, а также возможности пользователей.

Информационная система для проката автомобилей является актуальной и необходимой на текущий момент времени так как, большое количество людей для повседневного пользования предпочитают брать автомобиль в прокат, а не иметь личный автомобиль.

Информационная система позволяет упростить и ускорить бизнес-процессы, такие как регистрация пользователей, управление данными пользователей и оформление аренды авто. Система предоставляет пользователям простую возможность выбрать и взять автомобиль в прокат, без необходимости приезжать в салон. Администраторам система предоставляет возможность просматривать информацию о пользователях, арендах, автомобилях, вести учёт данных о проделанной работе и т.д. Курсовой проект направлен на достижение этих целей.

При разработке приложения были рассмотрены уже существующие приложения в данной предметной области, такие как «Яндекс Drive».

# ОБЩАЯ ЧАСТЬ

## Назначение и цели создания системы

### Объект исследования

Информационная система для проката автомобилей.

### Предмет исследования

Настольное приложение для проката автомобилей, взаимодействующее с базой данных MSSQL Server.

### Основная цель проекта

Разработка информационной системы для проката автомобилей, которая позволяет пользователю выбрать автомобиль и арендовать.

## Обзор и анализ предметной области

Приложение создаётся для удалённого обслуживания пользователей, а также для администрирования информационного ресурса. В базе данных должна храниться следующая информация: свободные авто для аренды, статус пользователей, дату и время аренды, координаты автомобилей и т.д.

В соответствии с предметной областью система строится с учётом следующих особенностей:

* Пользователь может арендовать только один автомобиль одновременно.
* Администраторы вносят изменения в информационный ресурс, путём изменения, удаления, добавления записей.
* Администраторы могут использовать систему как пользователи.

Выделим базовые сущности этой предметной области:

**Классы**. Атрибуты классов – код класса.

**Транспортные средства**. Атрибуты транспортных средств – код транспортного средства, фото транспортного средства, код класса, цена за час.

**Свидетельства о регистрации транспортных средств**. Атрибуты свидетельства о регистрации транспортных средств – серия, номер, код транспортного средства регистрационные номера, VIN, тип транспортного средства, категория транспортного средства, год выпуска, шасси, номер кузова, цвет, мощность двигателя, экологический класс, серия паспорта транспортного средства, номер паспорта транспортного средства, разрешенная масса транспортного средства, масса без нагрузки.

**Координаты транспортных средств**. Атрибуты координат транспортных средств – код координат, код транспортного средства, долгота, широта, дата и время.

**Пользователи.** Атрибуты пользователей – код пользователя, логин, код почтового ящика, пароль, код водительского удостоверения, фамилия, имя, отчество, дата рождения.

**Администраторы.** Атрибуты администраторов – код администратора, код пользователя.

**Водительские удостоверения.** Атрибуты водительских удостоверений – код водительского удостоверения, дата получения, код пользователя.

**Категории водительских удостоверений.** Атрибуты категорий водительских удостоверений – код категории, код водительского удостоверения, категория, дата получения, дата окончания.

**Аренды.** Атрибуты аренд – код аренды, код транспортного средства, код водительского удостоверения, дата начала аренды, время аренды, количество часов, цена, статус аренды.

**Типы ДТП.** Атрибуты типов ДТП – код типа ДТП, название типа ДТП.

**ДТП.** Атрибуты ДТП – код ДТП, код транспортного средства, код типа ДТП, код водительского удостоверения, цена повреждений, описание ДТП.

**Журнал.** Атрибуты журнала – код записи, запись.

## Жизненный цикл базы данных

Жизненный цикл базы данных включает следующие этапы:

* Анализ предметной области,
* Проектирование,
* Реализация,
* Тестирование,
* Эксплуатация и сопровождение.

### Анализ предметной области

Для разработки и построения логической модели базы данных будет использоваться ER Assistant. Для реализации физической базы данных проекта будет использован Microsoft SQL Server где будет храниться вся информация. Для написания программного продукта требуется Visual Studio 2022.

Разработчик располагает всеми необходимыми средствами для разработки баз данных: ПО построения логических моделей БД (ER Asistant), СУБД (MS SQL SERVER 2019), среда разработки ПО (Visual Studio 2022 Community).

Необходимо создать настольное приложение для проката автомобилей, взаимодействующее с базой данных SQL.

Требования к проекту:

* База данных должна быть расширяемой для добавления новых пользователей, администраторов, автомобилей, водительских удостоверений, и т.д.,
* Авторизированный пользователь должен иметь возможность просматривать список всех доступных автомобилей для аренды.

### Проектирование

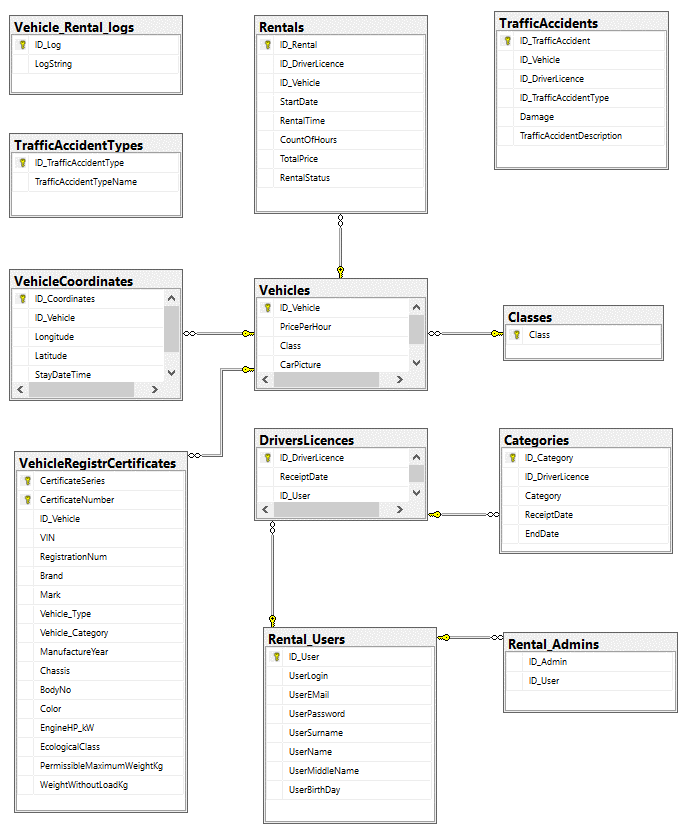


Рисунок 2 – Физическая модель базы данных

### Реализация

Разработка программы, позволяющей работать с БД производилась в Visual Studio 2022 на языке C#, на платформе Windows Presentation Foundation (WPF)

### Тестирование

Тестирование и отладка программного обеспечения выполнялись во время реализации. Все ошибки, выявленные при запуске или работе с приложением – устранены. Тестирование прошло успешно.

### Эксплуатация и сопровождение

Программный продукт готов к использованию и дальнейшему сопровождению.

## Выбор и характеристика СУБД

Для разработки информационной системы была выбрана реляционная СУБД «MS SQL SERVER 2019». Так как в системе предусматривалась концепция клиент-серверного взаимодействия, которая включает в себя сервер с базой данных и клиентов которых обслуживает этот сервер.

Данная архитектура была выбрана по причине того, что клиенты и сервер могут работать на разных устройствах в сети.

Выбранная архитектура позволяет распределить нагрузку на устройствах и обеспечить высокую производительность приложений, масштабировать приложения в случае необходимости. Также данная архитектура позволяет сделать части информационной системы независимыми, что делает возможным независимое изменение или обновление конкретной части.

Выбор СУБД MS SQL SERVER 2019 был обусловлен тем, что – эта СУБД обладает огромным сообществом пользователей, имеет высокую производительность и скорость работы с базами данных, бесплатное обслуживание, обеспечивает безопасность и защиту данных, имеет обширный функционал и гибкость настройки, а также имеет отличную совместимость с другими программными продуктами компании Microsoft.

Преимущества платформы:

* Высокая производительность,
* Безопасность,
* Надёжность.
* Масштабируемость

Из недостатков можно выделить то, что, MSSQL SERVER работает только на операционной системе Windows.

## Выбор и характеристика среды разработки приложения

Для разработки приложения была выбрана интегрированная среда разработки «Visual Studio 2022», включающая [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств, язык программирования c#, и модуль Entity Framework.

C# — это высокоуровневый объектно-ориентированный язык программирования общего назначения, разработанный компанией Microsoft в рамках платформы .NET Framework, который может использоваться для разработки разнообразных типов приложений для операционных систем Windows, macOS, Android, iOS, Linux, а также для веб-приложений и сервисов.

Entity Framework — это современный модуль сопоставления отношений объектов, который позволяет создавать чистый, переносимый и высокоуровневый уровень доступа к данным с помощью .NET (C#) в различных базах данных. Он поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы.

Entity Framework позволяет ускорить разработку приложений связанных с базами данных, а также упростить написание кода разработчику.

Visual Studio является лучшей IDE для C#, так как она также является продуктом корпорации Microsoft, как и C#. Microsoft старается создавать продукты с расчётом на то что они будут максимально взаимодействовать между собой.

Visual Studio 2022 позволяет создавать не только [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но и, приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологий Windows Presentation Foundation и [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), и т.д.

В число преимуществ Visual Studio входит следующее:

* Наличие бесплатной версии;
* Множество встроенных инструментов, позволяющих комфортно работать на C#;
* Поддержка .Net платформы;
* Возможность командной разработки.

# **СПЕЦИАЛЬНАЯ ЧАСТЬ**

## Постановка задачи

Средствами MS SQL SERVER спроектировать базу данных и создать приложение средствами C# и WPF, позволяющее, реализовать систему для проката автомобилей.

Система должна позволять

* Зарегистрировать пользователя;
* Осуществлять поиск автомобиля на карте, с использованием карты Google Maps;
* Выбор автомобиля по заданным критериям из списка имеющихся автомобилей;
* Арендовать выбранный автомобиль;
* Поддержку базы данных в актуальном состоянии;

Обеспечить защиту базы данных и приложения.

## Архитектура информационной системы

База данных

Сервер

Клиент

Рисунок 3 – Клиент-серверная модель

Архитектура информационной системы представляет собой клиент-серверную модель взаимодействия с базой данных. На сервере хранится база данных, в свою очередь пользователь клиент-приложения может взаимодействовать с сервером. Сначала пользователь должен пройти аутентификацию или регистрацию, Далее пользователь может проходить авторизацию, просматривать данные об авто, выбирать авто, арендавать авто и т.д. Сервер в данном случае - это компьютер пользователя. Клиент-приложение сначала обращается к серверу, только после этого сервер уже обращается к базе данных и получает все необходимые данные, которые нужны пользователю.

Преимущества:

* Все данные хранятся на сервере, который защищён лучше большинства клиентов;
* Распределение функций вычислительной системы между несколькими независимыми компьютерами в сети;
* Высокая надёжность;

Недостатки:

* Высокая стоимость оборудования.
* Информационная система полностью зависит от сервера;
* Сопровождение данной информационной системы требует отдельного специалиста — системного администратора,

## Логическая модель базы данных

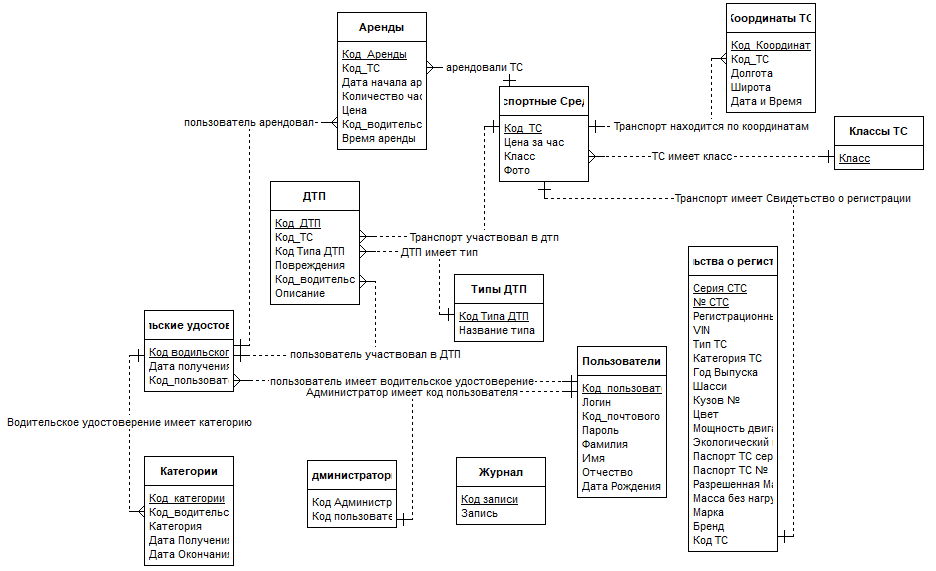


Рисунок 1 – Логическая модель базы данных

## Нормализация таблиц

Таблицы были приведены к первой нормальной форме, все атрибуты просты, далее неделимы.

Таблицы приведены ко второй нормальной форме так как каждый не ключевой атрибут функционально полно зависит от ключа.

Таблицы были приведены к третьей нормальной форме, отсутствует транзитивная зависимость.

Код класса\*

Рисунок 4 – «Классы»

Код транспортного средства\*

Серия СТС

Номер СТС

Код класса

Фото транспортного средства

Цена за час

Рисунок 5 – «Транспортные средства»

Код координат\*

Код транспортного средства

Долгота

Широта

Дата и время

Рисунок 6 – «Координаты транспортных средств»

Серия\*

Номер\*

код транспортного средства

Регистрационные номера

VIN

Тип транспортного средства

Категория транспортного средства

Год выпуска

Шасси

Номер кузова

Цвет

Мощность двигателя

Экологический класс

Серия паспорта транспортного средства

Номер паспорта транспортного средства

Разрешенная масса транспортного средства

Масса без нагрузки

Рисунок 7 – «Свидетельства о регистрации транспортных средств»

Код пользователя\*

Логин

Код почтового ящика

Пароль

Фамилия

Имя

Отчество

Дата рождения

Рисунок 8 – «Пользователи»

Код администратора\*

Код пользователя

Рисунок 9 – «Администраторы»

Код водительского удостоверения\*

Дата получения

Код пользователя

Рисунок 10 – «Водительские удостоверения»

Код категории\*

Код водительского удостоверения

Категория

Дата получения

Дата окончания

Рисунок 11 – «Категории водительских удостоверений»

Код аренды\*

Код транспортного средства

Код водительского удостоверения

Дата начала аренды

Время аренды

Количество часов

Цена

Статус аренды

Рисунок 12 – «Аренды»

Код типа ДТП\*

Название типа ДТП

Рисунок 13 – «Типы ДТП»

Код ДТП\*

Код транспортного средства

Код типа ДТП

Код водительского удостоверения

Цена повреждений

Описание ДТП

Рисунок 14 – «ДТП»

Код записи\*

Запись

Рисунок 15 – «Журнал»

## Описание таблиц

Таблица 1 - Classes (Классы)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| Class | Код класса | CHAR(20) | PK, NOT NULL |

Таблица 2 - Vehicles (Транспортные средства)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Vehicle | Код транспортного средства | INT | PK, NOT NULL |
| PricePerHour | Цена за час | MONEY | NOT NULL |
| Class | Код класса | CHAR(20) | FK, NOT NULL |
| CarPicture | Фото | VARBINARY  (MAX) |  |

Таблица 3 - VehicleRegistrCertificates (Свидетельства о регистрации транспортных средств)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| CertificateSeries | Серия\* | INT | PK, NOT NULL |
| СertificateNumber | Номер\* | INT | PK, NOT NULL |
| ID\_Vehicle | Код транспортного средства | INT | FK, NOT NULL |
| VIN | VIN | CHAR(14) | NOT NULL |
| RegistrationNum | Регистрационные знаки | CHAR(10) | NOT NULL |
| Brand | Бренд | NVARCHAR(100) | NOT NULL |
| Mark | Марка | NVARCHAR(100) | NOT NULL |
| Vehicle\_Type | Тип транспортного средства | VARCHAR(100) | NOT NULL |
| Vehicle\_Category | Категория транспортного средства | VARCHAR(6) | NOT NULL |
| ManufactureYear | Год выпуска | INT | NOT NULL |
| Chassis | Шасси | VARCHAR(100) | NOT NULL |
| BodyNo | Номер кузова | VARCHAR(40) | NOT NULL |
| Color | Цвет | VARCHAR(120) | NOT NULL |
| Название столбца | Описание | Тип | Примечание |
| EngineHP\_kW | Мощность двигателя | FLOAT | NOT NULL |
| EcologicalClass | Экологический класс | CHAR(100) | NOT NULL |
| PermissibleMaximum-  WeightKg | Разрешенная масса транспортного средства | INTEGER | NOT NULL |
| WeightWithoutLoadKg | Масса без нагрузки | INTEGER | NOT NULL |

Таблица 4 - VehicleCoordinates (Координаты транспортных средств)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Coordinates | Код координат | INT | PK, NOT NULL |
| ID\_Vehicle | Код транспортного средства | INT | FK, NOT NULL |
| Longitude | Долгота | FLOAT | NOT NULL |
| Latitude | Широта | FLOAT | NOT NULL |
| StayDateTime | Дата и время | DATETIME | NOT NULL |

Таблица 5 - Rental\_Users (Пользователи)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_User | Код пользователя\* | INT | PK, NOT NULL |
| UserLogin | Логин | NVARCHAR(120) | NOT NULL |
| UserEMail | Код почтового ящика | NVARCHAR(120) | NOT NULL |
| UserPassword | Пароль | NVARCHAR(120) | NOT NULL |
| UserSurname | Фамилия | NVARCHAR(120) | NOT NULL |
| UserName | Имя | NVARCHAR(120) | NOT NULL |
| UserMiddleName | Отчество | NVARCHAR(120) | NOT NULL |
| UserBirthDay | Дата рождения | DATE | NOT NULL |

Таблица 6 - Rental\_Admins (Администраторы)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Admin | Код администратора | INT | PK, NOT NULL |
| ID\_User | Код пользователя | INT | FK, NOT NULL |

Таблица 7 - DriversLicences (Водительские удостоверения)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | PK, NOT NULL |
| ReceiptDate | Дата получения | DATE | NOT NULL |
| ID\_User | Код пользователя | INT | FK, NOT NULL |

Таблица 8 - Categories (Категории водительских удостоверений)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Category | Код категории | INT | PK, NOT NULL |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | NOT NULL |
| Category | Категория | CHAR(3) | NOT NULL |
| ReceiptDate | Дата получения | DATE | NOT NULL |
| EndDate | Дата окончания | DATE | NOT NULL |

Таблица 9 - Rentals (Аренды)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Rental | Код аренды | INT | PK, NOT NULL |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | FK, NULL |
| ID\_Vehicle | Код транспортного средства | INT | FK, NULL |
| StartDate | Дата начала аренды | DATE | NOT NULL |
| Название столбца | Описание | Тип | Примечание |
| RentalTime | Время аренды | TIME | NOT NULL |
| CountOfHours | Количество часов | INT | NOT NULL |
| TotalPrice | Цена | MONEY | NOT NULL |

Таблица 10 - TrafficAccidentTypes (Типы ДТП)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_TrafficAccidentType | Код типа ДТП | INT | PK, NOT NULL |
| TrafficAccidentTypeName | Название типа ДТП | NVARCHAR(60) | NOT NULL |

Таблица 11 - TrafficAccidents (ДТП)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_TrafficAccident | Код ДТП | INT | PK, NOT NULL |
| ID\_Vehicle | Код транспортного средства | INT | FK, NULL |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | FK, NULL |
| ID\_TrafficAccidentType | Код типа ДТП | INT | FK, NULL |
| Damage | Цена повреждений | MONEY | NOT NULL |
| TrafficAccidentDescription | Описание ДТП | NVARCHAR  (MAX) | NOT NULL |

Таблица 12 – Vehicle\_Rental\_Logs

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Log | Код записи | INT | PK, NOT NULL |
| LogString | Запись | NARCHAR(MAX) | PK, NOT NULL |

## Другие объекты базы данных

Представление 1 – RentalsINFO

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Rental | Код аренды | INT | NOT NULL |
| ID\_Vehicle | Код транспортного средства | INT | NULL |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | NULL |
| RentalStatus | Статус аренды | CHAR(11) | NOT NULL |
| TotalPrice | Общая стоимость аренды | MONEY | NOT NULL |
| EndTime | Время окончания аренды | DATETIME | NOT NULL |

