МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-47 01 02 «Дизайн электронных и веб-изданий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство «Прокат автомобилей»

Исполнитель

студент 2 курса группы 9 Криштопчик Александра Владимировна

(Ф.И.О.)

Руководитель работы ассистент Гук В.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Белькевич Р.И.

(подпись)

Минск 2022

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc104770914)

[ВВЕДЕНИЕ 4](#_Toc104770915)

[1. Аналитический обзор литературы и формирование требований 5](#_Toc104770916)

[1.1. Анализ прототипов 5](#_Toc104770917)

[1.1.1. МОЙ АВТОПРОКАТ 5](#_Toc104770918)

[1.1.2. Discovercars.com 6](#_Toc104770919)

[1.1.3. Economybookings.com 6](#_Toc104770920)

[1.1.4. Rentalcars.com 7](#_Toc104770921)

[1.1.5. Qeeq.com 8](#_Toc104770922)

[1.2. Формирование требований к проектируемому программному средству 8](#_Toc104770923)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc104770924)

[2.1. Описание средств разработки 9](#_Toc104770925)

[2.1.1. Microsoft Visual Studio 2022 9](#_Toc104770926)

[2.2.2. Программная платформа .NET Framework 6.0 9](#_Toc104770927)

[2.2.3. Язык программирования C# 9](#_Toc104770928)

[2.2.4. Технология WPF 10](#_Toc104770929)

[2.2.5. Расширяемый язык разметки XAML 10](#_Toc104770930)

[2.2.6. Технология Entity Framework 10](#_Toc104770931)

[2.2.7. MS SQL Server 10](#_Toc104770932)

[2.3. Спецификация функциональных требований 10](#_Toc104770933)

[2.3.1. Требования к параметрам технических и программных средств 10](#_Toc104770934)

[2.3.2. Требования к организации входных данных 11](#_Toc104770935)

[2.4. Функциональность программного средства 11](#_Toc104770936)

[3. Проектирование программного средства 12](#_Toc104770937)

[3.1. Модель базы данных 12](#_Toc104770938)

[3.2. Проектирование архитектуры проекта 14](#_Toc104770939)

[3.3. Проектирование последовательностей проекта 15](#_Toc104770940)

[3.4. Взаимоотношения между классами 15](#_Toc104770941)

[3.5. Блок-схемы алгоритмов 16](#_Toc104770942)

[4. Реализация программного средства 17](#_Toc104770943)

[4.1. Описание функций приложения 17](#_Toc104770944)

[4.2. Описание классов и методов программного средства 17](#_Toc104770945)

[4.2.1. Обращение к модели данных 17](#_Toc104770946)

[4.2.2. Выполнение входа 18](#_Toc104770947)

[4.2.3. Регистрация пользователя 18](#_Toc104770948)

[4.2.4. Просмотр каталога 19](#_Toc104770949)

[4.2.5. Бронирование авто 19](#_Toc104770950)

[4.2.6. История прокатов и отмена еще не подтвержденной заявки на бронь автомобиля 19](#_Toc104770951)

[4.2.7. Личный кабинет 20](#_Toc104770952)

[4.2.8. Окно администратора 20](#_Toc104770953)

[5. Тестирование программного средства 21](#_Toc104770954)

[5.1. Тестирование авторизации и регистрации 21](#_Toc104770955)

[5.2. Тестирование поиска и бронирования автомобиля 22](#_Toc104770956)

[5.3. Тестирование изменения информации в личном кабинете 23](#_Toc104770957)

[5.4. Тестирование форм для администратора 24](#_Toc104770958)

[6. Руководство по установке и использованию 26](#_Toc104770959)

[ЗАКЛЮЧЕНИЕ 32](#_Toc104770960)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 33](#_Toc104770961)

[ПРИЛОЖЕНИЕ А 34](#_Toc104770962)

[ПРИЛОЖЕНИЕ Б 35](#_Toc104770963)

# ВВЕДЕНИЕ

Согласно открытым данным сервисы каршеринга развиваются быстрыми темпами. Данной услугой ежедневно пользуются уже тысячи людей, потому что она популярна и всецело выгодна.

Прокат автомобилей в любом городе — это по-настоящему полезный сервис. Такая модель аренды авто удобна для периодического пользования транспортным средством в том случае, когда необходимо сократить расходы на содержание собственного автомобиля. Прокат автомобилей придет на выручку и тем людям, которые не могут определиться с выбором той или иной модели четырехколесного экипажа. В этом случае каршеринг будет своеобразным тест-драйвом: сразу будет понятно, стоит ли приобретать конкретное транспортное средство себе в собственность или нет. В любом случае аренда транспортного средства обойдется гораздо выгоднее, чем заказ такси, а машина всегда будет под рукой.

Таким образом, без лишних переплат и малейших ожиданий есть возможность поехать куда угодно.

Каждая фирма, которая сдает автомобили на прокат, предусматривает приложение с предварительной регистрацией, т.к. личные данные необходимы для получения автомобиля. При использовании десктопного приложения с целью аренды автомобиля есть возможность получить список всех автомобилей с описанием технических данных и цены аренды на каждую модель. Разумеется, автомобили можно не только просто просмотреть, но и оставить заявку на бронь.

Итак, при необходимости свободы передвижения в пространстве, всегда можно оформить прокат автомобиля, сделав заявку на любой день. С той целью и создано программное средство по прокату автомобилей.

Целью курсового проекта является: развитие практических навыков вёрстки приложений на базе WPF (Windows Presentation Foundation) с использованием языка разметки — XAML (eXtensible Application Markup Language), и языка программирования — С#; разработка дизайна программного средства.

При выполнении курсового проекта будут использованы принципы и приемы ООП и база данных MS SQL Server.

Курсовой проект включает в себя разделы: введение, содержание, аналитический обзор литературы и формирование требований, анализ требований к программному средству и разработка функциональных требований, проектирование программного средства, реализация программного средства, тестирование программного средства, руководство по установке и использованию, заключение, список использованной литературы.

# Аналитический обзор литературы и формирование требований

## Анализ прототипов

Для разработки программного средства по прокату автомобилей необходимо проанализировать уже существующие аналоги в данной сфере. Анализ достоинств и недостатков уже существующих решений поможет сформировать более четкое представление о том, как должно выглядеть программное средство с учетом существующих разработок и внесением улучшений или изменений, а также сформулировать требования к данному программному средству, которое представляет собой настольное приложение.

### МОЙ АВТОПРОКАТ

Одним из альтернативных решений является интернет-ресурс «МОЙ АВТОПРОКАТ». Главными преимуществами данного ресурса является подбор автомобиля по любой цене, а также наличие системы скидок. Пользователь данного ресурса имеет возможность подбирать автомобиль по фильтрам, а также по периоду времени, в течение которого ему необходим автомобиль. Также осуществляется круглосуточная поддержка связи с администратором в режиме онлайн в случае наличия соответствующих вопросов касательно подбора авто.

Из выявленных недостатков данного ресурса является отсутствие наиболее детальной информации об автомобилях. Также можно отметить, что интерфейс не позиционирует себя как интуитивно понятный, т.к. присутствует избыточность информации, что делает затруднительным процесс оформления заявки на бронь.

Интерфейс интернет-ресурса представлен на рисунке 1.1.



Рисунок 1.1 – Интернет-ресурс «МОЙ АВТОПРОКАТ»

### Discovercars.com

Еще одним аналогом для анализа является интернет-ресурс «Discovercars.com». На сайте очень удобная форма поиска. Она помогает отсортировать предложения по цене или классу автомобиля, а также выбрать нужные опции: аренда без водителя, на сутки или двое, бюджетные недорогие авто или премиум класс. Ознакомиться с рейтингом прокатчиков можно уже на этапе выбора, что позволит сразу оставить только надежные фирмы.

Недостатков при анализе выявлено не было.

Интерфейс интернет-ресурса представлен на рисунке 1.2.

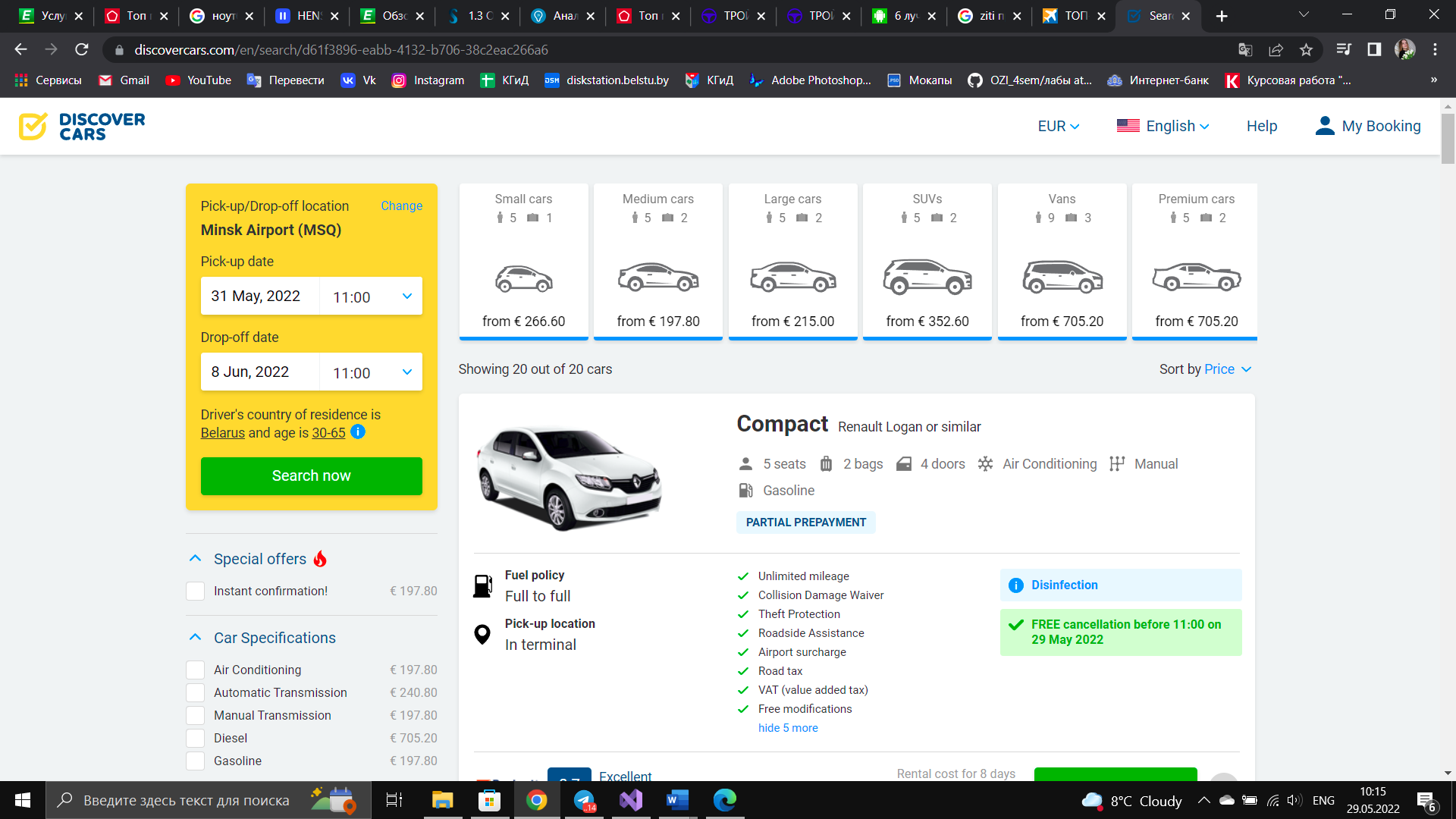


Рисунок 1.2 — Интернет-ресурс «Discovercars.com»

### Economybookings.com

Еще одним альтернативным решением является интернет-ресурс «Economybookimgs.com». У данного программного средства есть возможность аренды авто не только по дням, но и по часам. Преимуществом является наличие интуитивно понятного интерфейса без избыточности информации, как это было ранее в одном из интернет-ресурсов.

Также пользователь может ознакомиться более подробно с условиями аренды сделав клик по соответствующей кнопке, а также получить рейтинг компании, предоставляющей данный автомобиль, в рамках рассматриваемого интернет-ресурса.