Представление 2 – UsersINFO

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_User | Код пользователя | INT | NOT NULL |
| UserEMail | Код почтового ящика | NVARCHAR(120) | NOT NULL |
| UserSurname | Фамилия | NVARCHAR(120) | NOT NULL |
| UserName | Имя | NVARCHAR(120) | NOT NULL |
| UserMiddleName | Отчество | NVARCHAR(120) | NOT NULL |
| UserBirthDay | Дата рождения | DATE | NOT NULL |
| Previlege | Права доступа к базе данных | CHAR(24) | NULL |
| RentalsCount | Общее количество аренд | INT | NOT NULL |
| AccidentsCount | Общее количество аварий | INT | NOT NULL |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | NULL |
| ReceiptDate | Дата получения | DATETIME | NULL |
| RentStatus | Дата окончания | DATETIME | NULL |

Представление 3 – VehiclesINFO

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_Vehicle | Код транспортного средства | INT | NOT NULL |
| Brand | Бренд | NVARCHAR(100) | NOT NULL |
| Mark | Марка | NVARCHAR(100) | NOT NULL |
| Название столбца | Описание | Тип | Примечание |
| Color | Цвет | VARCHAR(120) | NOT NULL |
| Class | Код класса | CHAR(20) | NOT NULL |
| Vehicle\_Category | Категория транспортного средства | VARCHAR(6) | NOT NULL |
| PricePerHour | Цена за час | MONEY | NOT NULL |
| CarPicture | Фото | VARBINARY  (MAX) | NULL |
| Lat | Широта | FLOAT | NOT NULL |
| Lng | Долгота | FLOAT | NOT NULL |
| AccessStatus | Статус доступности транспортного средства | CHAR(11) | NOT NULL |
| DamageCost | Общая цена ущерба | MONEY | NULL |

Представление 4 – DriversLicencesCategoriesINFO

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| ID\_DriverLicence | Код водительского удостоверения | CHAR(10) | NOT NULL |
| Category | Категория | CHAR(3) | NOT NULL |
| ReceiptDate | Дата получения | DATE | NOT NULL |
| EndDate | Дата окончания | DATE | NOT NULL |

## Разработка приложения

### Диаграмма вариантов использования

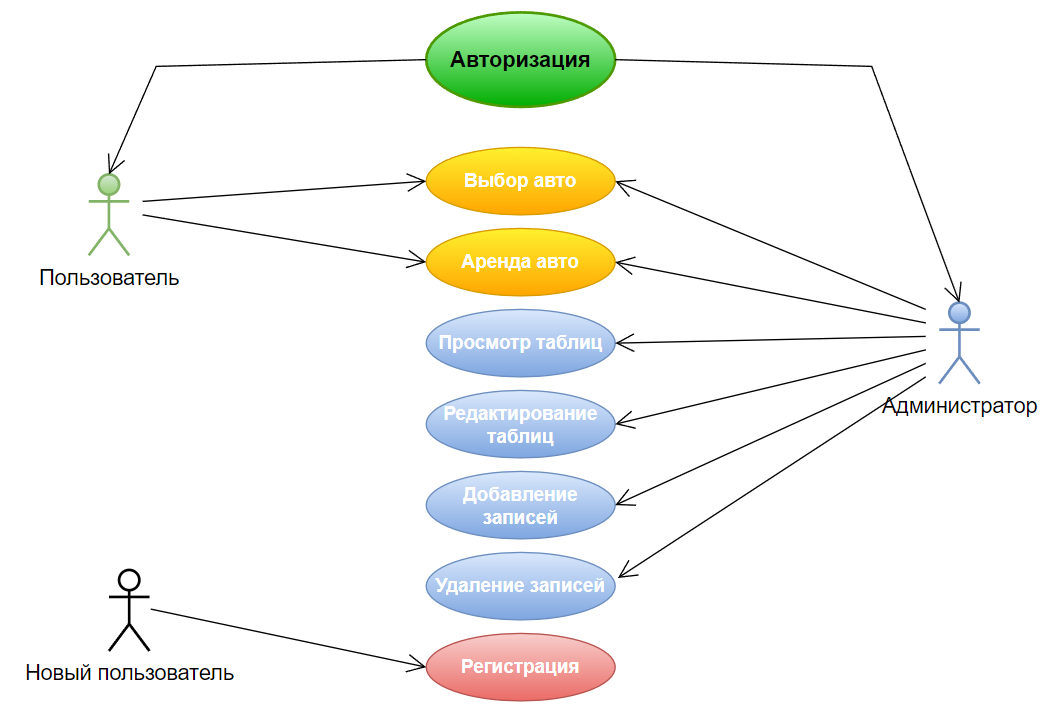


Рисунок 16 – Диаграмма вариантов использования

### Решение главной задачи проекта

Главной задачей проекта по созданию информационной системы для проката автомобилей, состоит из создания приложения, которое будет позволять брать автомобиль в прокат и эффективно управлять всеми бизнес процессами проката автомобилей: выбор конкретного автомобиля, выбор времени продолжительности проката, оплата, поездка и предоставление пользователю информации в удобной форме.

Подзадачи, которые необходимо решить для реализации главной задачи проекта:

* Пользователь регистрируется в системе
* Пользователь выбирает автомобиль
* Пользователь оплачивает прокат
* Пользователь арендует автомобиль

В данном фрагменте кода регистрация и авторизация пользователя (смотреть рисунок – 17, 18).

namespace CarSharingApplication

{

/// <summary>

/// Логика взаимодействия для Autorization.xaml

/// </summary>

public partial class Autorization : Window

{

private int condition = 0;

private string connectionString { get; set; } = App.GetConnectionString("USERHANDLERConnection");

public Autorization()

{

App.\_Logger.options = new JsonSerializerOptions

{

Encoder = JavaScriptEncoder.Create(UnicodeRanges.BasicLatin, UnicodeRanges.Cyrillic),

};

//App.\_Logger.LogPath = Environment.CurrentDirectory + @"\logs.json";

App.\_Logger.LogPath = App.GetConnectionString("LoggerConnection");

App.AppDataBase.createfunc += App.ContextCreateFunc;

VehicleEmulator.AsyncStartEmulate();

InitializeComponent();

Reg\_Button\_Click(null, null);

}

private void MakeRegister(object sender, RoutedEventArgs e)

{

bool registered = false;

if (Login.Text.Length < 5)

MessageBox.Show("Слишком мало символов в имени входа");

else if ((Email.Text.Length < 10) || (Email.Text.Count((c) => c == '@') != 1) || (Email.Text.Count((c) => c == '.') != 1) || (Email.Text.Contains("@.")) || (Email.Text.Contains(".@")))

MessageBox.Show("Почта не является корректной");

else if (Password.Password.Length < 8)

MessageBox.Show("Слишком короткий пароль");

else if (Password.Password != RepeatPassword.Password)

MessageBox.Show("Пароли не совпадают");

else if (UserSurname.Text.Length == 0)

MessageBox.Show("Вы не ввели фамилию");

else if (UserName.Text.Length == 0)

MessageBox.Show("Вы не ввели имя");

else if (UserMiddleName.Text.Length == 0)

MessageBox.Show("Вы не ввели отчество");

else if (BDatePicker.Text == "")

MessageBox.Show("Вы не ввели дату рождения");

else if (DateTime.Parse(BDatePicker.Text) >= DateTime.UtcNow.AddYears(-18) || DateTime.Parse(BDatePicker.Text) < DateTime.UtcNow.AddYears(-100) || DateTime.Parse(BDatePicker.Text) >= DateTime.UtcNow)

MessageBox.Show("Некоректная дата рождения");

else

{

try

{

#region NEW CODE SUPPORT FROM 25.04.2023

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

string encLogin = PasswordEncryptor.EncryptString(Login.Text);

string encPass = PasswordEncryptor.EncryptString(Password.Password);

App.AppDataBase.OpenConnection(connectionString);

if (App.AppDataBase.GetQueryResult<string>($"SELECT UserLogin FROM Rental\_Users WHERE UserLogin = '{encLogin}'").Count > 0)

{

MessageBox.Show("Логин занят");

return;

}

var answ = App.AppDataBase.GetScalarResult<int>($"SELECT [dbo].CheckExistingUser('{encLogin}','{encPass}')");

if (answ == -1)

{

if (App.AppDataBase.ExecuteNonQuery(

"EXEC REG\_USER " +

$"@UserLogin='{encLogin}', " +

$"@UserEmail='{Email.Text}', " +

$"@UserPassword='{encPass}', " +

$"@UserSurname='{UserSurname.Text}', " +

$"@UserName='{UserName.Text}', " +

$"@UserMiddleName='{UserMiddleName.Text}'," +

$"@UserBirthDayDate='{BDatePicker.Text}'"))

registered = true;

}

else if (answ > 0)

{

registered = true;

}

else

{

MessageBox.Show("Проверьте корректность введённых данных");

}

}

#endregion

}

catch (SqlException SqlEx)

{

App.\_Logger.Log(new LogMessage(null, this.Title, SqlEx.Message, SqlEx.ErrorCode.ToString(), LogType.DataBaseError));

}

finally

{

if (registered)

Login\_Button\_Click(null, null);

App.AppDataBase.CloseConnection();

}

}

}

private void MakeLogIn(object sender, RoutedEventArgs e)

{

try

{

#region NEW CODE SUPPORT FROM 25.04.2023

if (Login.Text.Length < 5)

MessageBox.Show("Слишком мало символов в имени входа");

else if (Password.Password.Length < 8)

MessageBox.Show("Слишком короткий пароль");

else

{

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

string encLogin = PasswordEncryptor.EncryptString(Login.Text);

string encPass = PasswordEncryptor.EncryptString(Password.Password);

//new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection"))

App.AppDataBase.OpenConnection(connectionString);

var answ = App.AppDataBase.GetScalarResult<int>(

$"SELECT [dbo].CheckExistingUser('{encLogin}','{encPass}')");

if (answ > 0)

{

this.Visibility = Visibility.Collapsed;

UsersINFO UserInfo = App.AppDataBase.GetScalarResult<UsersINFO>($"SELECT \* FROM UsersINFO WHERE ID\_User = {answ}");

if (UserInfo != null)

{

App.\_Logger.Log(new LogMessage((ulong)UserInfo.ID\_User, this.Title, "Вошёл в систему", null, null));

if (UserInfo.Previlege.TrimEnd() == "админ")

{

var ChoiceAuthorizationWindow = new ChoiceLoginWindow(ref UserInfo, this);

ChoiceAuthorizationWindow.Activate();

ChoiceAuthorizationWindow.Show();

}

else

{

if (UserInfo.RentStatus != "в поездке")

{

var CarSelWindow = new CarSelector(ref UserInfo, this, true);

CarSelWindow.Activate();

CarSelWindow.Show();

}

else

{

var TripWND = new TripWindow(ref UserInfo, this, true);

TripWND.Activate();

TripWND.Show();

this.Visibility = Visibility.Collapsed;

}

}

ClearFields();

}

}

else

App.\_Logger.Log(new LogMessage(null,this.Title,"Пользователь не найден","ошибка авторизации",LogType.UserMistake));

App.AppDataBase.CloseConnection();

}

}

#endregion

}

catch (SqlException SqlEx)

{

App.\_Logger.Log(new LogMessage(null,this.Title, SqlEx.Message, SqlEx.ErrorCode.ToString(), LogType.DataBaseError));

}

}

private void HideFields()

{

EmailGrid.Visibility = Visibility.Collapsed;

RepeatPasswordGrid.Visibility = Visibility.Collapsed;

UserSurnameGrid.Visibility = Visibility.Collapsed;

UserNameGrid.Visibility = Visibility.Collapsed;

UserMiddleNameGrid.Visibility = Visibility.Collapsed;

BDatePickerGrid.Visibility = Visibility.Collapsed;

}

private void ShowFields()

{

EmailGrid.Visibility = Visibility.Visible;

RepeatPasswordGrid.Visibility = Visibility.Visible;

UserSurnameGrid.Visibility = Visibility.Visible;

UserNameGrid.Visibility = Visibility.Visible;

UserMiddleNameGrid.Visibility = Visibility.Visible;

BDatePickerGrid.Visibility = Visibility.Visible;

}

private void Reg\_Button\_Click(object sender, RoutedEventArgs e)

{

if (condition != 2)

{

ShowFields();

lbl.Content = "Аутентификация";

inputButton.Click -= MakeLogIn;

inputButton.Click += MakeRegister;

inputButton.Content = "Зарегистрироваться";

}

}

private void Login\_Button\_Click(object sender, RoutedEventArgs e)

{

if (condition != 1)

{

HideFields();

lbl.Content = "Авторизация";

inputButton.Click -= MakeRegister;

inputButton.Click += MakeLogIn;

inputButton.Content = "Войти";

}

}

private void ClearFields()

{

Login.Clear();

Email.Clear();

Password.Clear();

RepeatPassword.Clear();

UserSurname.Clear();

UserName.Clear();

UserMiddleName.Clear();

BDatePicker.Text = "";

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

VehicleEmulator.StopEmulate();

}

}

}

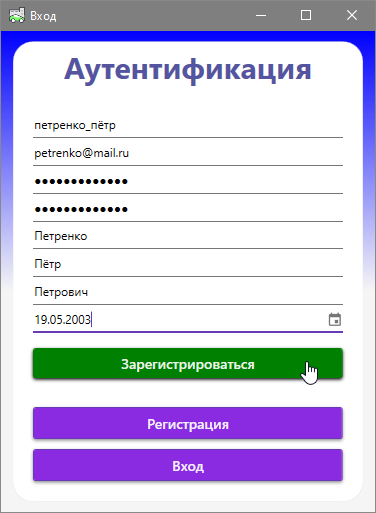


Рисунок 17 – скриншот «Регистрация пользователя»

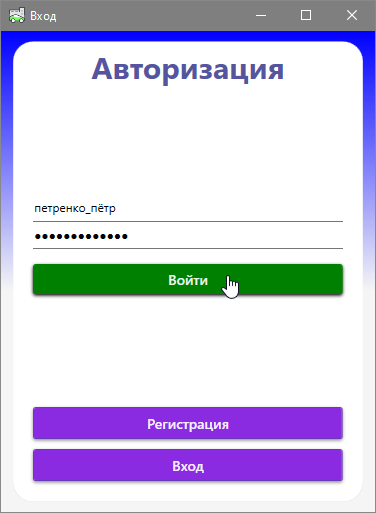


Рисунок 18 – скриншот «Авторизация пользователя»

В данном фрагменте кода осуществляется выбор автомобиля (смотреть рисунок – 19).

CREATE VIEW VehiclesINFO

AS

SELECT DISTINCT

Vehicles.ID\_Vehicle,

VehicleRegistrCertificates.Brand,

VehicleRegistrCertificates.Mark,

VehicleRegistrCertificates.Color,

Vehicles.Class,

VehicleRegistrCertificates.Vehicle\_Category,

Vehicles.PricePerHour,

Vehicles.CarPicture,

(SELECT TOP(1) Latitude

FROM VehicleCoordinates

WHERE VehicleCoordinates.ID\_Vehicle = Vehicles.ID\_Vehicle

ORDER BY StayDateTime DESC) as Lat,

(SELECT TOP(1) Longitude

FROM VehicleCoordinates

WHERE VehicleCoordinates.ID\_Vehicle = Vehicles.ID\_Vehicle

ORDER BY StayDateTime DESC) as Lng,

[dbo].GetVehicleStatus(Vehicles.ID\_Vehicle) as AccessStatus,

[dbo].TotalVehicleDamage(Vehicles.ID\_Vehicle) as DamageCost

FROM

Vehicles INNER JOIN VehicleRegistrCertificates

ON Vehicles.ID\_Vehicle = VehicleRegistrCertificates.ID\_Vehicle

namespace CarSharingApplication

{

/// <summary>

/// Логика взаимодействия для CarSelector.xaml

/// </summary>

public partial class CarSelector : Window

{

private UsersINFO \_User = null;

private bool \_ShowOwner;

private string ConnectionString { get { return App.GetConnectionString("CARHANDLERConnection"); } }

/// <summary>

/// Текст в случае если в системе нет автомобилей соответстующим критериям пользователя

/// </summary>

private string ZeroVehiclesByCriteries = "Отсутствуют транспотрные\nсредтва соответствующие\nзаданным критериям";

/// <summary>

/// Текст в случае если нет свободных автомобилей в системе

/// </summary>

private string HaveNotAvaliableVehicles = "В данный момент\nнет свободных авто\nзаходите позже";

public CarSelector(ref UsersINFO user, Window owner, bool showOwner)

{

this.Owner = owner;

\_ShowOwner = showOwner;

\_User = user;

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

InitializeComponent();

GetVehiclesData(VehiclesData.GetInstance);

try

{

var instance = VehiclesData.GetInstance;

PriceSlider.Minimum = Double.Parse((instance.vehiclesInfoList.Min(veh => veh.PricePerHour)).ToString());

PriceSlider.Maximum = Double.Parse((instance.vehiclesInfoList.Max(veh => veh.PricePerHour)).ToString());

PriceSlider.Value = PriceSlider.Maximum;

if (instance.vehiclesInfoList.Count > 0)

{

SetInfo(instance);

}

}

catch

{

PriceSlider.Minimum = 0.0;

PriceSlider.Maximum = 0.0;

PriceSlider.Value = 0.0;

SetVehicleInfo(null, HaveNotAvaliableVehicles);

}

}

/// <summary>

/// Показывать окно родителя при закрытии

/// </summary>

/// <param name="value"></param>

public void OwnerShow(bool value)

{

\_ShowOwner = value;

}

/// <summary>

/// Отобразить информацию в окно

/// </summary>

/// <param name="vehData"></param>

public void SetInfo(VehiclesData vehData)

{

vehData.selectedVehicle = vehData.vehiclesInfoList.First();

RentalMap.SetMarkers(GetMarkers(vehData.vehiclesInfoList));

SetVehicleInfo(vehData.selectedVehicle, ZeroVehiclesByCriteries);

}

/// <summary>

/// Получение/обновление данных о авто

/// </summary>

/// <param name="vehData"></param>

private void GetVehiclesData(VehiclesData vehData)

{

App.AppDataBase.OpenConnection(ConnectionString);

vehData.vehClasses = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(UPPER(Class)) FROM VehiclesINFO");

vehData.vehClasses.Add("\*ВСЕ");

ListViewVehicleClasses.ItemsSource = vehData.vehClasses.OrderBy(str => str);

vehData.vehBrands = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(UPPER(Brand)) FROM VehiclesINFO");

vehData.vehBrands.Add("\*ВСЕ");

ListViewVehicleBrands.ItemsSource = vehData.vehBrands.OrderBy(str => str);

vehData.vehiclesInfoList = App.AppDataBase.GetQueryResult<VehiclesINFO>(

"SELECT \* FROM VehiclesWithStatus ('доступен')");

vehData.vehiclesInfoList = OrderByPricePerHourDesc(vehData.vehiclesInfoList);

vehData.vehCategories = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(Vehicle\_Category) FROM VehiclesINFO");

vehData.vehCategories.Add("\*ВСЕ");

ListViewVehicleCategories.ItemsSource = vehData.vehCategories.OrderBy(str => str);

}

/// <summary>

/// Нажатие кнопки, выхода

/// </summary>

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

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

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

{

this.Close();

}

/// <summary>

/// Получить список маркеров из списка информации об авто

/// </summary>

/// <param name="vehiclesINFOs"></param>

/// <returns></returns>

private List<GMapMarker> GetMarkers(List<VehiclesINFO> vehiclesINFOs)

{

try

{

List<GMapMarker> VehiclesMarkers = new List<GMapMarker>();

foreach (var vehicle in vehiclesINFOs)

{

RentalMap.MapController.Markers.OrderBy(mark => mark.Tag);

if (vehicle.Lat != null && vehicle.Lng != null)

{

GMapMarker marker = new GMapMarker(new PointLatLng((double)vehicle.Lat, (double)vehicle.Lng));

marker.Tag = vehicle.ID\_Vehicle;

marker.Shape = new System.Windows.Controls.Image

{

Source = new BitmapImage(new Uri($@"{App.path}\Windows\Images\CarMarker2.png")),

Width = 30,

Height = 30,

ToolTip = $"{vehicle.Brand} {vehicle.Mark}",

Visibility = Visibility.Visible,

Tag = vehicle

};

marker.Shape.MouseEnter += (s, e) =>

{ Mouse.OverrideCursor = Cursors.Hand;};

marker.Shape.MouseLeave += (s, e) =>

{ Mouse.OverrideCursor = Cursors.Arrow; };

marker.Shape.MouseLeftButtonDown += MarkerClick;

VehiclesMarkers.Add(marker);

}

}

return VehiclesMarkers;

}

catch (SqlException sqelx)

{

MessageBox.Show(sqelx.Message);

return null;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

}

/// <summary>

/// Вывести информацию о автомобиле

/// </summary>

/// <param name="info"></param>

/// <param name="errorMessage"></param>

#nullable enable

private void SetVehicleInfo(VehiclesINFO? info, string errorMessage)

{

Card.SetVehicleInfo(info, errorMessage);

}

/// <summary>

/// Нажатие на маркер автобиля

/// </summary>

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

/// <param name="args"></param>

private void MarkerClick(object sender, MouseEventArgs args)

{

var marker = (System.Windows.Controls.Image)sender;

VehiclesData.GetInstance.selectedVehicle = (VehiclesINFO)(marker.Tag);

SetVehicleInfo(VehiclesData.GetInstance.selectedVehicle, ZeroVehiclesByCriteries);

RentalMap.MoveCursorToVehicleOnMap(VehiclesData.GetInstance.selectedVehicle);

}

/// <summary>

/// Найти автомобиль/автомобили по критериям

/// </summary>

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

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

private void SearchByCriteries(object sender, RoutedEventArgs e)

{

var instance = VehiclesData.GetInstance;

GetVehiclesData(instance);

List<VehiclesINFO> newvehicleslist = instance.vehiclesInfoList.Where(vehicle => Double.Parse(vehicle.PricePerHour.ToString()) <= PriceSlider.Value).ToList();

#nullable enable

if ((string)ListViewVehicleClasses.SelectedValue != "\*ВСЕ" && ListViewVehicleClasses.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Class.ToUpper().TrimEnd() == (string)ListViewVehicleClasses.SelectedValue).ToList();

}

#nullable enable

if ((string)ListViewVehicleBrands.SelectedValue != "\*ВСЕ" && ListViewVehicleBrands.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Brand.ToUpper().TrimEnd() == (string)ListViewVehicleBrands.SelectedValue).ToList();

}

#nullable enable

if ((string)ListViewVehicleCategories.SelectedValue != "\*ВСЕ" && ListViewVehicleCategories.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Vehicle\_Category.TrimEnd() == (string)ListViewVehicleCategories.SelectedValue).ToList();

}

newvehicleslist = OrderByPricePerHourDesc(newvehicleslist);

instance.vehiclesInfoList = newvehicleslist;

RentalMap.MapController.Markers.Clear();

GC.Collect();

RentalMap.SetMarkers(GetMarkers(newvehicleslist));

if (newvehicleslist.Count > 0)

{

instance.selectedVehicle = newvehicleslist.First();

SetVehicleInfo(instance.selectedVehicle, ZeroVehiclesByCriteries);

RentalMap.MoveCursorToVehicleOnMap(instance.selectedVehicle);

}

else

{

instance.selectedVehicle = null;

SetVehicleInfo(null, ZeroVehiclesByCriteries);

}

}

/// <summary>

/// Сортировать автомобили по цене

/// </summary>

/// <param name="list"></param>

/// <returns>List<VehiclesINFO</returns>

public List<VehiclesINFO> OrderByPricePerHourDesc(List<VehiclesINFO> list) => (from vehicle

in list

orderby vehicle.PricePerHour descending

select vehicle).ToList();

/// <summary>

/// Нажатие кнопки Арендовать

/// </summary>

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

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

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

{

var instance = VehiclesData.GetInstance;

if (instance.selectedVehicle != null)

{

var rentWindow = new VehicleRent(\_User, instance.selectedVehicle, this, true);

this.Visibility = Visibility.Collapsed;

rentWindow.Activate();

rentWindow.Show();

}

else

{

MessageBox.Show("Вы не выбрали авто");

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, "Пользователь не выбрал ТС", "Ошибка нажатия кнопки аренды ТС", 0));

}

}

/// <summary>

/// Нажатие кнопки Личный кабинет

/// </summary>

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

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

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

{

var persWindow = new PersonalAccount(ref \_User);

persWindow.Owner = this;

persWindow.Show();

this.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Закрытие окна

/// </summary>

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

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

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

GC.Collect();

}

else

{

this.Owner.Close();

}

}

/// <summary>

/// Показать информацию о следующем автомобиле

/// </summary>

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

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

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

{

var instance = VehiclesData.GetInstance;

var index = instance.vehiclesInfoList.IndexOf(instance.selectedVehicle);

if (instance.vehiclesInfoList.Count > 0)

{

if ((index < instance.vehiclesInfoList.Count-1))

{

instance.selectedVehicle = instance.vehiclesInfoList[index + 1];

}

else

{

instance.selectedVehicle = instance.vehiclesInfoList[0];

}

SetVehicleInfo(instance.selectedVehicle, ZeroVehiclesByCriteries);

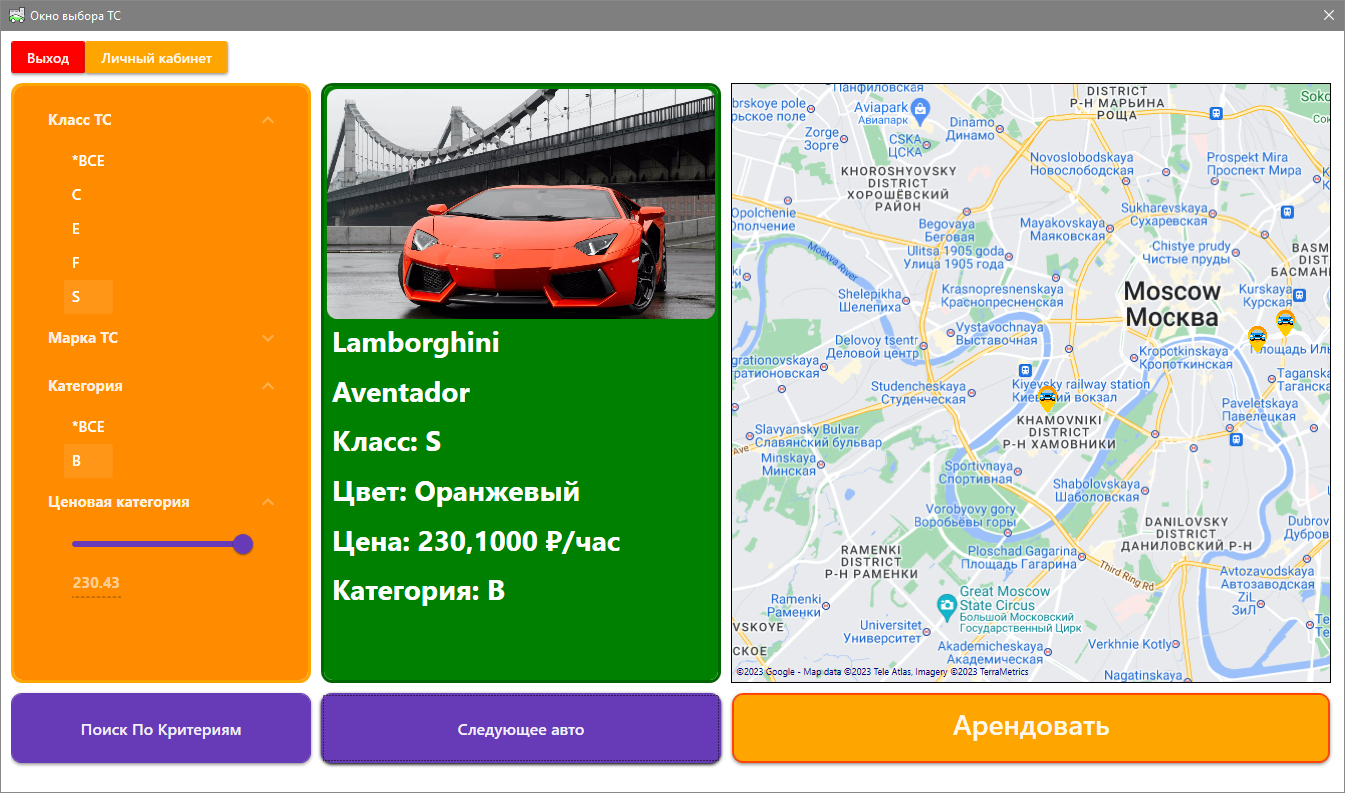
RentalMap.MoveCursorToVehicleOnMap(instance.selectedVehicle);

}

}

}

}



Риcунок 19 – скриншот «Выбор автомобиля»

В данном фрагменте кода осуществляется аренда автомобиля (смотреть рисунок – 20).

CREATE PROCEDURE Rent(

@DriverLicence CHAR(10),

@ID\_Vehicle INT,

@RentalTime TIME,

@CountOfHours INT,

@TotalPrice MONEY

)

AS

BEGIN TRY

BEGIN TRANSACTION RentVehicle

IF NOT ((SELECT [dbo].GetVehicleStatus(@ID\_Vehicle)) = 'доступен')

ROLLBACK TRANSACTION RentVehicle

INSERT Rentals VALUES

(@DriverLicence, @ID\_Vehicle, GETDATE(), @RentalTime, @CountOfHours, @TotalPrice, 'стандартная')

COMMIT TRANSACTION RentVehicle

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION RentVehicle

END CATCH

namespace CarSharingApplication.Windows.VehicleRent

{

/// <summary>

/// Логика взаимодействия для VehicleRent.xaml

/// </summary>

public partial class VehicleRent : Window

{

private UsersINFO \_User;

private VehiclesINFO \_Vehicle;

private bool \_ShowOwner;

private string DLHANDLERconnectionString { get; set; } = App.GetConnectionString("DLHANDLERConnection");

private string USERHANDLERconnectionString { get; set; } = App.GetConnectionString("USERHANDLERConnection");

public VehicleRent(UsersINFO user, VehiclesINFO Vehicle, Window owner, bool showOwner)

{

\_ShowOwner = showOwner;

this.Owner = owner;

InitializeComponent();

\_User = user;

\_Vehicle = Vehicle;

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

Card.SetVehicleInfo(Vehicle,"");

Picker.PricePerHour = (double)Vehicle.PricePerHour;

rentbtn.btn.Click += PayAndStart\_Click;

}

/// <summary>

/// Нажатие кнопки:

/// Оплатить и начать поездку

/// </summary>

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

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

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

{

if (rentbtn.cbox.IsChecked != true || CreditCard.isEmpty())

return;

if (\_User.ID\_DriverLicence == null)

{

MessageBox.Show("Вы не ввели данные о вод.удостоверении\nP.S. Это делается в личном кабинете");

this.Close();

return;

}

App.AppDataBase.OpenConnection(DLHANDLERconnectionString);

List<string> categories = App.AppDataBase.GetQueryResult<string>(

$"SELECT Category FROM [dbo].GetDriverLicenceCategories ('{\_User.ID\_DriverLicence}')");

App.AppDataBase.CloseConnection();

if (categories == null)

return;

if (!(categories.Contains(\_Vehicle.Vehicle\_Category.Trim().ToLower())))

{

MessageBox.Show("У вас отсутсвтует нужная категория в водительском удостоверении");

return;

}

App.AppDataBase.OpenConnection(USERHANDLERconnectionString);

if (App.AppDataBase.ExecuteNonQuery(

"EXEC Rent " +

$"@DriverLicence = '{\_User.ID\_DriverLicence}', " +

$"@ID\_Vehicle = {\_Vehicle.ID\_Vehicle}, " +

$"@RentalTime = '{DateTime.Now.ToString("HH:mm:ss")}', " +

$"@CountOfHours = {Picker.HourPicker.Value}, " +

$"@TotalPrice = {(double.Parse(\_Vehicle.PricePerHour.ToString()) \* Picker.HourPicker.Value).ToString().Replace(',', '.')}"))

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Арендовал ТС: {\_Vehicle.ID\_Vehicle}", null, LogType.UserAction));

try

{

\_ShowOwner = false;

var TripWND = new TripWindow(ref \_User, this.Owner.Owner, true);

TripWND.Activate();

TripWND.Show();

}

catch

{

\_ShowOwner = true;

}

{

this.Visibility = Visibility.Collapsed;

}

}

App.AppDataBase.CloseConnection();

}

/// <summary>

/// Закрытие окна

/// </summary>

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

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

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

}

else

{

this.Owner?.Close();

}

}

}

}

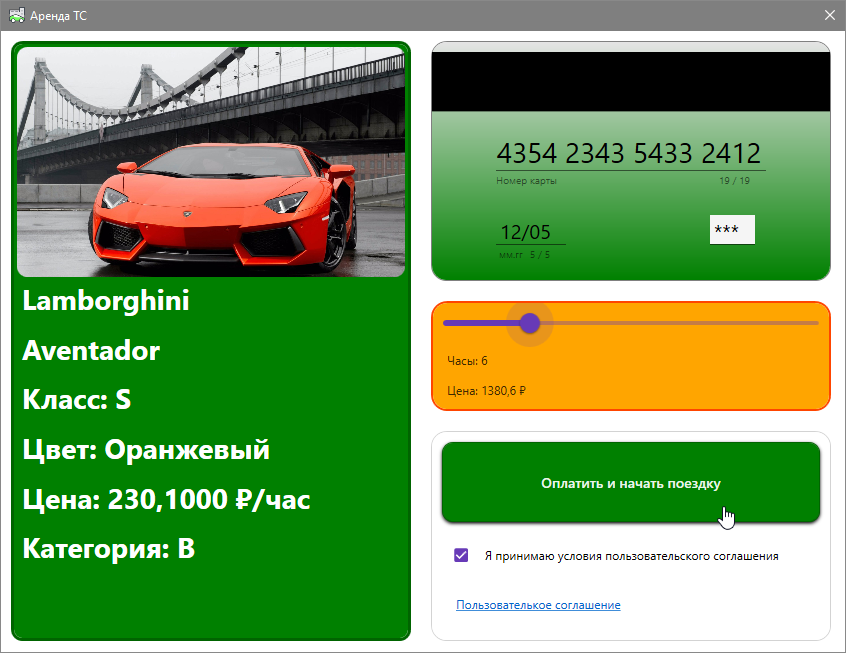


Рисунок 20 – скриншот «Аренда автомобиля»

В данном фрагменте кода осуществляется вывод информации об оренде (смотреть рисунок – 21).

CREATE VIEW RentalsINFO

AS

SELECT

ID\_Rental,

ID\_Vehicle,

ID\_DriverLicence,

RentalStatus,

TotalPrice,

DATEADD(SECOND,DATEPART(SECOND,Rentals.RentalTime),

DATEADD(MINUTE,DATEPART(MINUTE,Rentals.RentalTime),

DATEADD(HOUR,Rentals.CountOfHours+DATEPART(HOUR,Rentals.RentalTime),

CONVERT(DATETIME,StartDate)))) as EndTime

FROM Rentals

namespace CarSharingApplication.Windows.VehicleRent

{

/// <summary>

/// Логика взаимодействия для TripWindow.xaml

/// </summary>

public partial class TripWindow : Window

{

private bool \_ShowOwner;

private UsersINFO \_User;

private VehiclesINFO \_Vehicle;

private RentalsINFO \_Rental;

private string connectionString { get; set; } = App.GetConnectionString("CARHANDLERConnection");

public TripWindow(ref UsersINFO user, Window owner, bool showOwner)

{

this.Owner = owner;

this.Owner.Visibility = Visibility.Collapsed;

\_ShowOwner = showOwner;

\_User = user;

InitializeComponent();

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

App.AppDataBase.OpenConnection(connectionString);

\_Rental = App.AppDataBase.GetScalarResult<RentalsINFO>(

"SELECT TOP(1) \* FROM RentalsINFO " +

$"WHERE ID\_DriverLicence = {\_User.ID\_DriverLicence} " +

"AND EndTime > GETDATE() AND RentalStatus='стандартная'");

if (\_Rental != null)

{

rt.SetTime(\_Rental.EndTime);

\_Vehicle = App.AppDataBase.GetScalarResult<VehiclesINFO>(

$"SELECT \* FROM VehiclesINFO WHERE ID\_Vehicle = {\_Rental.ID\_Vehicle}");

Card.SetVehicleInfo(\_Vehicle, "Ошибка загрузки данных");

}

else

{

this.Close();

}

App.AppDataBase.CloseConnection();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Show();

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

}

else

{

this.Owner?.Close();

}

}

/// <summary>

/// Нажатие кнопки завершения поездки

/// </summary>

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

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

private void FinishTripClick(object sender, RoutedEventArgs e)

{

try

{

var a = MessageBox.Show("Вы уверены что хотите завершить поездку раньше срока?", "Закончить поездку", MessageBoxButton.YesNo);

if (a == MessageBoxResult.Yes)

{

App.AppDataBase.OpenConnection(connectionString);

if(App.AppDataBase.ExecuteNonQuery(

$"UPDATE Rentals SET RentalStatus='досрочная' WHERE ID\_Rental = {\_Rental.ID\_Rental}"))

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Досрочно окончил аренду ТС {\_Rental.ID\_Vehicle}", null, LogType.UserAction));

App.AppDataBase.CloseConnection();

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Не смог отменить поездку", ex.Message, LogType.ProgramError));

}

}

/// <summary>

/// Нажатие кнопки Произошла авария

/// </summary>

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

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

private void AccidentClick(object sender, RoutedEventArgs e)

{

var accidentWND = new AddAccident(ref \_User, ref \_Rental, this, true);

accidentWND.Activate();

accidentWND.Show();

this.Visibility = Visibility.Collapsed;

}

}

}

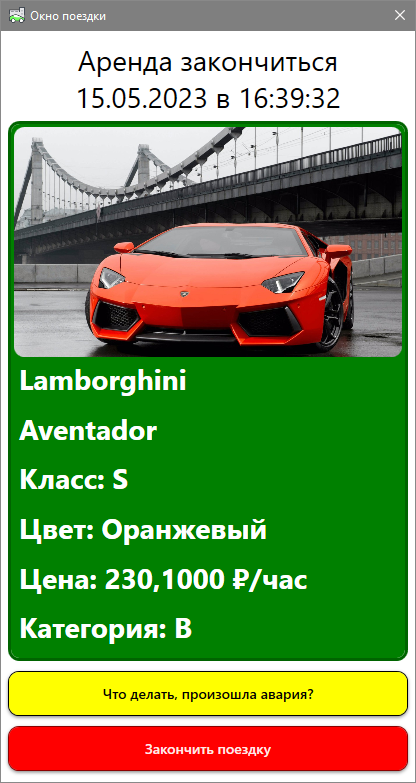


Рисунок 21 – скриншот «Окно поездки»

Поддержка базы данных в актуальном состоянии (смотреть рисуноки – 22-36):



Рисунок 22 – Скриншот «Окно администрирования»

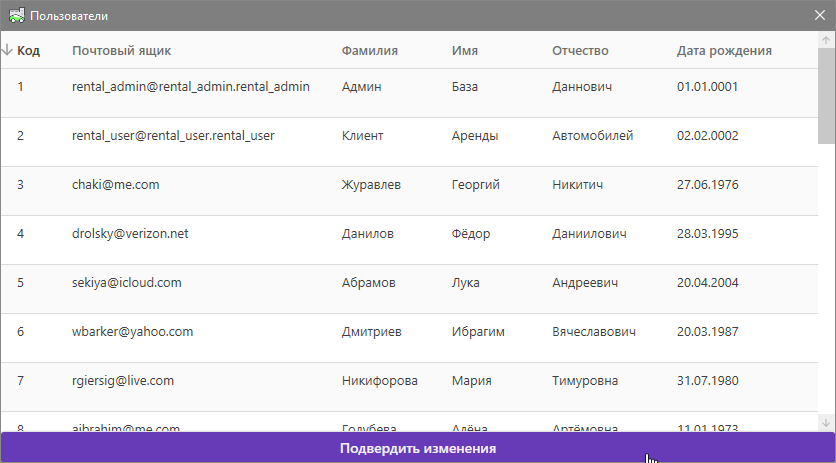


Рисунок 23 – Скриншот «Окно пользователей»

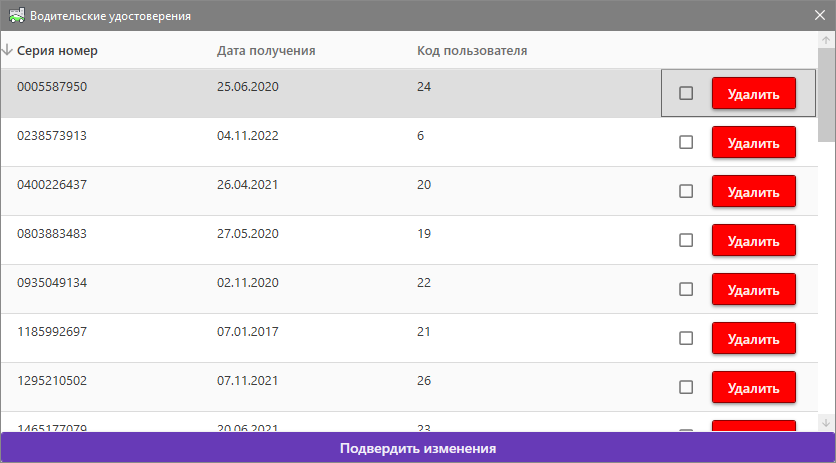


Рисунок 24 – Скриншот «Окно водительских удостоверений»