Из недостатков было обнаружено наличие функции «Сохранить», которая подразумевает отправку понравившейся модели авто пользователю на почту. Целесообразнее в данном случае было бы создать раздел «Избранное», куда бы пользователь также имел возможность добавлять понравившуюся модель авто, и просматривать этот список находясь непосредственно в интернет-ресурсе, а не покидая его с переходом на почту.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

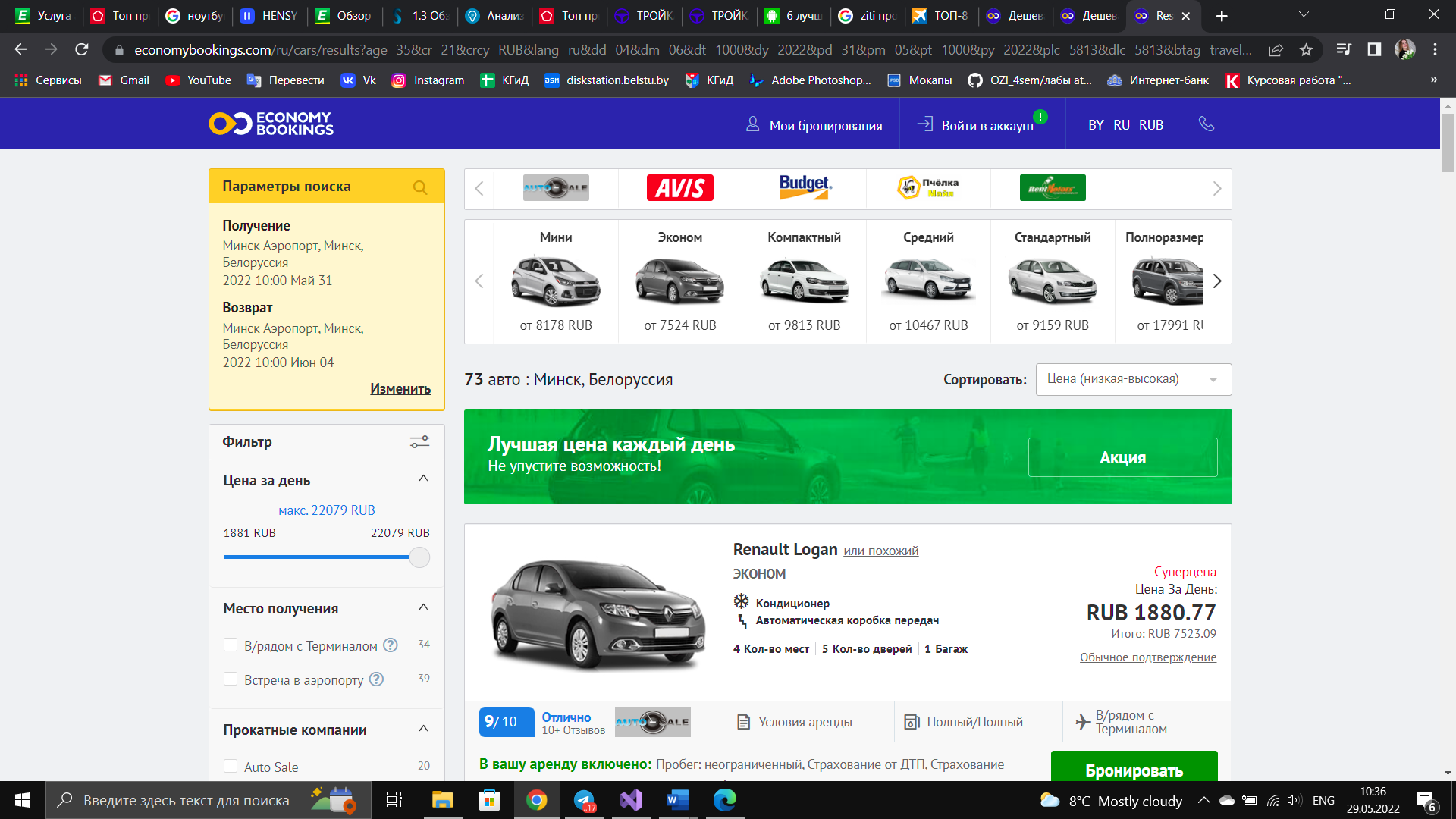


Рисунок 1.3 — Интернет-ресурс «Economybookings.com»

### Rentalcars.com

Также в качестве альтернативного решения можно рассмотреть интернет-ресурс «Rentalcars.com», достоинством которого является интуитивно понятный интерфейс. Выбрать автомобиль поможет наличие отзывов автолюбителей, которые уже воспользовались услугами того или иного прокатчика. При выборе авто есть возможность задавать фильтрацию для поиска, а также просматривать детальную информацию.

Недостатком является практически полное соответствие интерфейса у рассматриваемого интернет-сервиса с интернет-сервисом «Discovercars.com».

Интерфейс интернет-ресурса представлен на рисунке 1.4.