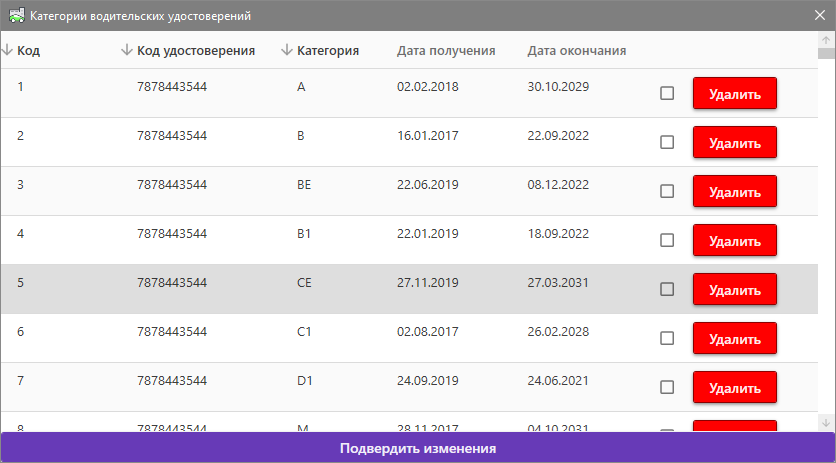


Рисунок 25 – Скриншот «Окно категорий водительских удостоверений»

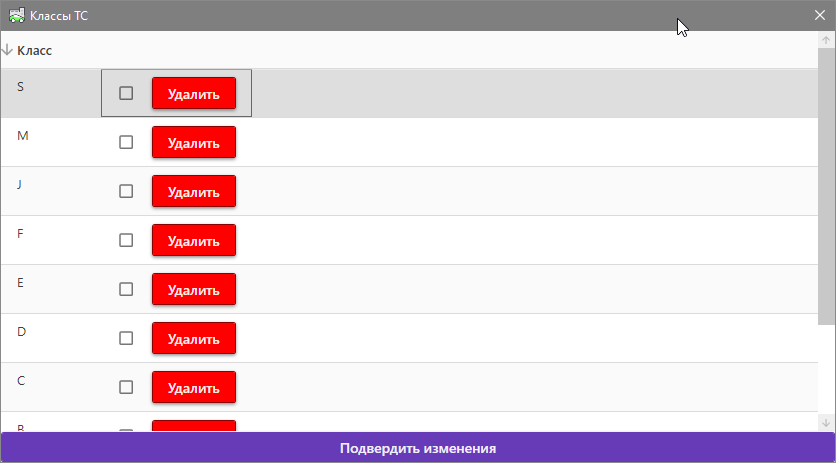


Рисунок 26 – Скриншот «Окно классов транспортных средств»

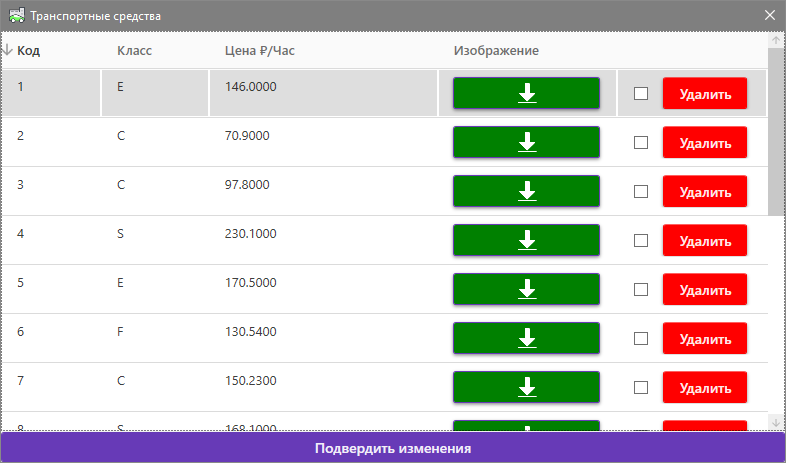


Рисунок 27 – Скриншот «Окно транспортных средств»

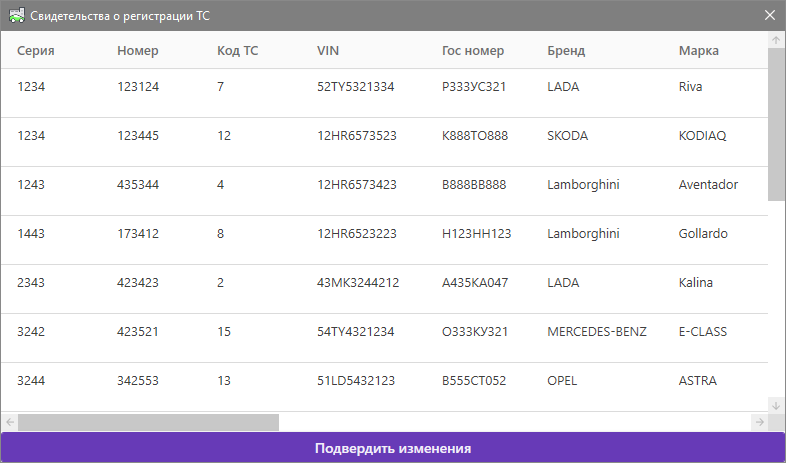


Рисунок 28 – Скриншот «Окно свидетельств о регистрации транспортных средств»

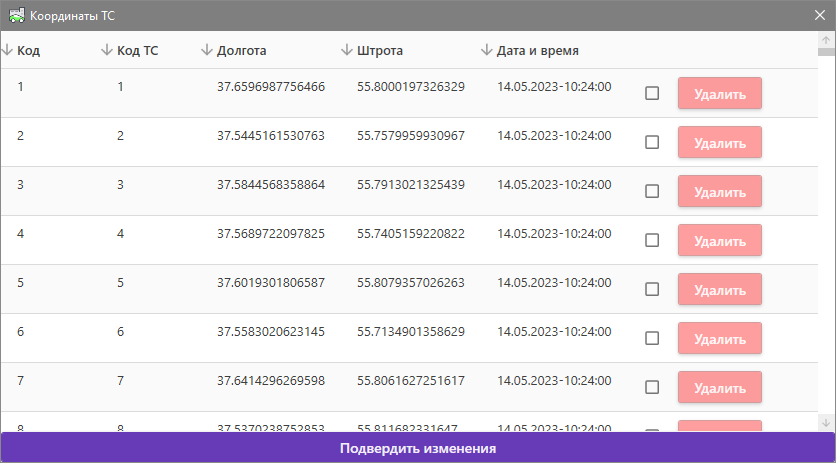


Рисунок 29 – Скриншот «Окно координат транспортных средств»

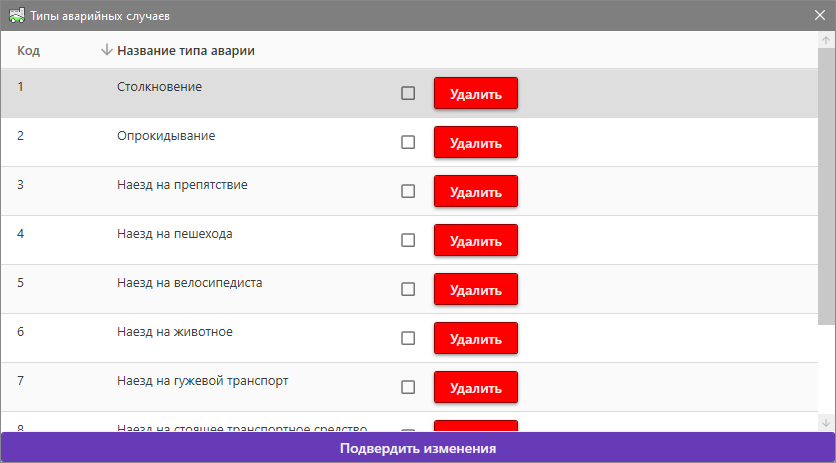


Рисунок 30 – Скриншот «Окно типов аварийных случаев»

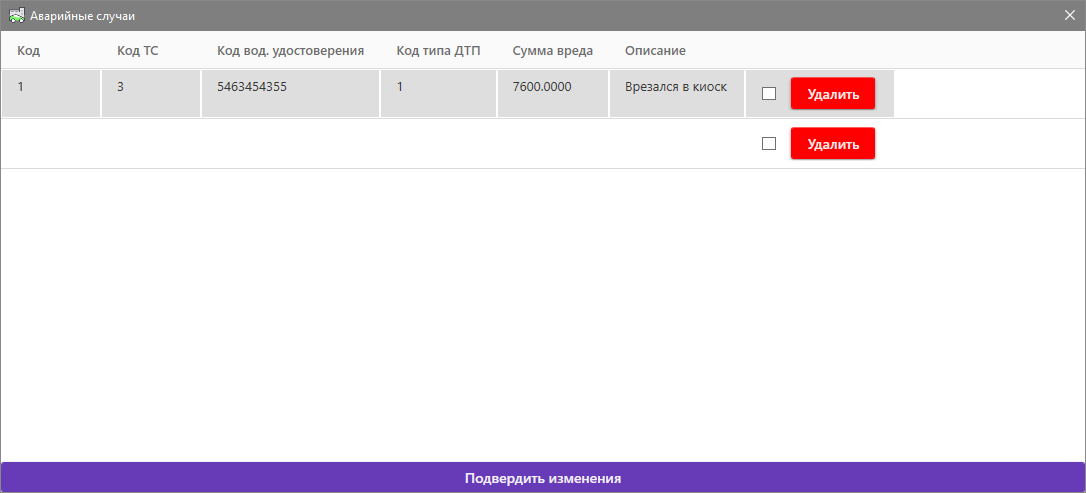


Рисунок 31 – Скриншот «Окно Аварийных случаев»

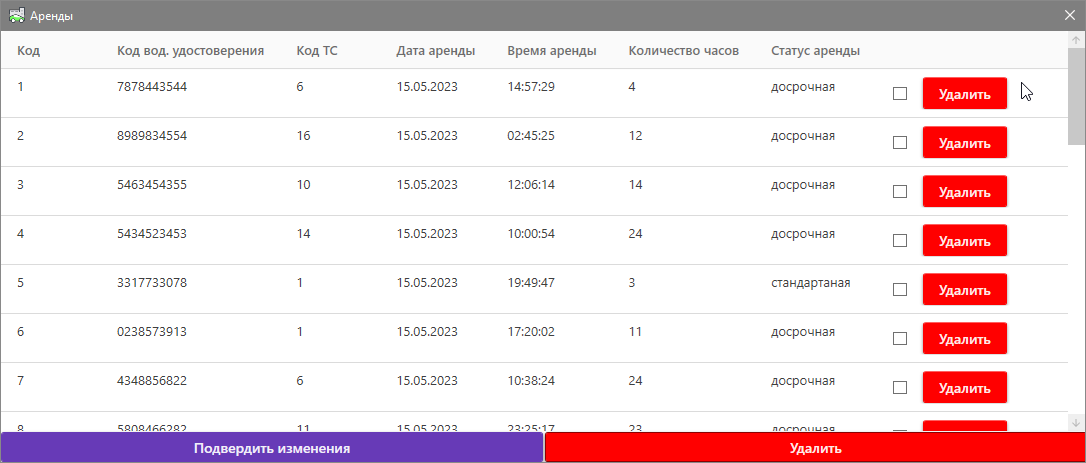


Рисунок 32 – Скриншот «Окно аренд»

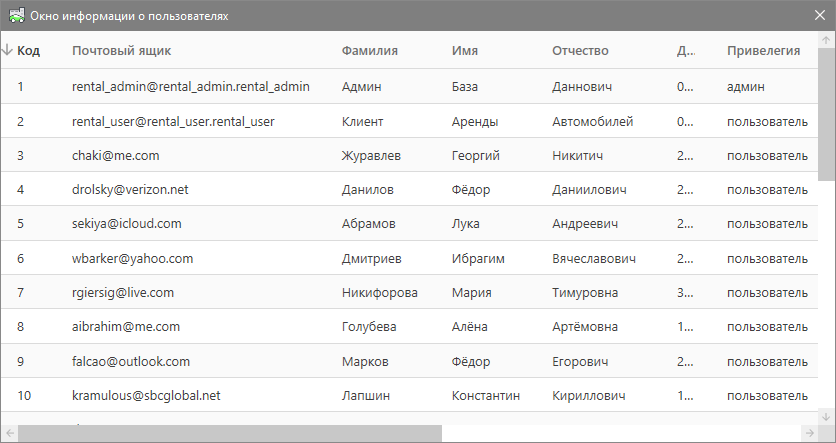


Рисунок 33 – Скриншот «Окно информации о пользователях»

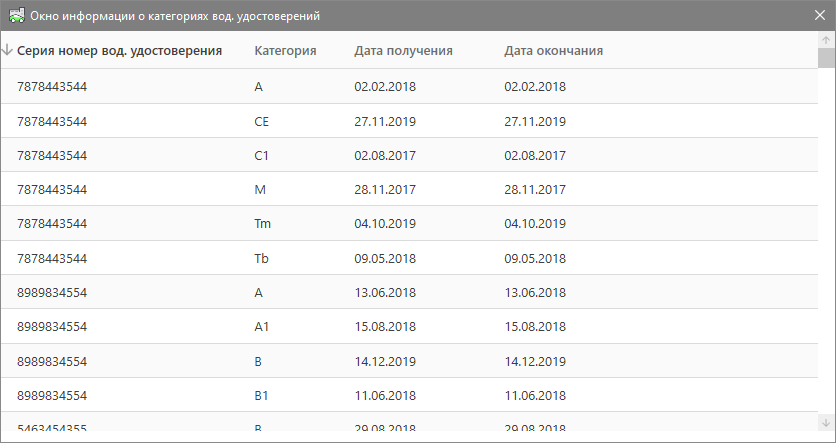


Рисунок 34 – Скриншот «Окно информации о категориях водительских удостоверений»

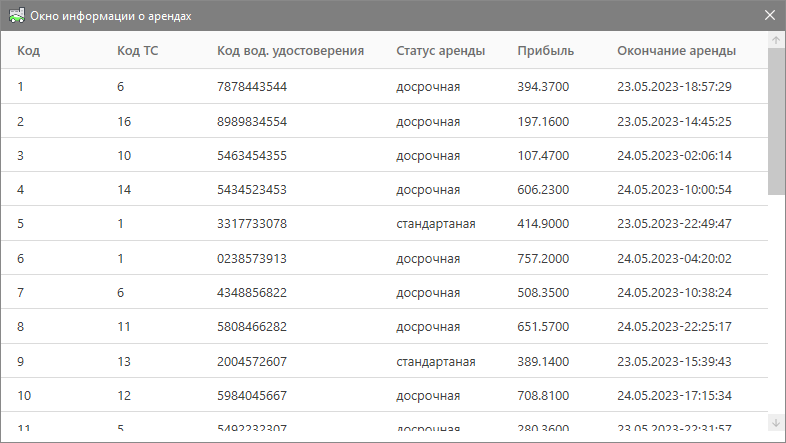


Рисунок 35 – Скриншот «Окно информации о арендах»



Рисунок 36 – Скриншот «Окно информации о ТС»

### Тестирование приложения

Таблица 10 – Метод проверки требований к приложению.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | Ожидаемый результат | Результат |
| 1 | Запуск приложения | Приложение подключается к серверу и базам данных | Успешно |
| 2 | Регистрация нового пользователя | Создается новая учетная запись в базе данных | Успешно |
| 3 | Авторизация пользователя | Зарегистрированный пользователь переходит на окно пользователя | Успешно |
| 4 | Авторизация администратора | Зарегистрированный администратор переходит на окно администратора | Успешно |
| 5 | Редактирование данных администратором | При выполнении одного из действий (Создания, удаления, изменения) происходит создание записи, удаление записи, и редактирование записи соответственно | Успешно |
| 6 | Выход | Переход в окно авторизации | Успешно |
| 8 | Добавление данных о водительском удостоверении | Добавление записи | Успешно |
| 8 | Выбор автомобиля | Переход в окно аренды | Успешно |
| 9 | Аренда автомобиля | Переход в окно поездки | Успешно |

### Защита информационной системы

Информационная система защищена от умышленного взлома, а также от ошибочных действий пользователя, администратора. При вводе некорректных данных в приложение выходит соответствующее предупреждающее сообщение об ошибке.

Информационная система при входе и регистрации проверяет корректность введённых данных и существование пользователя.

## Инструкция пользователю

Для пользования СУБД «Carsharing», скачать файлы из архива, запустить скрипт на создание SQL-базы данных находящийся в папке Carsharing/SQL под именем «CreateDataBase.sql», запустить скрипт на заполнение SQL-базы данных находящийся в папке Carsharing/SQL под именем «FillUp.sql». Чтобы начать пользоваться нужно пройти авторизацию или зарегистрироваться.

По умолчанию есть два пользователя

1. Клиент:

* Логин: rental\_user
* Пароль: rental\_user

1. Администратор:

* Логин: rental\_admin
* Пароль: rental\_admin

Для новых пользователей доступна функция регистрации.

Для зарегистрированных пользователей доступна функция авторизации

Для пользователей, являющихся администраторами доступны функции выбора таблиц для изменений и окно выбора автомобиля для аренды.

Для обычных пользователей доступна функция выбора автомобиля для аренды.

### Общие сведения об информационной системе

База данных

Сервер

Клиент

Рисунок 18 – Клиент-серверная модель

Вся информационная система завязана на клиент-серверной модели, где идет взаимодействие пользовательского приложения через сервер к базам данных. В качестве баз данных выступает MS SQL SERVER 2019.

В качестве клиента выступает приложение «Carsharing». Сервер расположен на компьютере пользователя.

### Требования к техническим средствам

В состав технических средств должен входить компьютер, включающий:

* оперативную память не менее 2Гб,
* процессор Intel начиная с Pentium II и старше или процессор AMD схожий по характеристикам,
* клавиатура и мышь или TouchPad,
* встроенный графический процессор или видеокарта,
* монитор,
* SSD или HDD диск.

### Требования к программным средствам

В состав программных средств должно входить:

* операционная система Windows 10 или версия выше,
* MS SQL SERVER 2019.
* Драйвер Microsoft .Net SDK 7 и выше

### Формы ввода

Все формы ввода, имеющиеся в приложении проверяют корректность введённых данных. В случае введения некорректных данных, приложение предупреждает и не даёт их ввести.

Формы ввода изображены на рисунках [17-36](#_Решение_главной_задачи).

### Прокат автомобиля

Данные об арендованном автомобиле автоматически заносятся в базу данных, и переносятся в окно поездки до завершения аренды.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была разработана информационная система, включающая в себя базу данных и приложение для проката автомобилей.

База данных полностью соответствует требованиям предметной области, все таблицы соответствуют правилам нормализации, что позволяет обеспечить целостность информации.

Средствами языка программирования C# было создано настольное приложение, функционал которого успешно решает все поставленные задачи курсового проекта. Исходя из этого можно сделать вывод, что задачи курсового проекта были выполнены.

В процессе выполнения курсового проекта был приобретен практический опыт создания сложных систем. Был пройден жизненный цикл программного обеспечения:

* анализ предметной области,
* проектирование,
* реализация,
* тестирование,
* эксплуатация и сопровождение.

Приложением могут пользоваться два типа пользователей ‑ администратор и клиент. Клиенту даны такие возможности как просмотр, выбор и аренда автомобилей. Возможности администратора включают в себя возможности клиента, а также добавление, удаление, редактирование записей.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Возможности языка C# и платформы .Net – электронный ресурс Джеффри Рихтер. CLR via C#: Программирование на платформе Microsoft .NET Framework 4.5 на языке C# 4-е издание URL: <https://777russia.ru/book/uploads/%D0%9F%D0%A0%D0%9E%D0%93%D0%A0%D0%90%D0%9C%D0%9C%D0%98%D0%A0%D0%9E%D0%92%D0%90%D0%9D%D0%98%D0%95/C%23/CLR%20via%20C%23%20%28.NET%204.5%29%2C%204th%20Edition%202012%20%28%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4%2C%20%D0%BD%D0%B0%20%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%BE%D0%BC%29.pdf>
2. Возможности Windows Presentation Foundation – электронный ресурс: Основы Windows Presentation Foundation. URL: <https://studylib.ru/doc/6224808/osnovy-windows-presentation-foundation>
3. Отладка кода C# - электронный ресурс: Учебник. Сведения об отладке кода C# с помощью Visual Studio URL: [https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-debugger?toc=%2Fvisualstudio%2  
   Fdebugger%2Ftoc.json&view=vs-2022](https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-debugger?toc=%2Fvisualstudio%2Fdebugger%2Ftoc.json&view=vs-2022)
4. Отладка кода XAML - электронный ресурс: Что такое горячая перезагрузка XAML для приложений WPF и UWP? (Visual Studio) URL: <https://learn.microsoft.com/ru-Ru/visualstudio/xaml-tools/xaml-hot-reload?view=vs-2019>
5. Подключение к базе данных MSSQL SERVER - электронный ресурс: Краткое руководство. Подключение к экземпляру SQL Server и выполнение запросов с помощью SQL Server Management Studio (SSMS) URL: <https://learn.microsoft.com/ru-ru/sql/ssms/quickstarts/ssms-connect-query-sql-server?view=aps-pdw-2016>

# ПРИЛОЖЕНИЕ А. SQL скрипты на создание и заполнение базы данных

GO

PRINT '==================================База Данных======================================='

USE master

IF EXISTS(SELECT \* FROM sys.databases WHERE LOWER([name]) = LOWER(N'VehicleRental'))

BEGIN

ALTER DATABASE VehicleRental

SET SINGLE\_USER WITH ROLLBACK IMMEDIATE

DROP DATABASE VehicleRental

PRINT 'Дропнул БД'

END ELSE PRINT 'Базы Данных не существует'

GO

IF EXISTS (SELECT \* FROM master.INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_TYPE = 'PROCEDURE' AND ROUTINE\_NAME = 'CreateDataBase')

BEGIN

DROP PROCEDURE CreateDataBase

PRINT 'Дропнул Процедуру создания БД'

END ELSE PRINT 'Error: Не удалось дропнуть процедуру создания БД'

GO

CREATE PROCEDURE CreateDataBase (@DBName NVARCHAR(max), @DBPath NVARCHAR(max))

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX) = 'CREATE DATABASE ' + @DBName + ' ON PRIMARY ' +

'( NAME = ' + @DBName + ', ' +

' FILENAME = ''' + @DBPath + '\' + @DBName + '.mdf'' , ' +

' SIZE = 5MB , MAXSIZE = UNLIMITED, FILEGROWTH = 5%) ' +

' LOG ON ' +

'( NAME = ' + @DBName + '\_Log, ' +

' FILENAME = ''' + @DBPath + '\' + @DBName + '\_Log.ldf'', ' +

' SIZE = 1MB , MAXSIZE = UNLIMITED, FILEGROWTH = 1%)'

EXEC sp\_executesql @SQL

END

PRINT 'Создал Процедуру создания БД'

/\*Создание БД\*/

GO

IF NOT EXISTS(SELECT \* FROM sys.databases WHERE LOWER([name]) = LOWER(N'VehicleRental'))

BEGIN

EXEC CreateDataBase @DBName = 'VehicleRental', @DBPath = 'C:\Users\Public'

PRINT 'Создал БД'

END

/\*Создание таблиц\*/

GO

PRINT '==================================Таблицы======================================='

USE VehicleRental

CREATE TABLE Vehicle\_Rental\_logs

(

ID\_Log INT IDENTITY(1,1) NOT NULL,

LogString NVARCHAR(MAX) NOT NULL

PRIMARY KEY (ID\_Log)

)

CREATE TABLE Rental\_Users

(

ID\_User INT IDENTITY(1,1) NOT NULL,

UserLogin NVARCHAR(120) UNIQUE NOT NULL,

UserEMail NVARCHAR(120) UNIQUE NOT NULL,

UserPassword NVARCHAR(120) NOT NULL,

UserSurname NVARCHAR(120) NOT NULL,

UserName NVARCHAR(120) NOT NULL,

UserMiddleName NVARCHAR(120) NOT NULL,

UserBirthDay DATE NOT NULL,

PRIMARY KEY (ID\_User)

)

PRINT 'Создал Таблицу Rental\_Users'

CREATE TABLE Rental\_Admins

(

ID\_Admin INT IDENTITY(1,1) NOT NULL,

ID\_User INT NOT NULL

)

PRINT 'Создал Таблицу Rental\_Admins'

CREATE TABLE DriversLicences

(

ID\_DriverLicence CHAR(10) PRIMARY KEY NOT NULL

CHECK (

ID\_DriverLicence LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

),

ReceiptDate DATE NOT NULL,

ID\_User INT NOT NULL

)

PRINT 'Создал Таблицу DriversLicences'

CREATE TABLE Categories

(

ID\_Category INT IDENTITY(1,1) NOT NULL,

ID\_DriverLicence CHAR(10) NOT NULL

CHECK (

ID\_DriverLicence LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

),

Category CHAR(3) NOT NULL,

ReceiptDate DATE NOT NULL,

EndDate DATE NOT NULL

PRIMARY KEY (ID\_Category)

)

PRINT 'Создал Таблицу Categories'

CREATE TABLE Classes

(

Class CHAR(20) PRIMARY KEY NOT NULL

)

PRINT 'Создал Таблицу Classes'

CREATE TABLE Vehicles

(

ID\_Vehicle INT IDENTITY(1,1) NOT NULL,

PricePerHour MONEY NOT NULL,

Class CHAR(20) NOT NULL,

CarPicture VARBINARY(MAX)

PRIMARY KEY (ID\_Vehicle)

)

PRINT 'Создал Таблицу Vehicles'

CREATE TABLE VehicleCoordinates

(

ID\_Coordinates INT IDENTITY(1,1) NOT NULL,

ID\_Vehicle INT NOT NULL,

Longitude FLOAT NOT NULL,

Latitude FLOAT NOT NULL,

StayDateTime DATETIME NOT NULL

PRIMARY KEY (ID\_Coordinates)

)

PRINT 'Создал Таблицу VehicleCoordinates'

CREATE TABLE Rentals

(

ID\_Rental INT IDENTITY(1,1) NOT NULL,

ID\_DriverLicence CHAR(10) NOT NULL

CHECK (

ID\_DriverLicence LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

),

ID\_Vehicle INT NULL,

StartDate DATE NOT NULL,

RentalTime TIME NOT NULL,

CountOfHours INT NOT NULL,

TotalPrice MONEY NOT NULL,

RentalStatus CHAR(12) DEFAULT 'стандартная' NOT NULL

PRIMARY KEY (ID\_Rental)

)

PRINT 'Создал Таблицу Rentals'

CREATE TABLE TrafficAccidentTypes

(

ID\_TrafficAccidentType INT IDENTITY(1,1) NOT NULL,

TrafficAccidentTypeName NVARCHAR(60) NOT NULL

PRIMARY KEY (ID\_TrafficAccidentType)

)

PRINT 'Создал Таблицу TrafficAccidentTypes'

CREATE TABLE TrafficAccidents

(

ID\_TrafficAccident INT IDENTITY(1,1) NOT NULL,

ID\_Vehicle INT NOT NULL,

ID\_DriverLicence CHAR(10) NOT NULL

CHECK (

ID\_DriverLicence LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

),

ID\_TrafficAccidentType INT NOT NULL,

Damage MONEY NOT NULL,

TrafficAccidentDescription NVARCHAR(MAX) NOT NULL

PRIMARY KEY (ID\_TrafficAccident)

)

PRINT 'Создал Таблицу TrafficAccidents'

CREATE TABLE VehicleRegistrCertificates

(

CertificateSeries INT NOT NULL,

CertificateNumber INT NOT NULL,

ID\_Vehicle INT NOT NULL,

VIN CHAR(14) NOT NULL,

RegistrationNum CHAR(10) UNIQUE NOT NULL

CHECK (

RegistrationNum

LIKE '[А-Я][0-9][0-9][0-9][А-Я][А-Я][0-9][0-9][0-9]'

),

Brand NVARCHAR(100) NOT NULL,

Mark NVARCHAR(100) NOT NULL,

Vehicle\_Type VARCHAR(100) NOT NULL,

Vehicle\_Category VARCHAR(6) NOT NULL

CHECK (

Vehicle\_Category = 'A'

OR Vehicle\_Category = 'B'

OR Vehicle\_Category = 'C'

OR Vehicle\_Category = 'D'

OR Vehicle\_Category = 'прицеп'

),

ManufactureYear INT NOT NULL,

Chassis VARCHAR(100) NOT NULL,

BodyNo VARCHAR(40) NOT NULL,

Color VARCHAR(120) NOT NULL,

EngineHP\_kW FLOAT NOT NULL,

EcologicalClass CHAR(100) NOT NULL,

PermissibleMaximumWeightKg INTEGER NOT NULL,

WeightWithoutLoadKg INTEGER NOT NULL,

PRIMARY KEY (CertificateSeries, CertificateNumber)

)

PRINT 'Создал Таблицу VehicleRegistrCertificates'

/\*Связи\*/

GO

PRINT '==================================Связи======================================='

USE VehicleRental

ALTER TABLE Rental\_Admins

ADD CONSTRAINT FK\_Rental\_Admins\_Rental\_Users

FOREIGN KEY (ID\_User) REFERENCES Rental\_Users (ID\_User)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_Rental\_Admins\_Rental\_Users'

GO

USE VehicleRental

ALTER TABLE DriversLicences

ADD CONSTRAINT FK\_DriversLicenses\_Rental\_Users

FOREIGN KEY (ID\_User) REFERENCES Rental\_Users (ID\_User)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_DriversLicenses\_Rental\_Users'

GO

USE VehicleRental

ALTER TABLE Categories

ADD CONSTRAINT FK\_Categories\_DriversLicenses

FOREIGN KEY (ID\_DriverLicence) REFERENCES DriversLicences (ID\_DriverLicence)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_Categories\_DriversLicenses'

GO

USE VehicleRental

ALTER TABLE Rentals

ADD CONSTRAINT FK\_Rentals\_DriversLicences

FOREIGN KEY (ID\_DriverLicence) REFERENCES DriversLicences (ID\_DriverLicence)

ON DELETE SET DEFAULT

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_Rentals\_DriversLicences'

GO

USE VehicleRental

ALTER TABLE Rentals

ADD CONSTRAINT FK\_Rentals\_Vehicles

FOREIGN KEY (ID\_Vehicle) REFERENCES Vehicles (ID\_Vehicle)

ON DELETE SET DEFAULT

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_Rentals\_Vehicles'

GO

USE VehicleRental

ALTER TABLE VehicleCoordinates

ADD CONSTRAINT FK\_VehicleCoordinates\_Vehicles

FOREIGN KEY (ID\_Vehicle) REFERENCES Vehicles (ID\_Vehicle)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_VehicleCoordinates\_Vehicles'

GO

USE VehicleRental

ALTER TABLE TrafficAccidents

ADD CONSTRAINT FK\_TrafficAccidents\_Vehicles

FOREIGN KEY (ID\_Vehicle) REFERENCES Vehicles (ID\_Vehicle)

ON DELETE SET DEFAULT

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_TrafficAccidents\_Vehicles'

GO

USE VehicleRental

ALTER TABLE TrafficAccidents

ADD CONSTRAINT FK\_TrafficAccidents\_TrafficAccidentTypes

FOREIGN KEY (ID\_TrafficAccidentType) REFERENCES TrafficAccidentTypes (ID\_TrafficAccidentType)

ON DELETE SET DEFAULT

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_TrafficAccidents\_TrafficAccidentTypes'

GO

USE VehicleRental

ALTER TABLE TrafficAccidents

ADD CONSTRAINT FK\_TrafficAccidents\_DriversLincences

FOREIGN KEY (ID\_DriverLicence) REFERENCES DriversLicences (ID\_DriverLicence)

ON DELETE SET DEFAULT

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_TrafficAccidents\_DriversLincences'

GO

USE VehicleRental

ALTER TABLE VehicleRegistrCertificates

ADD CONSTRAINT FK\_VehicleRegistrCertificates\_Vehicles

FOREIGN KEY (ID\_Vehicle)

REFERENCES Vehicles (ID\_Vehicle)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_VehicleRegistrCertificates\_Vehicles'

GO

USE VehicleRental

ALTER TABLE Vehicles

ADD CONSTRAINT FK\_Vehicles\_Classes

FOREIGN KEY (Class) REFERENCES Classes (Class)

ON DELETE CASCADE

ON UPDATE CASCADE

PRINT 'Создал Ключ FK\_Vehicles\_Classes'

/\*Роли\*/

GO

PRINT '==================================Роли======================================='

USE VehicleRental

/\*Хранимые процедуры\*/

GO

PRINT '==================================Хранимые процедуры======================================='

GO

CREATE PROCEDURE REG\_USER

@UserLogin NVARCHAR(MAX),

@UserEmail NVARCHAR(MAX),

@UserPassword NVARCHAR(MAX),

@UserSurname NVARCHAR(MAX),

@UserName NVARCHAR(MAX),

@UserMiddleName NVARCHAR(MAX),

@UserBirthDayDate NVARCHAR(10)

AS

BEGIN TRY

BEGIN TRANSACTION Registration

DECLARE @SqlCommand NVARCHAR(MAX) =

'USE VehicleRental '+

'INSERT INTO Rental\_Users '+

'(UserLogin, UserEMail, UserPassword, UserSurname, UserName, UserMiddleName, UserBirthDay) VALUES ('+

''''+@UserLogin+''', '''+@UserEmail+''', '''+@UserPassword+''', '+''''+@UserSurname+''', '''+@UserName+''', '''+@UserMiddleName+''', '''+@UserBirthDayDate+''')'

EXEC (@SqlCommand)

COMMIT TRANSACTION Registration

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION Registration

END CATCH

GO

PRINT 'Создал Хранимую процедуру REG\_USER'

GO

CREATE FUNCTION GetDriverLicenceByUserID(

@User\_ID INT

)

RETURNS TABLE

AS

RETURN SELECT TOP(1) \*

FROM DriversLicences

WHERE ID\_User = @User\_ID

ORDER BY ReceiptDate DESC

GO

PRINT 'Создал Хранимую процедуру GetDriverLicenceByUserID'

GO

CREATE PROCEDURE AddDriverLicenceToUser

(

@User\_ID INT,

@DriverLicence CHAR(10),

@ReceiptDate DATE

)

AS

BEGIN TRY

BEGIN TRANSACTION AddDriverLicence

INSERT DriversLicences VALUES

(@DriverLicence, @ReceiptDate, @User\_ID)

COMMIT TRANSACTION AddDriverLicence

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION AddDriverLicence

END CATCH

GO

PRINT 'Создал Хранимую процедуру AddDriverLicenceToUser'

GO

CREATE PROCEDURE AddCategoryToDriverLicence

(

@DriverLicence\_ID CHAR(10),

@Category CHAR(3),

@ReceiptDate DATE,

@EndDate DATE

)

AS

BEGIN TRY

BEGIN TRANSACTION AddCategory

INSERT Categories VALUES

(@DriverLicence\_ID, @Category, @ReceiptDate, @EndDate)

COMMIT TRANSACTION AddCategory

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION AddCategory

END CATCH

GO

PRINT 'Создал Хранимую процедуру AddCategoryToDriverLicence'

GO

CREATE PROCEDURE Rent(

@DriverLicence CHAR(10),

@ID\_Vehicle INT,

@RentalTime TIME,

@CountOfHours INT,

@TotalPrice MONEY

)

AS

BEGIN TRY

BEGIN TRANSACTION RentVehicle

IF NOT ((SELECT [dbo].GetVehicleStatus(@ID\_Vehicle)) = 'доступен')

ROLLBACK TRANSACTION RentVehicle

INSERT Rentals VALUES

(@DriverLicence, @ID\_Vehicle, GETDATE(), @RentalTime, @CountOfHours, @TotalPrice, 'стандартная')

COMMIT TRANSACTION RentVehicle

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION RentVehicle

END CATCH

GO

PRINT 'Создал Хранимую процедуру Rent'

GO

PRINT '==================================Функции======================================='

GO

CREATE FUNCTION CheckExistingUser

(

@login NVARCHAR(MAX),

@Password NVARCHAR(MAX)

)

RETURNS INT

AS

BEGIN

DECLARE @ID INT = -1

(SELECT @ID = ID\_User FROM Rental\_Users WHERE Rental\_Users.UserLogin = @login AND UserPassword = @Password)

RETURN @ID

END

GO

PRINT 'Создал Функции CheckExistingUser'

GO

CREATE FUNCTION DBStatus(

@ID\_User INT

)

RETURNS char(24)

BEGIN

DECLARE @answ CHAR(24);

IF (EXISTS (SELECT \* FROM Rental\_Admins WHERE ID\_User = @ID\_User))

SET @answ = 'админ'

ELSE IF (EXISTS (SELECT \* FROM Rental\_Users WHERE ID\_User = @ID\_User))

SET @answ = 'пользователь'

ELSE SET @answ = 'пользователь отсутствует'

RETURN @answ

END

GO

PRINT 'Создал Функции DBStatus'

GO

CREATE FUNCTION GetCoordinatesFunc(

@Vehicle\_ID INT

)

RETURNS table

AS

RETURN(

SELECT TOP(1) \*

FROM VehicleCoordinates

WHERE VehicleCoordinates.ID\_Vehicle = @Vehicle\_ID

ORDER BY StayDateTime DESC

)

GO

PRINT 'Создал Функцию GetCoordinatesFunc'

GO

CREATE FUNCTION GetVehicleStatus(

@Vehicle\_ID INT

)

RETURNS char(11)

AS

BEGIN

DECLARE @ret char(11) = 'отсутствует'

IF EXISTS(SELECT \* FROM Vehicles WHERE ID\_Vehicle = @Vehicle\_ID)

IF NOT EXISTS (SELECT \*,

DATEADD(HOUR, ABS(DATEDIFF(HOUR,RentalTime,0))+ABS(CountOfHours),Convert(datetime, StartDate, 102)) as rt

FROM Rentals

WHERE DATEADD(HOUR, ABS(DATEDIFF(HOUR,RentalTime,0))+ABS(CountOfHours),Convert(datetime, StartDate, 0))

> GETDATE()

AND ID\_Vehicle = @Vehicle\_ID

AND YEAR(DATEADD(HOUR, ABS(DATEDIFF(HOUR,RentalTime,0))+ABS(CountOfHours),Convert(datetime, StartDate, 0)))

= YEAR(GETDATE()) and RentalStatus = 'стандартная')

SET @ret = 'доступен'

ELSE SET @ret = 'занят'

RETURN @ret

END

GO

PRINT 'Создал Функцию GetVehicleStatus'

GO

CREATE FUNCTION GetCoordinates(

@Vehicle\_ID INT

)

RETURNS table

AS

RETURN

SELECT TOP(1) ID\_Vehicle, Latitude, Longitude

FROM VehicleCoordinates

WHERE ID\_Vehicle = @Vehicle\_ID

ORDER BY StayDateTime DESC

GO

PRINT 'Создал Функцию GetCoordinates'

USE VehicleRental

GO

CREATE FUNCTION GetUserDriverLicences

(

@ID\_USER INT

)

RETURNS TABLE

AS

RETURN (SELECT \* FROM DriversLicences WHERE DriversLicences.ID\_User = @ID\_USER)

GO

PRINT 'Создал Функцию GetUserDriverLicences'

USE VehicleRental

GO

CREATE FUNCTION UserRentalCount

(

@ID\_USER INT

)

RETURNS INT

AS

BEGIN

DECLARE @Count INT = 0

SELECT @Count = COUNT(\*) FROM Rentals

INNER JOIN [dbo].GetUserDriverLicences(@ID\_USER) as LICENCES

ON LICENCES.ID\_DriverLicence = Rentals.ID\_DriverLicence

RETURN @Count

END

GO

PRINT 'Создал Функцию UserRentalCount'

USE VehicleRental

GO

CREATE FUNCTION CountUserAccidents

(

@ID\_User INT

)

RETURNS INT

AS

BEGIN

DECLARE @Count INT

SELECT @Count=COUNT(\*) FROM TrafficAccidents

INNER JOIN [dbo].GetUserDriverLicences(@ID\_User) as LICENCES

ON LICENCES.ID\_DriverLicence = TrafficAccidents.ID\_DriverLicence

RETURN @Count

END

GO

PRINT 'Создал Функцию CountUserAccidents'

GO

CREATE FUNCTION VehicleAccidents

(

@ID\_Vehicle INT

)

RETURNS TABLE

AS

RETURN SELECT \* FROM TrafficAccidents WHERE ID\_Vehicle = @ID\_Vehicle

GO

PRINT 'Создал Функцию VehicleAccidents'

GO

CREATE FUNCTION CountVehicleAccidents

(

@ID\_Vehicle INT

)