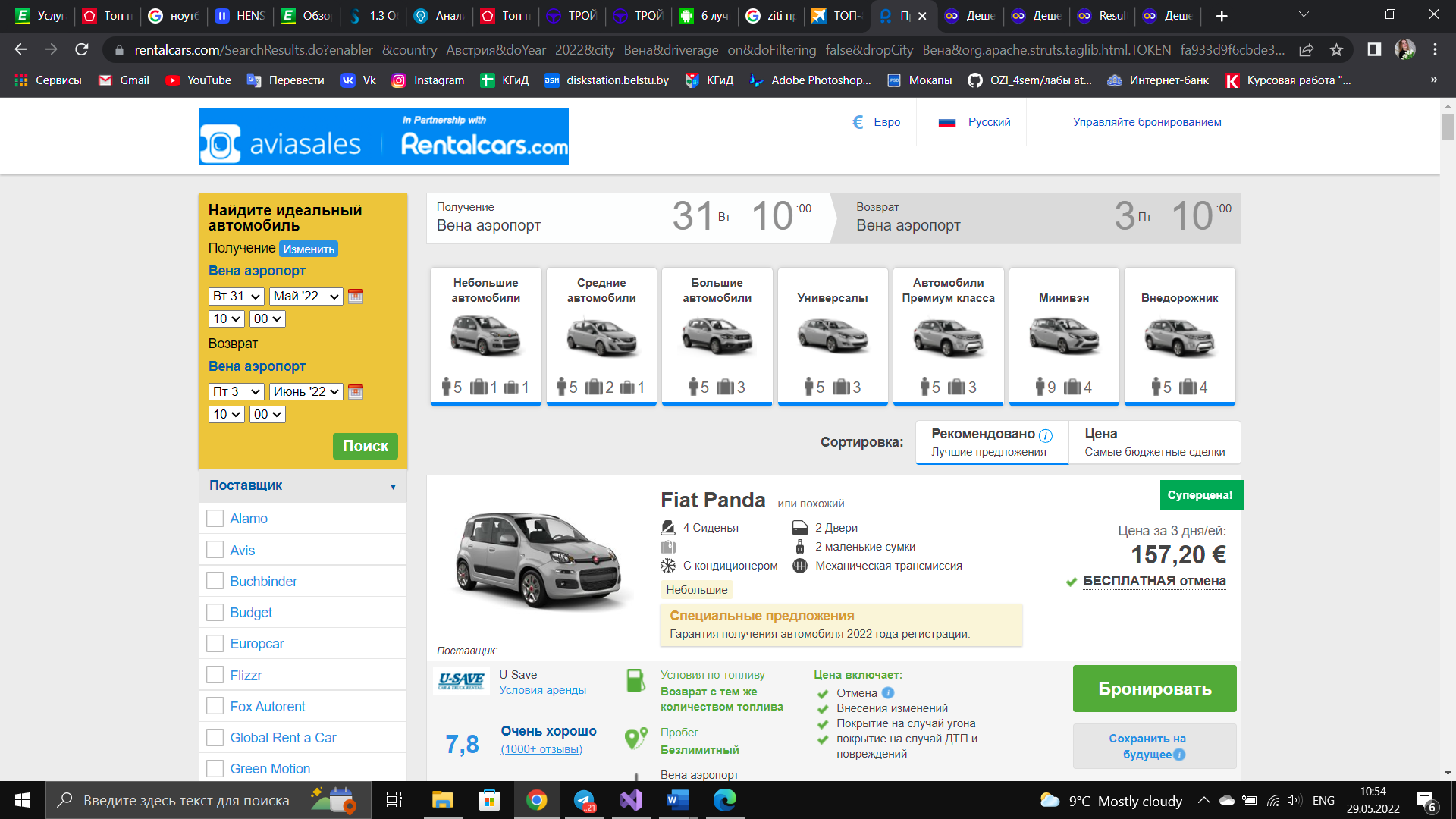


Рисунок 1.4 — Интернет-ресурс «Economybookings.com»

### Qeeq.com

В перечень рассматриваемых аналогов также можно включить интернет-ресурс «Qeeq.com». Из достоинств следует отметить отсутствие в дизайне интерфейса ярких, «кричащих» цветов, отвлекающих внимание. Есть возможность сортировки автомобилей по выбранным параметрам, а также наличие рейтинговой системы.

Из недостатков можно выделить неполное описание каждой модели автомобиля.

Интерфейс интернет-ресурса представлен на рисунке 1.5.



Рисунок 1.5 — Интернет-ресурс «Qeeq.com»

## Формирование требований к проектируемому программному средству

Проанализировав достоинства и недостатки вышеперечисленных аналогов, можно сформировать требования, предъявляемые к проектируемому программному средству. Таким образом, программное средство должно поддерживать следующие нижеперечисленные функции:

* управление администратором базой данных;
* уведомление администратора о поступлении новой заявки на бронь;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность просмотра каталога автомобилей;
* возможность поиска подходящего авто по заданным параметрам
* возможность оформления заявки на бронь;
* возможность пользователя просматривать информацию о всех его прокатах;
* возможность просмотра личной информации пользователя и ее изменение.

# Анализ требований к программному средству и разработка функциональных требований

## Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Framework 6.0;
* язык программирования C#;
* технология WPF;
* расширяемый язык разметки XAML;
* технология Entity Framework;
* MS SQL Server.

### Microsoft Visual Studio 2022

Microsoft Visual Studio 2022 — это интегрированная среда разработки для написания, отладки и сборки кода, а также последующей публикации приложений для ОС Windows, как консольных, так и десктопных с графическим интерфейсом, созданным с использованием таких технологий, как WinForms или WPF.Функциональная структура среды включает в себя: редактор исходного кода, который включает множество дополнительных функций, как автодополнение IntelliSense, рефакторинг кода и т. д.; отладчик кода; редактор форм, предназначенный для упрощённого конструирования графических интерфейсов; веб-редактор; дизайнер классов; дизайнер схем баз данных [1].

### Программная платформа .NET Framework 6.0

Платформа .NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году, которая поддерживает создание и выполнение веб-служб и приложений Windows, обеспечивающая согласованную объектно-ориентированной среду программирования для локального сохранения и выполнения объектного кода, локального выполнения кода, распределенного в Интернете, либо для его удаленного выполнения. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования [2].

### Язык программирования C#

В качестве языка программирования используется C# – это объектно-ориентированный язык программирования. Он поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в то же время гибких, масштабируемых и расширяемых приложений. С# продолжает активно развиваться и с каждой новой версией появляется все больше интересных функциональностей. Используется как основной язык в технологии WPF [3].

### Технология WPF

Технология Windows Presentation Foundation (WPF) — это часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов. Идея, лежащая в основе введения инфраструктуры WPF, заключалась в том, чтобы пользователь мог разработать богатый пользовательский интерфейс, состоящий из 3D, анимации, насыщенного цвета с минимальной сложностью кода. WPF можно рассматривать как эволюцию над WinForms. Он производит простую в использовании графическую систему отображения для окон. Он предлагает простое разделение между пользовательским интерфейсом и бизнес-логикой [4].

### Расширяемый язык разметки XAML

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. Применительно к WPF данный язык используется прежде всего для создания пользовательского интерфейса декларативным путем.

### Технология Entity Framework

Для взаимодействия с базой данных в приложении WPF используется технология Entity Framework. Она представляет собой специальную объектно-ориентированную технологию доступа к данным на базе фреймворка .NET. Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL [6].

### MS SQL Server

Для управления базами данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта. Основной используемый язык запросов — это Transact-SQL [5].

## Спецификация функциональных требований

### Требования к параметрам технических и программных средств

Программное средство может функционировать на ЭВМ со следующими минимальными характеристиками:

1. процессор Intel® Core™ i3 с тактовой частотой 2.4 ГГц и лучше;
2. оперативная память 256 МБ и более;
3. операционная система: Windows 10.

### Требования к организации входных данных

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с клавиатуры текста: логин, пароль. После аутентификации пользователю предоставляется возможность работы с системой.

Данные, вводимые пользователем, должны проверяться на корректность в процессе аутентификации.