RETURNS INT

AS

BEGIN

DECLARE @Count INT

SELECT @Count = COUNT(\*) FROM TrafficAccidents WHERE ID\_Vehicle = @ID\_Vehicle

RETURN @Count

END

GO

PRINT 'Создал Функцию CountVehicleAccidents'

GO

CREATE FUNCTION TotalVehicleDamage

(

@ID\_Vehicle INT

)

RETURNS MONEY

AS

BEGIN

DECLARE @TotalSum MONEY

SELECT @TotalSum = SUM(Damage) FROM [dbo].VehicleAccidents(@ID\_Vehicle)

RETURN @TotalSum

END

GO

PRINT 'Создал Функцию TotalVehicleDamage'

GO

CREATE FUNCTION GetUserStatus

(

@ID\_DriverLicence CHAR(10)

)

RETURNS CHAR(9) -- 'арендует' 'нет'

AS

BEGIN

DECLARE @ret char(9) = 'нет'

if EXISTS( SELECT ID\_DriverLicence

FROM Rentals

WHERE Rentals.ID\_DriverLicence = @ID\_DriverLicence

AND DATEADD(HOUR,Rentals.CountOfHours+DATEPART(HOUR,CONVERT(DATETIME,Rentals.RentalTime)),CONVERT(DATETIME,Rentals.StartDate)) < GETDATE())

SET @ret = 'арендует'

RETURN @ret

END

GO

PRINT 'Создал Функцию GetUserStatus'

GO

PRINT '==================================Представления======================================='

USE VehicleRental

GO

CREATE VIEW RentalsINFO

AS

SELECT

ID\_Rental,

ID\_Vehicle,

ID\_DriverLicence,

RentalStatus,

TotalPrice,

DATEADD(SECOND,DATEPART(SECOND,Rentals.RentalTime),

DATEADD(MINUTE,DATEPART(MINUTE,Rentals.RentalTime),

DATEADD(HOUR,Rentals.CountOfHours+DATEPART(HOUR,Rentals.RentalTime),

CONVERT(DATETIME,StartDate)))) as EndTime

FROM Rentals

GO

USE VehicleRental

GO

CREATE FUNCTION TripStatus

(

@DriverLicence CHAR(10)

)

RETURNS CHAR(9)

AS

BEGIN

DECLARE @Ret CHAR(9)

IF EXISTS(SELECT TOP (1) \*

FROM RentalsINFO

WHERE RentalsINFO.ID\_DriverLicence = @DriverLicence

AND EndTime > GETDATE() AND LOWER(RentalStatus) = 'стандартная')

SET @Ret = 'в поездке'

ELSE

SET @Ret = NULL

RETURN @Ret

END

GO

USE VehicleRental

GO

CREATE VIEW VehiclesINFO

AS

SELECT DISTINCT

Vehicles.ID\_Vehicle,

VehicleRegistrCertificates.Brand,

VehicleRegistrCertificates.Mark,

VehicleRegistrCertificates.Color,

Vehicles.Class,

VehicleRegistrCertificates.Vehicle\_Category,

Vehicles.PricePerHour,

Vehicles.CarPicture,

(SELECT TOP(1) Latitude

FROM VehicleCoordinates

WHERE VehicleCoordinates.ID\_Vehicle = Vehicles.ID\_Vehicle

ORDER BY StayDateTime DESC) as Lat,

(SELECT TOP(1) Longitude

FROM VehicleCoordinates

WHERE VehicleCoordinates.ID\_Vehicle = Vehicles.ID\_Vehicle

ORDER BY StayDateTime DESC) as Lng,

[dbo].GetVehicleStatus(Vehicles.ID\_Vehicle) as AccessStatus,

[dbo].TotalVehicleDamage(Vehicles.ID\_Vehicle) as DamageCost

FROM

Vehicles INNER JOIN VehicleRegistrCertificates

ON Vehicles.ID\_Vehicle = VehicleRegistrCertificates.ID\_Vehicle

GO

PRINT 'Создал представление VehiclesINFO'

GO

USE VehicleRental

GO

CREATE VIEW UsersINFO

AS

SELECT

ID\_User,

UserEMail,

UserSurname,

UserName,

UserMiddleName,

UserBirthDay,

[dbo].DBStatus(Rental\_Users.ID\_User) as Previlege,

[dbo].UserRentalCount(Rental\_Users.ID\_User) as RentalsCount,

[dbo].CountUserAccidents(Rental\_Users.ID\_User) as AccidentsCount,

(SELECT ID\_DriverLicence FROM [dbo].GetDriverLicenceByUserID(Rental\_Users.ID\_User)) as ID\_DriverLicence, --??

(SELECT ReceiptDate FROM [dbo].GetDriverLicenceByUserID(Rental\_Users.ID\_User)) as ReceiptDate, --??,

[dbo].TripStatus((SELECT ID\_DriverLicence FROM [dbo].GetDriverLicenceByUserID(Rental\_Users.ID\_User))) as RentStatus

FROM Rental\_Users

GO

PRINT 'Создал представление UsersINFO'

GO

CREATE VIEW DriversLicencesCategoriesINFO

AS

SELECT ID\_DriverLicence,Category, ReceiptDate, EndDate

FROM Categories

WHERE Categories.EndDate >= GETDATE()

GO

PRINT 'Создал представление DriversLicencesCategoriesINFO'

GO

PRINT '==================================Функции 2====================================='

GO

CREATE FUNCTION VehiclesWithStatus

(

@Status char(8)

)

RETURNS table

AS

RETURN (

SELECT \* FROM VehiclesINFO

WHERE LOWER(AccessStatus) = LOWER(@Status))

GO

PRINT 'Создал Функцию VehiclesWithStatus'

GO

CREATE FUNCTION GetDriverLicenceCategories

(

@ID\_DriverLicence CHAR(10)

)

RETURNS TABLE

AS

RETURN

(SELECT

TRIM(LOWER(Category)) as Category,

ReceiptDate,

EndDate

FROM

DriversLicencesCategoriesINFO

WHERE

ID\_DriverLicence = @ID\_DriverLicence)

GO

PRINT 'Создал Функцию GetDriverLicenceCategories'

GO

PRINT '==================================Пользователи======================================='

GO

PRINT '==================================Встроенные пользователи (приложние)======================================='

USE VehicleRental

CREATE LOGIN USERHANDLER WITH PASSWORD = 'USERHANDLER'

CREATE USER DB\_USER\_USERHANDLER FOR LOGIN USERHANDLER

GRANT INSERT, SELECT ON Rental\_Users to DB\_USER\_USERHANDLER

GRANT EXECUTE ON CheckExistingUser to DB\_USER\_USERHANDLER

GRANT SELECT ON UsersINFO to DB\_USER\_USERHANDLER

GRANT SELECT ON Rental\_Admins to DB\_USER\_USERHANDLER

GRANT SELECT ON GetDriverLicenceByUserID to DB\_USER\_USERHANDLER

GRANT EXEC ON AddDriverLicenceToUser to DB\_USER\_USERHANDLER

GRANT EXEC ON AddCategoryToDriverLicence to DB\_USER\_USERHANDLER

GRANT EXEC ON REG\_USER TO DB\_USER\_USERHANDLER

GRANT EXEC ON Rent TO DB\_USER\_USERHANDLER

--GRANT EXEC ON CHECK\_USER TO DB\_USER\_USERHANDLER

CREATE LOGIN CARHANDLER WITH PASSWORD = 'CARHANDLER'

CREATE USER DB\_USER\_CARHANDLER FOR LOGIN CARHANDLER

GRANT INSERT, SELECT, UPDATE ON Vehicles TO DB\_USER\_CARHANDLER

GRANT INSERT, SELECT, UPDATE ON VehicleCoordinates TO DB\_USER\_CARHANDLER

GRANT INSERT, SELECT, UPDATE ON Classes TO DB\_USER\_CARHANDLER

--GRANT INSERT, SELECT, UPDATE, ON VehiclesPassports TO DB\_USER\_CARHANDLER

GRANT INSERT, SELECT, UPDATE ON VehicleRegistrCertificates TO DB\_USER\_CARHANDLER

GRANT SELECT ON VehiclesINFO TO DB\_USER\_CARHANDLER

GRANT SELECT ON VehiclesWithStatus TO DB\_USER\_CARHANDLER

GRANT SELECT ON RentalsINFO TO DB\_USER\_CARHANDLER

GRANT SELECT, UPDATE ON Rentals TO DB\_USER\_CARHANDLER

GRANT INSERT, SELECT ON TrafficAccidents TO DB\_USER\_CARHANDLER

GRANT INSERT, SELECT ON TrafficAccidentTypes TO DB\_USER\_CARHANDLER

CREATE LOGIN DLHANDLER WITH PASSWORD = 'DLHANDLER'

CREATE USER DB\_USER\_DLHANDLER FOR LOGIN DLHANDLER

GRANT INSERT, SELECT, UPDATE ON DriversLicences TO DB\_USER\_DLHANDLER

GRANT SELECT ON GetDriverLicenceCategories TO DB\_USER\_DLHANDLER

CREATE LOGIN DATABASEADMIN WITH PASSWORD = 'MODERATEDATABASE'

CREATE USER DB\_ADMIN\_HANDLER FOR LOGIN DATABASEADMIN

GRANT INSERT, SELECT, UPDATE, DELETE ON Vehicles TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON VehicleCoordinates TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON VehicleRegistrCertificates TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON Categories TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON Classes TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON DriversLicences TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON Rental\_Users to DB\_ADMIN\_HANDLER

GRANT SELECT ON Rental\_Admins to DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON Rentals TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON TrafficAccidents TO DB\_ADMIN\_HANDLER

GRANT INSERT, SELECT, UPDATE, DELETE ON TrafficAccidentTypes TO DB\_ADMIN\_HANDLER

GRANT SELECT ON UsersINFO TO DB\_ADMIN\_HANDLER

GRANT SELECT ON RentalsINFO TO DB\_ADMIN\_HANDLER

GRANT SELECT ON DriversLicencesCategoriesINFO TO DB\_ADMIN\_HANDLER

GRANT SELECT ON VehiclesINFO TO DB\_ADMIN\_HANDLER

GRANT EXEC ON REG\_USER TO DB\_USER\_USERHANDLER

CREATE LOGIN DBLOGGER WITH PASSWORD = 'LOGDATABASE'

CREATE USER LOGGER FOR LOGIN DBLOGGER

GRANT SELECT, INSERT ON Vehicle\_Rental\_Logs TO LOGGER

USE VehicleRental

GO

USE VehicleRental

INSERT Rental\_Users VALUES

('eqYY5T3L49ZRJbW/V9rosA==', 'rental\_admin@rental\_admin.rental\_admin', 'eqYY5T3L49ZRJbW/V9rosA==', 'AdminSurname', 'AdminName', 'AdminMiddlename', '2023-04-01'),

('/dnKYZHqiXNRZX3gcUdwwg==', 'rental\_user@rental\_user.rental\_user', '/dnKYZHqiXNRZX3gcUdwwg==', 'rental\_user', 'rental\_user', 'rental\_user', '2023-04-06'),

('jp6uPv45moXS5qVs0UkNwA==', 'chaki@me.com', '/gzw+eMOrPavQsQwRWU5dw==', 'Журавлев', 'Георгий', 'Никитич', '27.06.1976'),

('OUE1lvANhEI=', 'drolsky@verizon.net', 'yWOdkJjT2OhE+GyLsNBQhg==', 'Данилов', 'Фёдор', 'Даниилович', '28.03.1995'),

('/niF9okVx9M=', 'sekiya@icloud.com', '5rhTvoSsG/GgqucpRZnPtQ==', 'Абрамов', 'Лука', 'Андреевич', '20.04.2004'),

('F0CnAOR1XnjlDV1NlgToQg==', 'wbarker@yahoo.com', 'WjJce2gIy+eqxPZ/igAynQ==', 'Дмитриев', 'Ибрагим', 'Вячеславович', '20.03.1987'),

('TgYb56aMtz3x4BI0aDEBWQ==', 'rgiersig@live.com', 'q7s22MRGK9Q7FIqJDxdTJg==', 'Никифорова', 'Мария', 'Тимуровна', '31.07.1980'),

('wco4QPqNMnj5QduIrnDeeA==', 'aibrahim@me.com', 'T20KYiCgB4j+yhvqLzZ58g==', 'Голубева', 'Алёна', 'Артёмовна', '11.01.1973'),

('cgS+lczeUTw=', 'falcao@outlook.com', 'RgjqSmu+4y1lXdv9u1U4PA==', 'Марков', 'Фёдор', 'Егорович', '25.09.1978'),

('d+qxcyHz+Ng=', 'kramulous@sbcglobal.net', 'Y1CnhCWNVY0GpdbuBkSi8A==', 'Лапшин', 'Константин', 'Кириллович', '11.03.1996'),

('LuDnU2xPKnRsK1gnXUa+lg==', 'draper@comcast.net', 'Sbt/J/FbmuWmvU1fAcBirw==', 'Королева', 'Алиса', 'Фёдоровна', '27.12.2001'),

('p6zKL1c1vck=', 'multiplx@icloud.com', 'tRAOb7dBifcO9mNoQ7M3tg==', 'Ананьев', 'Марк', 'Романович', '05.03.1984'),

('72hLvgWDLRQ=', 'zeitlin@aol.com', 'BgPFPhbG1jZL9W4my2wj7g==', 'Седов', 'Данил', 'Максимович', '29.04.2004'),

('Y3Q3p+gxPLg=', 'caidaperl@aol.com', '9Lb+O8n8XDnHcyxMKqzpwA==', 'Иванов', 'Роберт', 'Фёдорович', '14.04.1979'),

('4xgobG9ZXYg=', 'miturria@aol.com', 'H0XUoAx8KKu1hi8CqNwx6Q==', 'Петрова', 'Виктория', 'Григорьевна', '06.01.1993'),

('NMAvQVgTACZ6/YBFhj+yTw==', 'malin@gmail.com', 'UOFgto+c+wWUR4Ibnyq+Jg==', 'Денисова', 'Арина', 'Романовна', '27.06.1999'),

('pOCii6Aj53jnl8lqKTdBXA==', 'phizntrg@icloud.com', 'MEteP3+9OXGK52nCED2nvw==', 'Тихомиров', 'Марк', 'Леонидович', '30.07.1995'),

('CfpfoogLm/52Rgp/XtQ+SQ==', 'leslie@outlook.com', 'GvgNDZ+2Oaq/McmXw60NHA==', 'Медведев', 'Марк', 'Михайлович', '22.08.2003'),

('IoKkAeLDFnU=', 'bsikdar@yahoo.ca', 'BMkfbs7m76UjZwHTSazceA==', 'Ульянов', 'Борис', 'Артёмович', '10.02.1973'),

('KK3HyGlsbpogb2F4baJPTQ==', 'kaiser@hotmail.com', 'Cr9BBzcRk8TW1v0Dp5VedQ==', 'Винокуров', 'Кирилл', 'Арсентьевич', '08.03.1974'),

('1QskxW76iM4=', 'jsbach@aol.com', 'wDHCrHCRhYxuUTq50NVurQ==', 'Панин', 'Максим', 'Маркович', '20.09.1995'),

('xECcRbiwcoLWy4r0a/XOAA==', 'benanov@sbcglobal.net', '6+I0pJRGljj+jgrcgsZ64g==', 'Васильев', 'Тимур', 'Максимович', '10.12.2003'),

('87VyEy0SEg14hB0ud6WzCg==', 'eidac@sbcglobal.net', 'U1fxfzlDrnVJ87ByHqJfkQ==', 'Дементьева', 'Полина', 'Викторовна', '30.09.1971'),

('lnC1Htm3kpk=', 'parasite@outlook.com', 'lD3qXWkqJ3G5bT4xXgv1hQ==', 'Андреев', 'Александр', 'Тимофеевич', '07.01.1981'),

('PyQi8acKryj+X/r0IpG/Ng==', 'moxfulder@yahoo.com', 'XermscU5w6YJuUDIaZI0hQ==', 'Кудрявцев', 'Сергей', 'Дмитриевич', '16.05.2000'),

('1DD8/KCSlXYQWWOpkVRDcw==', 'animats@optonline.net', 'y1Z/E4huq7CcLK5WLb7hVA==', 'Александров', 'Семён', 'Андреевич', '11.04.2002'),

('PIbgzNreJZY=', 'jeteve@me.com', 'q5pI3eYae9XROuF+pylTTg==', 'Хохлов', 'Ярослав', 'Иванович', '20.11.1978'),

('cYRfx32GTbQ=', 'lamky@icloud.com', 'CIJYXKYn/HgtqomdzsNcTQ==', 'Лебедев', 'Богдан', 'Александрович', '17.07.1984')--,

GO

USE VehicleRental

INSERT DriversLicences VALUES

('7878443544', '2012-03-12' ,2),

('8989834554', '2016-04-12' ,1),

('5463454355', '2017-09-12' ,2),

('5434523453', '2018-05-12' ,1)

GO

EXEC AddCategoryToDriverLicence @DriverLicence\_ID='7878443544', @Category='B', @ReceiptDate = '2012-09-12', @EndDate = '2022-09-12'

GO

EXEC AddCategoryToDriverLicence @DriverLicence\_ID='8989834554', @Category='B', @ReceiptDate = '2016-04-12', @EndDate = '2026-04-12'

GO

EXEC AddCategoryToDriverLicence @DriverLicence\_ID='5463454355', @Category='A', @ReceiptDate = '2017-08-12', @EndDate = '2027-08-12'

GO

EXEC AddCategoryToDriverLicence @DriverLicence\_ID='5434523453', @Category='B', @ReceiptDate = '2018-05-12', @EndDate = '2028-05-12'

GO

USE VehicleRental

INSERT Classes (Class) VALUES

('Бизнес'),

('Эконом'),

('Стандарт'),

('Спорт'),

('Комфорт')

GO

USE VehicleRental

INSERT Vehicles VALUES

(146.0, 'Бизнес', NULL),

(70.9, 'Эконом', NULL),

(97.8, 'Стандарт', NULL),

(230.1, 'Спорт', NULL),

(170.5, 'Бизнес', NULL),

(130.54, 'Комфорт', NULL),

(150.23, 'Стандарт', NULL),

(168.1, 'Спорт', NULL),

(146.4, 'Бизнес', NULL),

(70.32, 'Эконом', NULL),

(97.12, 'Стандарт', NULL),

(230.43, 'Спорт', NULL),

(170.85, 'Бизнес', NULL),

(130.5, 'Комфорт', NULL),

(150.7, 'Стандарт', NULL),

(160.3, 'Спорт', NULL)

GO

USE VehicleRental

INSERT VehicleRegistrCertificates VALUES

(4234, 123343, 1 ,'52LD5432123', 'О435РУ777', 'MERCEDES-BENZ', 'Gelandewagen', 'Внедорожник', 'B', 2021, 'NO', '2342342341', 'Черный', 200, 'Четвёртый', 1200, 1100),

(2343, 423423, 2 ,'43MK3244212', 'А435КА047', 'LADA', 'Kalina', 'Седан', 'B', 2011, 'NO', '234SDF2341', 'Синий', 120, 'Четвёртый', 1200, 1100),

(6432, 234212, 3 ,'54TY4321134', 'Л834РС777', 'KIA', 'RIO', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Белый', 120, 'Четвёртый', 1200, 1100),

(1243, 435344, 4 ,'12HR6573423', 'В888ВВ888', 'Lamborghini ', 'Aventador', 'Спортивная', 'B', 2014, 'NO', '435423FD342', 'Оранжевый', 450, 'Четвёртый', 1200, 1100),

(3452, 123412, 5 ,'52HJ5652523', 'А555АА052', 'MERCEDES-BENZ', 'C-CLASS', 'Седан', 'B', 2021, 'NO', '2342342341', 'Серый', 200, 'Четвёртый', 1200, 1100),

(5476, 452323, 6 ,'42MK3244212', 'С347БА112', 'LADA', 'Granta', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Коричневый', 120, 'Четвёртый', 1200, 1100),

(1234, 123124, 7 ,'52TY5321334', 'Р333УС321', 'LADA', 'Riva', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Вишневый', 120, 'Четвёртый', 1200, 1100),

(1443, 173412, 8 ,'12HR6523223', 'Н123НН123', 'Lamborghini ', 'Gollardo', 'Спортивная', 'B', 2011, 'NO', '435423FD342', 'Белый', 450, 'Четвёртый', 1200, 1100),

(4313, 435123, 9 ,'23HG5324143', 'О435РП542', 'SKODA', 'OCTAVIA II', 'Седан', 'B', 2021, 'NO', '2342342341', 'Белый', 200, 'Четвёртый', 1200, 1100),