## Функциональность программного средства

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* просмотр каталога с автомобилями;
* поиск автомобилей по заданным параметрам;
* оформление заявки на бронь автомобиля;
* отмена не подтвержденной заявки на бронь;
* просмотр информации о данных пользователя и об истории прокатов.

Для администратора:

* авторизация;
* получение письма на почте о поступившей новой заявке на бронь;
* просмотр поступивших новых заявок;
* изменение статуса заявки;
* просмотр доступных автомобилей для проката;
* загрузка изображения для автомобиля;
* редактирование стоимости проката автомобиля;
* добавление/удаление промокодов;
* просмотр информации о клиентах;
* просмотр всех ДТП, относящихся к заданному клиенту.

# Проектирование программного средства

## Модель базы данных

Для функциональности ПС необходимо создание базы данных для хранения информации приложения, поэтому для данного курсового проекта была спроектирована база данных, которая получила название CAR\_RENT. Для управления базой данных используется Microsoft SQL Server. Диаграмма базы данных изображена на рисунке 3.1.

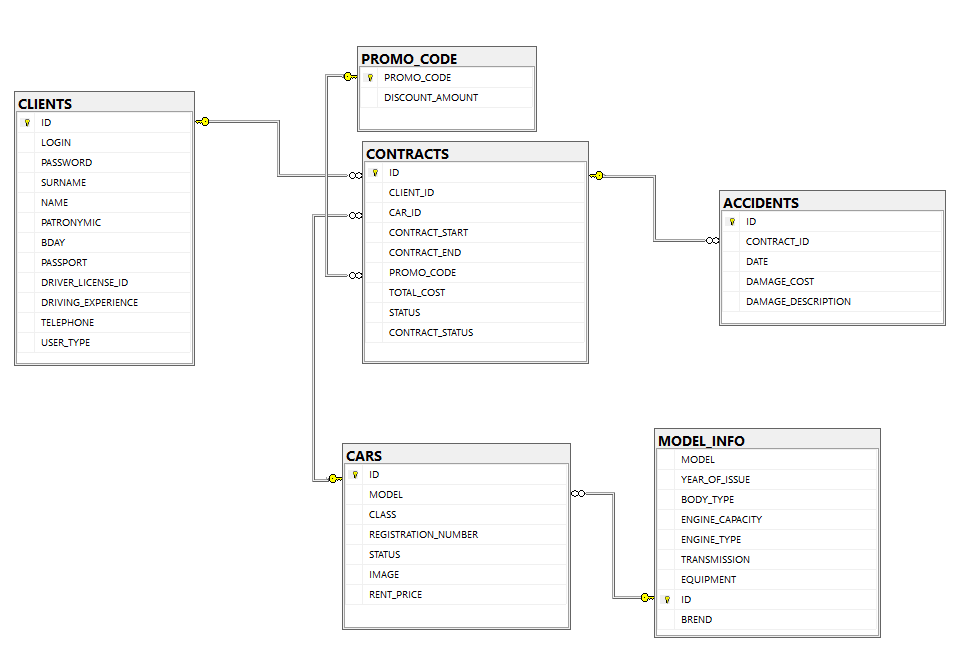


Рисунок 3.1 —Диаграмма базы данных

Всего в базе данных 6 таблиц. В таблице CLIENTS хранятся все пользователи, зарегистрированные в приложении, CARS – все автомобили, MODEL\_INFO – более детальная информация для конкретного автомобиля, PROMO\_CODE – промокоды, ACCIDENTS – ДТП, CONTRACTS – контракты.

Таблица **CLIENTS** состоит из 12 столбцов:

* ID – первичный ключ таблицы;
* LOGIN – логин пользователя;
* PASSWORD – пароль пользователя;
* SURNAME – фамилия пользователя;
* NAME ­– имя пользователя;
* PATRONYMIC ­– отчество пользователя;
* BDAY­ – дата рождения пользователя;
* PASSPORT – серия и номер паспорта;
* DRIVER\_LICENSE\_ID – номер водительского удостоверения;
* DRIVING\_EXPERIENCE – опыт вождения;
* TELEPHONE – мобильный номер телефона;
* USER\_TYPE – тип пользователя (0 – пользователь, 1 – администратор).

Таблица **MODEL\_INFO** состоит из 9 столбцов:

* ID – ключевое поле таблицы;
* BREND – марка автомобиля; ­­
* MODEL – модель автомобиля;
* YEAR\_OF\_ISSUE – год выпуска автомобиля;
* BODY\_TYPE – тип кузова;
* ENGINE\_CAPACITY– объем двигателя;
* ENGINE\_TYPE – тип двигателя;
* TRANSMISSION ­– трансмиссия;
* EQUIPMENT – оборудование.

Таблица **CARS** состоит из 7 столбцов:

* ID – ключевое поле таблиц;
* MODEL – вторичный ключ; поле, ссылающееся на поле ID таблицы «MODEL\_INFO»;
* CLASS – класс автомобиля;
* REGISTRATION\_NUMBER – регистрационный номер автомобиля;
* STATUS ­– статус автомобиля;
* IMAGE – фото автомобиля;
* RENT\_PRICE – стоимость проката автомобиля.

Таблица **PROMO\_CODE** состоит из 2 столбцов:

* PROMO\_CODE – ключевое поле таблицы с наименованием промокода;
* DISCOUNT\_AMOUNT – размер скидки.

Таблица **CONTRACTS** состоит из 2 столбцов:

* ID – ключевое поле таблицы;
* CLIENT\_ID – вторичный ключ; поле, ссылающееся на поле ID таблицы «CLIENTS»;
* CAR\_ID – вторичный ключ; поле, ссылающееся на поле ID таблицы «CARS»;
* CONTRACT\_START – дата начала аренды автомобиля;
* CONTRACT\_END – дата окончания аренды автомобиля;
* PROMO\_CODE – вторичный ключ; поле, ссылающееся на поле ID таблицы «PROMO\_CODE»;
* TOTAL\_COST – итоговая стоимость проката;
* STATUS – статус заявки;
* CONTRACT\_STATUS – статус контракта.

Таблица **ACCIDENTS** состоит из 5 столбцов:

* ID – ключевое поле таблицы;
* CONTRACT\_ID – вторичный ключ; поле, ссылающееся на поле ID таблицы «CONTRACTS»; ­­
* DATE – дата ДТП;
* DAMAGE\_COST – величина ущерба;
* DAMAGE\_DESCRIPTION – описание повреждения;

## Проектирование архитектуры проекта

Для общего представления функционала, доступного каждой группе пользователей, используется диаграмма вариантов использования. На ней отображаются два типа основных сущностей: варианты использования и группы пользователей.

Группы пользователей в диаграмме - это актеры. Они обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействия с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена на рисунке 3.2.

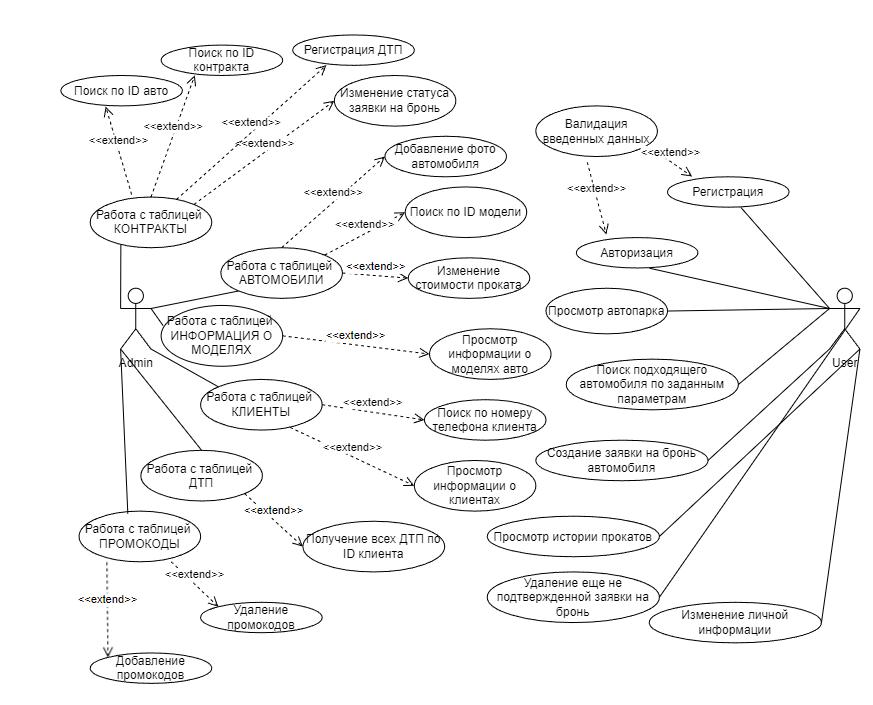


Рисунок 3.2 — Диаграмма использования приложения

Опираясь на данную диаграмму вариантов использования, опишем функционал использования: точкой входа в приложение является окно регистрации или авторизации. Во время ввода данных обязательно происходит валидация введенных данных. Кроме пользователя в диаграмме присутствует администратор, который работает с данными, находящимися в таблицах, где он может производить поиск по определенным полям, менять статус заявки на бронь, оформлять ДТП, менять стоимость проката авто, загружать фото авто. Сам же пользователь может проводить поиск, оформлять заявку на бронь или отменять ее, если она еще не подтверждена, просматривать историю прокатов, просматривать информацию в личном кабинете и редактировать ее при необходимости.

## Проектирование последовательностей проекта

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Такая диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Диаграмма последовательностей для поиска и бронирования подходящего автомобиля по заданным параметрам представлена на рис. 3.3.

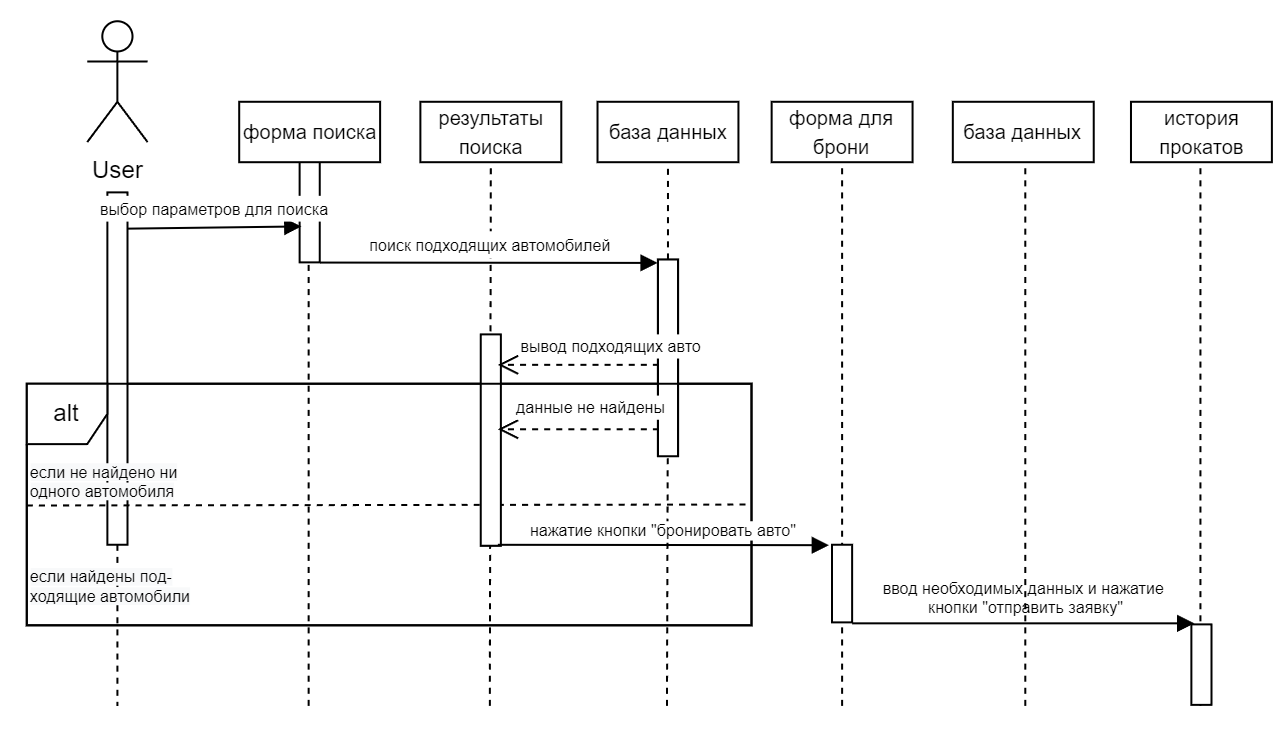


Рисунок 3.3 — Диаграмма последовательностей для оформления брони

## Взаимоотношения между классами

Диаграмма классов иллюстрирует структуру иерархии классов системы, их атрибутов, методов, интерфейсов и взаимосвязей между ними. Пример диаграммы классов представлен на рисунке 3.4.