(5674, 324344, 10 ,'43DF2345243', 'С435ВК047', 'RENAULT', 'SCÉNICII', 'Седан', 'B', 2011, 'NO', '234SDF2341', 'Коричневый', 120, 'Четвёртый', 1200, 1100),

(3245, 763456, 11 ,'52TY4321234', 'У834КО777', 'BMW', 'X1 (E84)', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Белый', 120, 'Четвёртый', 1200, 1100),

(1234, 123445, 12 ,'12HR6573523', 'К888ТО888', 'SKODA ', 'KODIAQ', 'Внедорожник', 'B', 2014, 'NO', '435423FD342', 'Белый', 450, 'Четвёртый', 1200, 1100),

(3244, 342553, 13 ,'51LD5432123', 'В555СТ052', 'OPEL', 'ASTRA', 'Седан', 'B', 2021, 'NO', '2342342341', 'Серый', 200, 'Четвёртый', 1200, 1100),

(5234, 143322, 14 ,'43MK3243452', 'С347УВ112', 'TOYOTA', 'HILUX VII', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Серый', 120, 'Четвёртый', 1200, 1100),

(3242, 423521, 15 ,'54TY4321234', 'О333КУ321', 'MERCEDES-BENZ', 'E-CLASS', 'Седан', 'B', 2021, 'NO', '234SDF2341', 'Серебристый', 120, 'Четвёртый', 1200, 1100),

(3544, 643422, 16 ,'43HY1123423', 'Н123ТК123', 'Porsche ', 'Cayenne', 'Внедорожник', 'B', 2011, 'NO', '435423FD342', 'Синий', 450, 'Четвёртый', 1200, 1100)

GO

DECLARE @YESTERDAY SMALLDATETIME = DATEADD(day, -1, GETDATE())

USE VehicleRental

INSERT VehicleCoordinates VALUES

(1, 37.628310329218785, 55.783562264913726, @YESTERDAY),

(2, 37.62496863084447, 55.77968409403343, @YESTERDAY),

(3, 37.60988838934319, 55.771697388745174, @YESTERDAY),

(4, 37.63784366449218, 55.80634960848172, @YESTERDAY),

(5, 37.64351317690392, 55.75885752573489, @YESTERDAY),

(6, 37.650152394817006, 55.79318597301102, @YESTERDAY),

(7, 37.64382665235103, 55.767218191902586, @YESTERDAY),

(8, 37.631991990646156, 55.754565408911894, @YESTERDAY),

(9, 37.60679471254363, 55.76311732978595, @YESTERDAY),

(10, 37.60886742602116, 55.7729332932597, @YESTERDAY),

(11, 37.61629258607224, 55.79079936483427, @YESTERDAY),

(12, 37.62285999673869, 55.81227549857955, @YESTERDAY),

(13, 37.61330564444986, 55.79759711490164, @YESTERDAY),

(14, 37.64433763929462, 55.81538058758365, @YESTERDAY),

(15, 37.644649815415036, 55.79659831246247, @YESTERDAY),

(16, 37.612114115149616, 55.78038656884442, @YESTERDAY)

GO

USE VehicleRental

INSERT Rentals VALUES

('8989834554', 1 , GETDATE(), '12:00', 2, 123.7, 'досрочная'),

('5463454355', 3 , GETDATE(), '15:00', 3, 304.2, 'досрочная'),

('8989834554', 4 , GETDATE(), '12:00', 1, 855.2, 'досрочная'),

('5463454355', 7 , GETDATE(), '15:00', 3, 1610.7, 'стандартная')

GO

USE VehicleRental

INSERT TrafficAccidentTypes VALUES

('ДТП'),

('Неисправность тс, не связанная с пользователем')

GO

USE VehicleRental

INSERT TrafficAccidents VALUES (3, 5463454355,1, 7600.0, 'Врезался в киоск')

GO

USE VehicleRental

INSERT Rental\_Admins VALUES (1)

# ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ

Отмечу здесь несколько важных окон из моей программы.

## Окно «Входа»

XAML:

<Window x:Class="CarSharingApplication.Autorization"

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

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

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

xmlns:local="clr-namespace:CarSharingApplication"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:TextValidation="clr-namespace:CarSharingApplication.Validation"

MinHeight="520" MinWidth="390"

Height="520" Width="390"

Title="Вход"

WindowStartupLocation="CenterScreen"

WindowState="Maximized"

Icon="../Images/icon.jpeg"

>

<Window.DataContext>

<TextValidation:AuthValidation/>

</Window.DataContext>

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Light.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/MaterialDesignColor.DeepPurple.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/MaterialDesignColor.Lime.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Window.Background>

<LinearGradientBrush EndPoint="0.5,0.6" StartPoint="0.5,0">

<GradientStop Color="Blue"/>

<GradientStop Color="WhiteSmoke" Offset=".9"/>

</LinearGradientBrush>

</Window.Background>

<Grid x:Name="MainGrid" VerticalAlignment="Center">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition MinWidth="250" Width="350"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Border x:Name="AutorizarionField" Grid.Column="1" Background="White" MinHeight="450" Height="auto" CornerRadius="20" VerticalAlignment="Top" BorderBrush="DarkGray" BorderThickness="0.2">

<Grid x:Name="AutorizationGrid" VerticalAlignment="Top">

<Grid.RowDefinitions>

<RowDefinition MinHeight="20"/>

<RowDefinition MinHeight="300"/>

<RowDefinition MaxHeight="100" MinHeight="100"/>

</Grid.RowDefinitions>

<Grid Grid.Row="1" x:Name="InputGrid" VerticalAlignment="Center">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition MinHeight="60"/>

</Grid.RowDefinitions>

<Grid x:Name="LoginGrid" Grid.Row="0" Tag="0">

<TextBox Text="{Binding login\_, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" x:Name="Login" Margin="20 0 20 0" Padding="0 6 0 5" CharacterCasing="Lower" KeyDown="Login\_TextChanged" materialDesign:HintAssist.Hint="Имя для входа"/>

</Grid>

<Grid x:Name="EmailGrid" Grid.Row="1" Tag="1">

<TextBox Text="{Binding email\_, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" x:Name="Email" Margin="20,0,20,0" Padding="0 6 0 5" CharacterCasing="Lower" Height="28" VerticalAlignment="Bottom" materialDesign:HintAssist.Hint="Почта"/>

</Grid>

<Grid x:Name="PasswordGrid" Grid.Row="2" Tag="2">

<PasswordBox x:Name="Password" Margin="20 0 20 0" Padding="0 6 0 5" materialDesign:HintAssist.Hint="Пароль"></PasswordBox>

</Grid>

<Grid x:Name="RepeatPasswordGrid" Grid.Row="3" Tag="3">

<PasswordBox x:Name="RepeatPassword" Margin="20 0 20 0" Padding="0 6 0 5" materialDesign:HintAssist.Hint="Повторите пароль"></PasswordBox>

</Grid>

<Grid x:Name="UserSurnameGrid" Grid.Row="4" Tag="4">

<TextBox Text="{Binding surnname\_, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" x:Name="UserSurname" Margin="20 0 20 0" Padding="0 6 0 5" materialDesign:HintAssist.Hint="Фамилия"/>

</Grid>

<Grid x:Name="UserNameGrid" Grid.Row="5" Tag="5">

<TextBox Text="{Binding username\_, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" x:Name="UserName" Margin="20 0 20 0" Padding="0 6 0 5" materialDesign:HintAssist.Hint="Имя"/>

</Grid>

<Grid x:Name="UserMiddleNameGrid" Grid.Row="6" Tag="6">

<TextBox Text="{Binding middlename\_, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" x:Name="UserMiddleName" Margin="20 0 20 0" Padding="0 6 0 5" materialDesign:HintAssist.Hint="Отчество"/>

</Grid>

<Grid x:Name="BDatePickerGrid" Grid.Row="7" Tag="7">

<DatePicker x:Name="BDatePicker" Panel.ZIndex="1" Margin="20 0 20 0" Padding="0 6 0 5" SelectedDateFormat="Short" materialDesign:HintAssist.Hint="Дата рождения"></DatePicker>

</Grid>

<Grid x:Name="inputButtonGrid" Grid.Row="8" Tag="8" Margin="0" >

<Button x:Name="inputButton" Margin="20 0 20 0" VerticalAlignment="Center" Background="green" BorderBrush="green" MinHeight="30" MaxHeight="30"></Button>

</Grid>

</Grid>

<Grid x:Name="Button\_Grid" Grid.Row="2" Margin="0,0,0,10" VerticalAlignment="Bottom">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Grid.Row="0" Margin="20,0,20,10" VerticalAlignment="Top" Click="Reg\_Button\_Click" Background="BlueViolet">Регистрация</Button>

<Button Grid.Row="1" Margin="20,0,20,10" Click="Login\_Button\_Click" Background="BlueViolet">Вход</Button>

</Grid>

<Label

x:Name="lbl"

Grid.Column="1"

Height="60"

VerticalAlignment="Top"

Panel.ZIndex="1"

FontSize="30"

HorizontalAlignment="Center"

VerticalContentAlignment="Center"

IsEnabled="False"

FontFamily="Adobe Caslon Pro Bold"

Foreground="#E5313193"

FontWeight="Bold"

Opacity="1" >

<!--Авторизация-->

</Label>

</Grid>

</Border>

</Grid>

</Window>

C#:

using CarSharingApplication.LogLibrary;

using CarSharingApplication.SQL.Linq;

using CarSharingApplication.Windows.Authorization;

using CarSharingApplication.Windows.VehicleRent;

using System;

using System.Data.SqlClient;

using System.Linq;

using System.Security.Cryptography;

using System.Text.Encodings.Web;

using System.Text.Json;

using System.Text.Unicode;

using System.Windows;

using System.Windows.Input;

namespace CarSharingApplication

{

/// <summary>

/// Логика взаимодействия для Autorization.xaml

/// </summary>

public partial class Autorization : Window

{

private int condition = 0;

public Autorization()

{

App.\_Logger.options = new JsonSerializerOptions

{

Encoder = JavaScriptEncoder.Create(UnicodeRanges.BasicLatin, UnicodeRanges.Cyrillic),

};

App.\_Logger.LogPath = Environment.CurrentDirectory + @"\logs.json";

InitializeComponent();

Reg\_Button\_Click(null, null);

}

private void MakeRegister(object sender, RoutedEventArgs e)

{

bool registered = false;

if (Login.Text.Length < 5)

MessageBox.Show("Слишком мало символов в имени входа");

else if ((Email.Text.Length < 10) || (Email.Text.Count((c) => c == '@') != 1) || (Email.Text.Count((c) => c == '.') != 1) || (Email.Text.Contains("@.")) || (Email.Text.Contains(".@")))

MessageBox.Show("Почта не является корректной");

else if (Password.Password.Length < 8)

MessageBox.Show("Слишком короткий пароль");

else if (Password.Password != RepeatPassword.Password)

MessageBox.Show("Пароли не совпадают");

else if (UserSurname.Text.Length == 0)

MessageBox.Show("Вы не ввели фамилию");

else if (UserName.Text.Length == 0)

MessageBox.Show("Вы не ввели имя");

else if (UserMiddleName.Text.Length == 0)

MessageBox.Show("Вы не ввели отчество");

else if (BDatePicker.Text == "")

MessageBox.Show("Вы не ввели дату рождения");

else if (DateTime.Parse(BDatePicker.Text) >= DateTime.UtcNow.AddYears(-18) || DateTime.Parse(BDatePicker.Text) < DateTime.UtcNow.AddYears(-100) || DateTime.Parse(BDatePicker.Text) >= DateTime.UtcNow)

MessageBox.Show("Некоректная дата рождения");

else

{

try

{

#region NEW CODE SUPPORT FROM 25.04.2023

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

string encLogin = PasswordEncryptor.EncryptString(Login.Text);

string encPass = PasswordEncryptor.EncryptString(Password.Password);

if (App.GetQueryResult<string>(new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection")),

$"SELECT UserLogin FROM Rental\_Users WHERE UserLogin = '{encLogin}'").Count > 0)

{

MessageBox.Show("Логин занят");

return;

}

var answ = App.GetScalarResult<int>(new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection")),

$"SELECT [dbo].CheckExistingUser('{encLogin}','{encPass}')");

if (answ == -1)

{

if (App.ExecuteNonQuery(new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection")),

"EXEC REG\_USER " +

$"@UserLogin='{encLogin}', " +

$"@UserEmail='{Email.Text}', " +

$"@UserPassword='{encPass}', " +

$"@UserSurname='{UserSurname.Text}', " +

$"@UserName='{UserName.Text}', " +

$"@UserMiddleName='{UserMiddleName.Text}'," +

$"@UserBirthDayDate='{BDatePicker.Text}'"))

registered = true;

}

else if (answ > 0)

{

registered = true;

}

else

{

MessageBox.Show("Проверьте корректность введённых данных");

}

}

#endregion

}

catch (SqlException SqlEx)

{

App.\_Logger.Log(new LogMessage(null, this.Title, SqlEx.Message, SqlEx.ErrorCode.ToString(), LogType.DataBaseError));

}

finally

{

if (registered)

Login\_Button\_Click(null, null);

}

}

}

private void MakeLogIn(object sender, RoutedEventArgs e)

{

try

{

#region NEW CODE SUPPORT FROM 25.04.2023

if (Login.Text.Length < 5)

MessageBox.Show("Слишком мало символов в имени входа");

else if (Password.Password.Length < 8)

MessageBox.Show("Слишком короткий пароль");

else

{

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

string encLogin = PasswordEncryptor.EncryptString(Login.Text);

string encPass = PasswordEncryptor.EncryptString(Password.Password);

var answ = App.GetScalarResult<int>(new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection")),

$"SELECT [dbo].CheckExistingUser('{encLogin}','{encPass}')");

if (answ > 0)

{

this.Visibility = Visibility.Collapsed;

UsersINFO UserInfo = App.GetScalarResult<UsersINFO>(new CarSharingDataBaseClassesDataContext(App.GetConnectionString("USERHANDLERConnection")), $"SELECT \* FROM UsersINFO WHERE ID\_User = {answ}");

if (UserInfo != null)

{

App.\_Logger.Log(new LogMessage((ulong)UserInfo.ID\_User, this.Title, "Вошёл в систему", null, null));

if (UserInfo.Previlege.TrimEnd() == "админ")

{

var ChoiceAuthorizationWindow = new ChoiceLoginWindow(ref UserInfo, this);

ChoiceAuthorizationWindow.Activate();

ChoiceAuthorizationWindow.Show();

}

else

{

if (UserInfo.RentStatus != "в поездке")

{

var CarSelWindow = new CarSelector(ref UserInfo, this, true);

CarSelWindow.Activate();

CarSelWindow.Show();

}

else

{

var TripWND = new TripWindow(ref UserInfo, this, true);

TripWND.Activate();

TripWND.Show();

this.Visibility = Visibility.Collapsed;

}

}

ClearFields();

}

}

else App.\_Logger.Log(new LogMessage(null,this.Title,"Пользователь не найден","ошибка авторизации",LogType.UserMistake));

}

}

#endregion

}

catch (SqlException SqlEx)

{

App.\_Logger.Log(new LogMessage(null,this.Title, SqlEx.Message, SqlEx.ErrorCode.ToString(), LogType.DataBaseError));

}

}

private void HideFields()

{

EmailGrid.Visibility = Visibility.Collapsed;

RepeatPasswordGrid.Visibility = Visibility.Collapsed;

UserSurnameGrid.Visibility = Visibility.Collapsed;

UserNameGrid.Visibility = Visibility.Collapsed;

UserMiddleNameGrid.Visibility = Visibility.Collapsed;

BDatePickerGrid.Visibility = Visibility.Collapsed;

}

private void ShowFields()

{

EmailGrid.Visibility = Visibility.Visible;

RepeatPasswordGrid.Visibility = Visibility.Visible;

UserSurnameGrid.Visibility = Visibility.Visible;

UserNameGrid.Visibility = Visibility.Visible;

UserMiddleNameGrid.Visibility = Visibility.Visible;

BDatePickerGrid.Visibility = Visibility.Visible;

}

private void Reg\_Button\_Click(object sender, RoutedEventArgs e)

{

if (condition != 2)

{

ShowFields();

lbl.Content = "Аутентификация";

inputButton.Click -= MakeLogIn;

inputButton.Click += MakeRegister;

inputButton.Content = "Зарегистрироваться";

}

}

private void Login\_Button\_Click(object sender, RoutedEventArgs e)

{

if (condition != 1)

{

HideFields();

lbl.Content = "Авторизация";

inputButton.Click -= MakeRegister;

inputButton.Click += MakeLogIn;

inputButton.Content = "Войти";

}

}

private void ClearFields()

{

Login.Clear();

Email.Clear();

Password.Clear();

RepeatPassword.Clear();

UserSurname.Clear();

UserName.Clear();

UserMiddleName.Clear();

BDatePicker.Text = "";

}

private void Login\_TextChanged(object sender, KeyEventArgs e)

{

}

}

}

## Окно «Выбора авто»

XAML:

<Window x:Class="CarSharingApplication.CarSelector"

xmlns:WindowsPresentation="clr-namespace:GMap.NET.WindowsPresentation;assembly=GMap.NET.WindowsPresentation"

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

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

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

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

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

xmlns:Models="clr-namespace:CarSharingApplication.Windows.XAMLModels"

xmlns:local="clr-namespace:CarSharingApplication"

mc:Ignorable="d"

Icon="../images/icon.jpeg"

MinHeight="800"

Width="1359"

ResizeMode="NoResize"

WindowStartupLocation="CenterScreen"

Closing="Window\_Closing"

Title="Окно выбора ТС">

<Grid>

<StackPanel Margin="10" Width="1339">

<Border

VerticalAlignment="Top"

Margin="0 0 0 0"

>

<WrapPanel>

<Button x:Name="BackButton" Background="red" BorderBrush="red" Click="BackButton\_Click" >Выход</Button>

<Button x:Name="PersonalAccountButton" Background="Orange" BorderBrush="Orange" Click="PersonalAccountButton\_Click"> Личный кабинет</Button>

</WrapPanel>

</Border>

<WrapPanel HorizontalAlignment="Left" Height="620">

<Border

VerticalAlignment="Top"

Width="300"

MinHeight="600"

Height="600"

CornerRadius="10"

Background="DarkOrange"

BorderBrush="Orange"

BorderThickness="3"

Margin="0 10 10 0">

<ScrollViewer Margin="10" HorizontalAlignment="Stretch" HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto">

<StackPanel>

<Expander Header="Класс ТС" Width="280" Foreground="White" FontSize="36" Background="Transparent" HorizontalAlignment="Left" FontWeight="Bold" FontFamily="segoi ui">

<ListView x:Name="ListViewVehicleClasses" Padding="0" Foreground="White" FontSize="15" FontWeight="DemiBold" Margin="40 0" HorizontalAlignment="Left"/>

</Expander>

<Expander Header="Марка ТС" Width="280" Foreground="White" FontSize="36" Background="Transparent" HorizontalAlignment="Left" FontWeight="Bold" FontFamily="segoi ui">

<ListView x:Name="ListViewVehicleBrands" Padding="0" Foreground="White" FontSize="15" FontWeight="DemiBold" Margin="40 0" HorizontalAlignment="Left"/>

</Expander>

<Expander Header="Категория" Width="280" Foreground="White" FontSize="36" Background="Transparent" HorizontalAlignment="Left" FontWeight="Bold" FontFamily="segoi ui">

<ListView x:Name="ListViewVehicleCategories" Padding="0" Foreground="White" FontSize="15" FontWeight="DemiBold" Margin="40 0" HorizontalAlignment="Left"/>

</Expander>

<Expander Header="Ценовая категория" Width="280" Foreground="White" FontSize="36" Background="Transparent" HorizontalAlignment="Left" FontWeight="Bold" FontFamily="segoi ui">

<ListView x:Name="ListPrice" Padding="0" Foreground="White" FontSize="15" FontWeight="DemiBold" Margin="40 0" HorizontalAlignment="Left">

<Slider x:Name="PriceSlider" Width="180"></Slider>

<TextBox x:Name="MaxPrice" Text="{Binding ElementName=PriceSlider, Path=Value, StringFormat=N2}" FontWeight="Bold" IsEnabled="False" materialDesign:TextFieldAssist.UnderlineBrush="Transparent"/>

</ListView>

</Expander>

</StackPanel>

</ScrollViewer>

</Border>

<Models:VehicleCard x:Name="Card" Margin="0 10 10 0" Height="600"/>

<Models:Map x:Name="RentalMap" Margin="0 10 10 0" Width="600" Height="600" InitializeLat="55.755969527097506" InitializeLng="37.61763538248735"/>

</WrapPanel>

<WrapPanel>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition MinWidth="300"/>