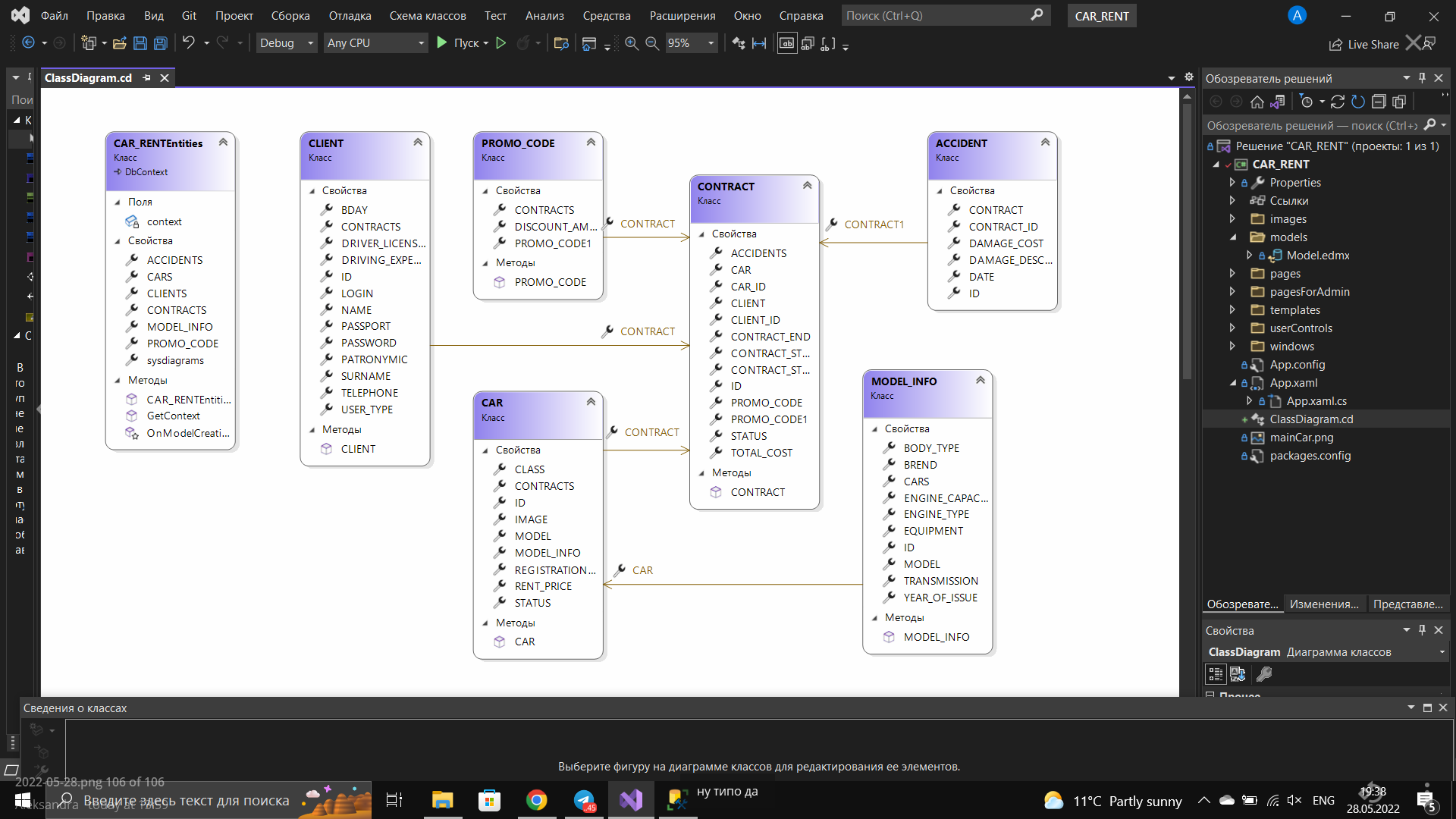


Рисунок 3.4 — Диаграмма классов

## Блок-схемы алгоритмов

Блок-схемы — это тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности.

Блок схемы являются важными декларативными и описательными элементами, которые позволяют в стандартизированной форме представить алгоритм работы программы, ее части или любого другого процесса. Пример блок-схемы для процесса регистрации представлен в приложении А.

# Реализация программного средства

## Описание функций приложения

Приложение имеет следующие функции:

* Регистрация пользователя;
* Авторизация пользователя;
* Просмотр каталога с автомобилями;
* Возможность поиска автомобиля по заданным параметрам;
* Отправка заявки на бронь автомобиля;
* Возможность использования промокода для получения определенной скидки;
* Возможность просмотра историю прокатов;
* Отмена еще не подтвержденной заявки на бронь;
* Редактирование личной информации.

## Описание классов и методов программного средства

### Обращение к модели данных

Для связи приложения с базой данных в файле «App.config»в секции <connectionStrings> автоматически создаётся строка соединения с базой данных на SQL Server (Листинг 4.1).

|  |
| --- |
| <connectionStrings>  <add name="CAR\_RENTEntities" connectionString="metadata=res://\*/models.Model.csdl|res://\*/models.Model.ssdl|res://\*/models.Model.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=WIN-4K4KDT4B1EH;initial catalog=CAR\_RENT;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />  </connectionStrings> |

Листинг 4.1 – Подключение к базе данных

Далее автоматически создаются сущности, которые представляют собой классы C#. Каждой таблице базы данных соответствует определенный класс (Рисунок 4.2).

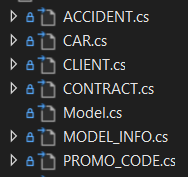


Рисунок 4.1 – Сущности базы данных

Для подключения и работы с информацией из базы данных также создается контекст данных, который представляет собой класс, унаследованный от DbContext. Для обращения к модели данных и последующего использования используется паттерн SingleTon (Листинг 4.2).

|  |
| --- |
| private static CAR\_RENTEntities context;  public static CAR\_RENTEntities GetContext()  {  if (context == null)  context = new CAR\_RENTEntities();  return context;  } |

Рисунок 4.2 – Контекст данных

Всё создается автоматически благодаря тому, что в проекте использовался принцип DataBaseFirst, для добавления уже существующей базы данных.