<ColumnDefinition MinWidth="410"/>

<ColumnDefinition MinWidth="610"/>

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Height="70" FontSize="16" Click="SearchByCriteries" materialDesign:ButtonAssist.CornerRadius="10" VerticalAlignment="Top">Поиск По Критериям</Button>

<Button Grid.Column="1" Margin="10 0 0 0" Height="70" FontSize="16" materialDesign:ButtonAssist.CornerRadius="10" VerticalAlignment="Top">Поиск Ближайшего Транспортного Средства</Button>

<Button Grid.Column="2" Margin="11 0 0 0" Width="598" Height="70" materialDesign:ButtonAssist.CornerRadius="10" VerticalAlignment="Top" Background="Orange" BorderBrush="OrangeRed" BorderThickness="2" Padding="0 0 0 10" FontSize="28" Click="RentalButton\_Click">Арендовать</Button>

<!--<Button Margin="10 0 0 0" Height="100" FontSize="16" Width="300" Click="PreviousVehicleButtonClick">

<materialDesign:PackIcon Kind="ArrowLeftBoldOutline" Width="216" Height="66" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Button>

<Button Margin="0 0 0 0" Height="100" FontSize="16" Width="300" Click="NextVehicleButtonClick">

<materialDesign:PackIcon Kind="ArrowRightBoldOutline" Width="216" Height="66" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Button>-->

</Grid>

</WrapPanel>

</StackPanel>

</Grid>

</Window>

C#:

using CarSharingApplication.LogLibrary;

using CarSharingApplication.SQL.Linq;

using CarSharingApplication.Windows.Moderating.EditWindows.Users;

using CarSharingApplication.Windows.VehicleRent;

using CarSharingApplication.Windows.VehicleSelector;

using GMap.NET;

using GMap.NET.WindowsPresentation;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media.Imaging;

namespace CarSharingApplication

{

/// <summary>

/// Логика взаимодействия для CarSelector.xaml

/// </summary>

public partial class CarSelector : Window

{

private UsersINFO \_User = null;

private bool \_ShowOwner;

private string ConnectionString { get { return App.GetConnectionString("CARHANDLERConnection"); } }

/// <summary>

/// Текст в случае если в системе нет автомобилей соответстующим критериям пользователя

/// </summary>

private string ZeroVehiclesByCriteries = "Отсутствуют транспотрные\nсредтва соответствующие\nзаданным критериям";

/// <summary>

/// Текст в случае если нет свободных автомобилей в системе

/// </summary>

private string HaveNotAvaliableVehicles = "В данный момент\nнет свободных авто\nзаходите позже";

public CarSelector(ref UsersINFO user, Window owner, bool showOwner)

{

this.Owner = owner;

\_ShowOwner = showOwner;

\_User = user;

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

InitializeComponent();

GetVehiclesData(VehiclesData.GetInstance);

try

{

var instance = VehiclesData.GetInstance;

PriceSlider.Minimum = Double.Parse((instance.vehiclesInfoList.Min(veh => veh.PricePerHour)).ToString());

PriceSlider.Maximum = Double.Parse((instance.vehiclesInfoList.Max(veh => veh.PricePerHour)).ToString());

PriceSlider.Value = PriceSlider.Maximum;

if (instance.vehiclesInfoList.Count > 0)

{

SetInfo(instance);

}

}

catch

{

PriceSlider.Minimum = 0.0;

PriceSlider.Maximum = 0.0;

PriceSlider.Value = 0.0;

SetVehicleInfo(null, HaveNotAvaliableVehicles);

}

}

/// <summary>

/// Показывать окно родителя при закрытии

/// </summary>

/// <param name="value"></param>

public void OwnerShow(bool value)

{

\_ShowOwner = value;

}

/// <summary>

/// Отобразить информацию в окно

/// </summary>

/// <param name="vehData"></param>

public void SetInfo(VehiclesData vehData)

{

vehData.selectedVehicle = vehData.vehiclesInfoList.First();

RentalMap.SetMarkers(GetMarkers(vehData.vehiclesInfoList));

SetVehicleInfo(vehData.selectedVehicle, ZeroVehiclesByCriteries);

}

/// <summary>

/// Получение/обновление данных о авто

/// </summary>

/// <param name="vehData"></param>

private void GetVehiclesData(VehiclesData vehData)

{

App.AppDataBase.OpenConnection(ConnectionString);

vehData.vehClasses = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(UPPER(Class)) FROM VehiclesINFO");

vehData.vehClasses.Add("\*ВСЕ");

ListViewVehicleClasses.ItemsSource = vehData.vehClasses.OrderBy(str => str);

vehData.vehBrands = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(UPPER(Brand)) FROM VehiclesINFO");

vehData.vehBrands.Add("\*ВСЕ");

ListViewVehicleBrands.ItemsSource = vehData.vehBrands.OrderBy(str => str);

vehData.vehiclesInfoList = App.AppDataBase.GetQueryResult<VehiclesINFO>(

"SELECT \* FROM VehiclesWithStatus ('доступен')");

vehData.vehiclesInfoList = OrderByPricePerHourDesc(vehData.vehiclesInfoList);

vehData.vehCategories = App.AppDataBase.GetQueryResult<string>(

"SELECT DISTINCT TRIM(Vehicle\_Category) FROM VehiclesINFO");

vehData.vehCategories.Add("\*ВСЕ");

ListViewVehicleCategories.ItemsSource = vehData.vehCategories.OrderBy(str => str);

}

/// <summary>

/// Нажатие кнопки, выхода

/// </summary>

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

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

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

{

this.Close();

}

/// <summary>

/// Получить список маркеров из списка информации об авто

/// </summary>

/// <param name="vehiclesINFOs"></param>

/// <returns></returns>

private List<GMapMarker> GetMarkers(List<VehiclesINFO> vehiclesINFOs)

{

try

{

List<GMapMarker> VehiclesMarkers = new List<GMapMarker>();

foreach (var vehicle in vehiclesINFOs)

{

RentalMap.MapController.Markers.OrderBy(mark => mark.Tag);

if (vehicle.Lat != null && vehicle.Lng != null)

{

GMapMarker marker = new GMapMarker(new PointLatLng((double)vehicle.Lat, (double)vehicle.Lng));

marker.Tag = vehicle.ID\_Vehicle;

marker.Shape = new System.Windows.Controls.Image

{

Source = new BitmapImage(new Uri($@"{App.path}\Windows\Images\CarMarker2.png")),

Width = 30,

Height = 30,

ToolTip = $"{vehicle.Brand} {vehicle.Mark}",

Visibility = Visibility.Visible,

Tag = vehicle

};

marker.Shape.MouseEnter += (s, e) =>

{ Mouse.OverrideCursor = Cursors.Hand;};

marker.Shape.MouseLeave += (s, e) =>

{ Mouse.OverrideCursor = Cursors.Arrow; };

marker.Shape.MouseLeftButtonDown += MarkerClick;

VehiclesMarkers.Add(marker);

}

}

return VehiclesMarkers;

}

catch (SqlException sqelx)

{

MessageBox.Show(sqelx.Message);

return null;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

}

/// <summary>

/// Вывести информацию о автомобиле

/// </summary>

/// <param name="info"></param>

/// <param name="errorMessage"></param>

#nullable enable

private void SetVehicleInfo(VehiclesINFO? info, string errorMessage)

{

Card.SetVehicleInfo(info, errorMessage);

}

/// <summary>

/// Нажатие на маркер автобиля

/// </summary>

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

/// <param name="args"></param>

private void MarkerClick(object sender, MouseEventArgs args)

{

var marker = (System.Windows.Controls.Image)sender;

VehiclesData.GetInstance.selectedVehicle = (VehiclesINFO)(marker.Tag);

SetVehicleInfo(VehiclesData.GetInstance.selectedVehicle, ZeroVehiclesByCriteries);

RentalMap.MoveCursorToVehicleOnMap(VehiclesData.GetInstance.selectedVehicle);

}

/// <summary>

/// Найти автомобиль/автомобили по критериям

/// </summary>

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

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

private void SearchByCriteries(object sender, RoutedEventArgs e)

{

var instance = VehiclesData.GetInstance;

GetVehiclesData(instance);

List<VehiclesINFO> newvehicleslist = instance.vehiclesInfoList.Where(vehicle => Double.Parse(vehicle.PricePerHour.ToString()) <= PriceSlider.Value).ToList();

#nullable enable

if ((string)ListViewVehicleClasses.SelectedValue != "\*ВСЕ" && ListViewVehicleClasses.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Class.ToUpper().TrimEnd() == (string)ListViewVehicleClasses.SelectedValue).ToList();

}

#nullable enable

if ((string)ListViewVehicleBrands.SelectedValue != "\*ВСЕ" && ListViewVehicleBrands.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Brand.ToUpper().TrimEnd() == (string)ListViewVehicleBrands.SelectedValue).ToList();

}

#nullable enable

if ((string)ListViewVehicleCategories.SelectedValue != "\*ВСЕ" && ListViewVehicleCategories.SelectedValue != null)

{

newvehicleslist = newvehicleslist.Where(vehicle => vehicle.Vehicle\_Category.TrimEnd() == (string)ListViewVehicleCategories.SelectedValue).ToList();

}

newvehicleslist = OrderByPricePerHourDesc(newvehicleslist);

instance.vehiclesInfoList = newvehicleslist;

RentalMap.MapController.Markers.Clear();

GC.Collect();

RentalMap.SetMarkers(GetMarkers(newvehicleslist));

if (newvehicleslist.Count > 0)

{

instance.selectedVehicle = newvehicleslist.First();

SetVehicleInfo(instance.selectedVehicle, ZeroVehiclesByCriteries);

RentalMap.MoveCursorToVehicleOnMap(instance.selectedVehicle);

}

else

{

instance.selectedVehicle = null;

SetVehicleInfo(null, ZeroVehiclesByCriteries);

}

}

/// <summary>

/// Сортировать автомобили по цене

/// </summary>

/// <param name="list"></param>

/// <returns>List<VehiclesINFO</returns>

public List<VehiclesINFO> OrderByPricePerHourDesc(List<VehiclesINFO> list) => (from vehicle

in list

orderby vehicle.PricePerHour descending

select vehicle).ToList();

/// <summary>

/// Нажатие кнопки Арендовать

/// </summary>

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

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

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

{

var instance = VehiclesData.GetInstance;

if (instance.selectedVehicle != null)

{

var rentWindow = new VehicleRent(\_User, instance.selectedVehicle, this, true);

this.Visibility = Visibility.Collapsed;

rentWindow.Activate();

rentWindow.Show();

}

else

{

MessageBox.Show("Вы не выбрали авто");

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, "Пользователь не выбрал ТС", "Ошибка нажатия кнопки аренды ТС", 0));

}

}

/// <summary>

/// Нажатие кнопки Личный кабинет

/// </summary>

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

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

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

{

var persWindow = new PersonalAccount(ref \_User);

persWindow.Owner = this;

persWindow.Show();

this.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Закрытие окна

/// </summary>

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

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

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

GC.Collect();

}

else

{

this.Owner.Close();

}

}

/// <summary>

/// Показать информацию о следующем автомобиле

/// </summary>

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

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

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

{

var instance = VehiclesData.GetInstance;

var index = instance.vehiclesInfoList.IndexOf(instance.selectedVehicle);

if (instance.vehiclesInfoList.Count > 0)

{

if ((index < instance.vehiclesInfoList.Count-1))

{

instance.selectedVehicle = instance.vehiclesInfoList[index + 1];

}

else

{

instance.selectedVehicle = instance.vehiclesInfoList[0];

}

SetVehicleInfo(instance.selectedVehicle, ZeroVehiclesByCriteries);

RentalMap.MoveCursorToVehicleOnMap(instance.selectedVehicle);

}

}

}

}

## Окно «Аренда автомобиля»

XAML:

<Window x:Class="CarSharingApplication.Windows.VehicleRent.VehicleRent"

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:CarSharingApplication.Windows.VehicleRent"

xmlns:models="clr-namespace:CarSharingApplication.Windows.XAMLModels"

mc:Ignorable="d"

Icon="../Images/icon.jpeg"

Height="660" Width="860"

Closing="Window\_Closing"

ResizeMode="NoResize"

Title="Аренда ТС"

>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="420"/>

<ColumnDefinition Width="420"/>

</Grid.ColumnDefinitions>

<models:VehicleCard x:Name="Card" Grid.Column="0" VerticalAlignment="Top" Margin="0 10 0 10"/>

<StackPanel Grid.Column="1">

<models:CreditCard x:Name="CreditCard" VerticalAlignment="Top" Margin="0 10 0 10"/>

<models:RentalPricePicker x:Name="Picker" Margin="0 10 0 10"/>

<models:RentButton x:Name="rentbtn" Margin="0 10 0 10"></models:RentButton>

</StackPanel>

</Grid>

</Window>

C#:

using CarSharingApplication.LogLibrary;

using CarSharingApplication.SQL.Linq;

using System;

using System.Collections.Generic;

using System.Windows;

namespace CarSharingApplication.Windows.VehicleRent

{

/// <summary>

/// Логика взаимодействия для VehicleRent.xaml

/// </summary>

public partial class VehicleRent : Window

{

private UsersINFO \_User;

private VehiclesINFO \_Vehicle;

private bool \_ShowOwner;

private string DLHANDLERconnectionString { get; set; } = App.GetConnectionString("DLHANDLERConnection");

private string USERHANDLERconnectionString { get; set; } = App.GetConnectionString("USERHANDLERConnection");

public VehicleRent(UsersINFO user, VehiclesINFO Vehicle, Window owner, bool showOwner)

{

\_ShowOwner = showOwner;

this.Owner = owner;

InitializeComponent();

\_User = user;

\_Vehicle = Vehicle;

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

Card.SetVehicleInfo(Vehicle,"");

Picker.PricePerHour = (double)Vehicle.PricePerHour;

rentbtn.btn.Click += PayAndStart\_Click;

}

/// <summary>

/// Нажатие кнопки:

/// Оплатить и начать поездку

/// </summary>

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

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

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

{

if (rentbtn.cbox.IsChecked != true || CreditCard.isEmpty())

return;

if (\_User.ID\_DriverLicence == null)

{

MessageBox.Show("Вы не ввели данные о вод.удостоверении\nP.S. Это делается в личном кабинете");

this.Close();

return;

}

App.AppDataBase.OpenConnection(DLHANDLERconnectionString);

List<string> categories = App.AppDataBase.GetQueryResult<string>(

$"SELECT Category FROM [dbo].GetDriverLicenceCategories ('{\_User.ID\_DriverLicence}')");

App.AppDataBase.CloseConnection();

if (categories == null)

return;

if (!(categories.Contains(\_Vehicle.Vehicle\_Category.Trim().ToLower())))

{

MessageBox.Show("У вас отсутсвтует нужная категория в водительском удостоверении");

return;

}

App.AppDataBase.OpenConnection(USERHANDLERconnectionString);

if (App.AppDataBase.ExecuteNonQuery(

"EXEC Rent " +

$"@DriverLicence = '{\_User.ID\_DriverLicence}', " +

$"@ID\_Vehicle = {\_Vehicle.ID\_Vehicle}, " +

$"@RentalTime = '{DateTime.Now.ToString("HH:mm:ss")}', " +

$"@CountOfHours = {Picker.HourPicker.Value}, " +

$"@TotalPrice = {(double.Parse(\_Vehicle.PricePerHour.ToString()) \* Picker.HourPicker.Value).ToString().Replace(',', '.')}"))

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Арендовал ТС: {\_Vehicle.ID\_Vehicle}", null, LogType.UserAction));

try

{

\_ShowOwner = false;

var TripWND = new TripWindow(ref \_User, this.Owner.Owner, true);

TripWND.Activate();

TripWND.Show();

}

catch

{

\_ShowOwner = true;

}

{

this.Visibility = Visibility.Collapsed;

}

}

App.AppDataBase.CloseConnection();

}

/// <summary>

/// Закрытие окна

/// </summary>

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

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

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

}

else

{

this.Owner?.Close();

}

}

}

}

## Окно «Поездка»

XAML:

<Window x:Class="CarSharingApplication.Windows.VehicleRent.TripWindow"

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:CarSharingApplication.Windows.VehicleRent"

xmlns:models="clr-namespace:CarSharingApplication.Windows.XAMLModels"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

ResizeMode="NoResize"

Height="790" Width="430"

Closing="Window\_Closing"

Title="Окно поездки"

Icon="..\Images\icon.jpeg"

>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="80"/>

<RowDefinition Height="560"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<models:RentTimer x:Name="rt" Grid.Row="0" Margin="0 10 0 0" Height="80" Width="400"></models:RentTimer>

<models:VehicleCard x:Name="Card" Margin="0 0 0 0" Grid.Row="1" Height="540"/>

<Button Grid.Row="2" Margin="7 0 7 0" Height="45" Background="Yellow" VerticalAlignment="Top" BorderBrush="Black" Foreground="Black" BorderThickness="1" materialDesign:ButtonAssist.CornerRadius="10" Click="AccidentClick">Что делать, произошла авария?</Button>

<Button Grid.Row="3" Margin="7 5 7 0" Height="45" Background="Red" VerticalAlignment="Top" BorderBrush="DarkRed" BorderThickness="1" materialDesign:ButtonAssist.CornerRadius="10" Click="FinishTripClick">Закончить поездку</Button>

</Grid>

</Window>

C#:

using CarSharingApplication.LogLibrary;

using CarSharingApplication.SQL.Linq;

using CarSharingApplication.Windows.Moderating.EditWindows.Accidents;

using System;

using System.Configuration;

using System.Windows;

namespace CarSharingApplication.Windows.VehicleRent

{

/// <summary>

/// Логика взаимодействия для TripWindow.xaml

/// </summary>

public partial class TripWindow : Window

{

private bool \_ShowOwner;

private UsersINFO \_User;

private VehiclesINFO \_Vehicle;

private RentalsINFO \_Rental;

private string connectionString { get; set; } = App.GetConnectionString("CARHANDLERConnection");

public TripWindow(ref UsersINFO user, Window owner, bool showOwner)

{

this.Owner = owner;

this.Owner.Visibility = Visibility.Collapsed;

\_ShowOwner = showOwner;

\_User = user;

InitializeComponent();

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Просматривает {this.Title}", null, LogType.UserAction));

App.AppDataBase.OpenConnection(connectionString);

\_Rental = App.AppDataBase.GetScalarResult<RentalsINFO>(

"SELECT TOP(1) \* FROM RentalsINFO " +

$"WHERE ID\_DriverLicence = {\_User.ID\_DriverLicence} " +

"AND EndTime > GETDATE() AND RentalStatus='стандартная'");

if (\_Rental != null)

{

rt.SetTime(\_Rental.EndTime);

\_Vehicle = App.AppDataBase.GetScalarResult<VehiclesINFO>(

$"SELECT \* FROM VehiclesINFO WHERE ID\_Vehicle = {\_Rental.ID\_Vehicle}");

Card.SetVehicleInfo(\_Vehicle, "Ошибка загрузки данных");

}

else

{

this.Close();

}

App.AppDataBase.CloseConnection();

}

/// <summary>

/// Закрытие окна

/// </summary>

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

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

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Перестал просматривать {this.Title}", null, LogType.UserAction));

if (\_ShowOwner == true)

{

this.Owner.Show();

this.Owner.Visibility = Visibility.Visible;

this.Owner.Activate();

}

else

{

this.Owner?.Close();

}

}

/// <summary>

/// Нажатие кнопки завершения поездки

/// </summary>

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

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

private void FinishTripClick(object sender, RoutedEventArgs e)

{

try

{

var a = MessageBox.Show("Вы уверены что хотите завершить поездку раньше срока?", "Закончить поездку", MessageBoxButton.YesNo);

if (a == MessageBoxResult.Yes)

{

App.AppDataBase.OpenConnection(connectionString);

if(App.AppDataBase.ExecuteNonQuery(

$"UPDATE Rentals SET RentalStatus='досрочная' WHERE ID\_Rental = {\_Rental.ID\_Rental}"))

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Досрочно окончил аренду ТС {\_Rental.ID\_Vehicle}", null, LogType.UserAction));

App.AppDataBase.CloseConnection();

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

App.\_Logger.Log(new LogMessage((ulong)\_User.ID\_User, this.Title, $"Не смог отменить поездку", ex.Message, LogType.ProgramError));

}

}

/// <summary>

/// Нажатие кнопки Произошла авария

/// </summary>

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

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

private void AccidentClick(object sender, RoutedEventArgs e)

{

var accidentWND = new AddAccident(ref \_User, ref \_Rental, this, true);

accidentWND.Activate();

accidentWND.Show();

this.Visibility = Visibility.Collapsed;

}

}

}

И т.д.