### Выполнение входа

Необходимый функционал для входа пользователя в аккаунт, а также для валидации введенных данных, реализован в классе LogPage.

Основным методом класса является метод Come\_Click (), выполняющийся при нажатии пользователя на кнопку «Войти». При успешной валидации введённых данных выполняется проверка на существование введённого логина в базе данных, и при нахождении значения выполняется проверка на соответствие введённого пароля и хранящегося в базе данных пароля. При успешном прохождении всех проверок пользователь перенаправляется на главную страницу каталога, в противном случае – получает сообщение о неправильном пароле, т.к. при вводе данных в область с логином пользователь также получает сообщение о том, найден пользователь с таким логином или нет.

Если необходимо зарегистрироваться, то при вызове метода Hyperlink\_Click(), происходит переход на первую страницу с регистрацией.

Листинг реализации методов Come\_Click() и Hyperlink\_Click() представлены в приложении Б.

### Регистрация пользователя

Для регистрации пользователя реализованы два класса. В процессе заполнения полей информацией происходит и валидация, т.к. она выполняется при событии SelectionChanged(). Если введенные данные не соответствуют предъявленным требованиям, то пользователь не информируется соответствующими «галочками» об успешно введенных данных, а также у пользователя отсутствует возможность перейти на следующую страницу и продолжить процесс авторизации.

В случае успешного прохождения валидации на первой странице регистрации и наличия всех «галочек» пользователь, нажимая на стрелку «далее», попадает на новую страницу и продолжает процесс регистрации с теми же условиями, что и на первой странице.

В конечном итоге, когда валидация пройдена, пользователь также кликает по кнопке «далее» и при успешном внесении данных в базу данных информируется соответствующим окном.

Листинг реализации процесса регистрации представлен в приложении Б.

### Просмотр каталога

После успешной авторизации или регистрации пользователь направляется на основную страницу программного средства, представленную классом CatalogPage, позволяющую перейти к просмотру либо всех автомобилей, либо относящихся к определенному классу, выборка которых происходит при выборе одного из RadioButton и вызове метода Search().

При нажатии на кнопку «Бронирование авто» на одном из автомобилей, представленным классом Car, вызывается метод Rent(), который получает данные о том, какой именно автомобиль выбрал пользователя. После чего пользователь перенаправляется на новое окно.

Листинг метода Search() и класс Car представлены в приложении Б.

### Бронирование авто

После выбора понравившегося автомобиля при нажатии на кнопку «Бронирование авто» на одном из автомобилей вызывается метод Rent(), реализованный в классе Car(), который получает данные о том, какой именно автомобиль выбрал пользователь. В этом же методе вызывается окно, которое также получает данные о том, какой именно это автомобиль. По нажатию кнопки «Отправить заявку» вызывается метод send\_Click() класса rent(), который как отправляет письмо администратору о том, что поступила новая заявка, так и сохраняет новый контракт в базу данных.

Листинг реализации методов Rent() и send\_Click() представлены в приложении Б.

### История прокатов и отмена еще не подтвержденной заявки на бронь автомобиля

После завершения процедуры бронирования происходит перенаправление пользователя на страницу, где размещена история прокатов. Заполнение списка с перечнем прокатов, содержащих также и статус, происходит при инициализации с помощью пользовательского элемента управления HistoryContract, где в зависимости от статуса заявки, а также от периода аренды автомобиля отображается статус проката. Если пользователь хочет отменить заявку, то при нажатии по кнопке «Отменить заявку» вызывается метод cancel\_Click(), после чего статус проката отображается как «Прокат отменен».

Листинг реализации класса HistoryContract представлен в приложении Б.

### Личный кабинет

Персональная страница, которая позволяет просмотреть данные пользователя, изменить их, а также изменить и пароль, функционирует при помощи класса UserInfoPage. Данный класс содержит методы с предварительной валидацией для внесения изменений в личные данные, которые были уже использованы и ранее в регистрации пользователей.

### Окно администратора

В окно администратора пользователь перенаправляется из окна входа при соответствующем значении поля USER\_TYPE. Данное окно представлено классом AdminWindow и содержит в себе центральный фрейм и панель навигации между таблицами, которая представлена контейнером компоновки StackPanel, содержащем в себе навигационные кнопки, которые осуществляют переход по необходимым страницам: контракты, автомобили, информация о моделях, клиенты, ДТП, промокоды. С помощью центрального фрейма происходит постраничная навигация.

Все методы по поиску каких-либо данных, основаны на одном и том же принципе, где в DataGrid перебираются записи на соответствие значений из определенного столбца тому значению, которое записывается в поле ввода. Методы вызываются либо при нажатии на кнопку «Найти», либо, где отсутствует данная кнопка, при срабатывании события TextChanged().

Метод Edit\_Click() используется для внесения изменений статуса заявки только в том случае, если запись из DataGrid выбрана, в противном случае появляется соответствующее сообщение об ошибке. На том же принципе основаны функции изменения стоимости проката и загрузки фото.

Метод Add\_Click() запускается при нажатии на кнопку «Добавить» на странице с промокодами. Если поля не заполнены или такой промокод уже существует, то администратор получает соответствующее сообщение об ошибке.

Метод Delete\_Click() срабатывает при нажатии на кнопку «Удалить» на странице с промокодами и обращается к базе данных, удаляя из соответствующей таблицы выбранный фильм. Если промокод не был выбран в списке, то администратор получает сообщение об ошибке.

Листинг реализации функционала администратора представлен в приложении Б.

# Тестирование программного средства

Тестирование программного средства является необходимым этапом при его разработке. На этом этапе тестировщик предусматривает все возможные ошибки и неточности, с которыми может столкнуться пользователь во время использования программного средства, с целью их дальнейшего устранения.

## Тестирование авторизации и регистрации

В момент авторизации или регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, несуществующий логин, неверный или некорректный пароль, некорректные номер телефона, номер паспорта, номер водительского удостоверения и т. д. Такие исключения предусматриваются и обрабатываются программным средством. Для того, чтобы пользователю было понятно в каком поле были введены неправильные или некорректные данные, в окне авторизации выводятся соответствующие подсказки, а в процессе авторизации всплывают подсказки для корректного ввода информации и если введенные данные корректны, то появляются «галочки», свидетельствующие об успешном завершении процесса валидации. Примеры обработки разных исключительных ситуаций представлены на рисунках 5.1 – 5.3.

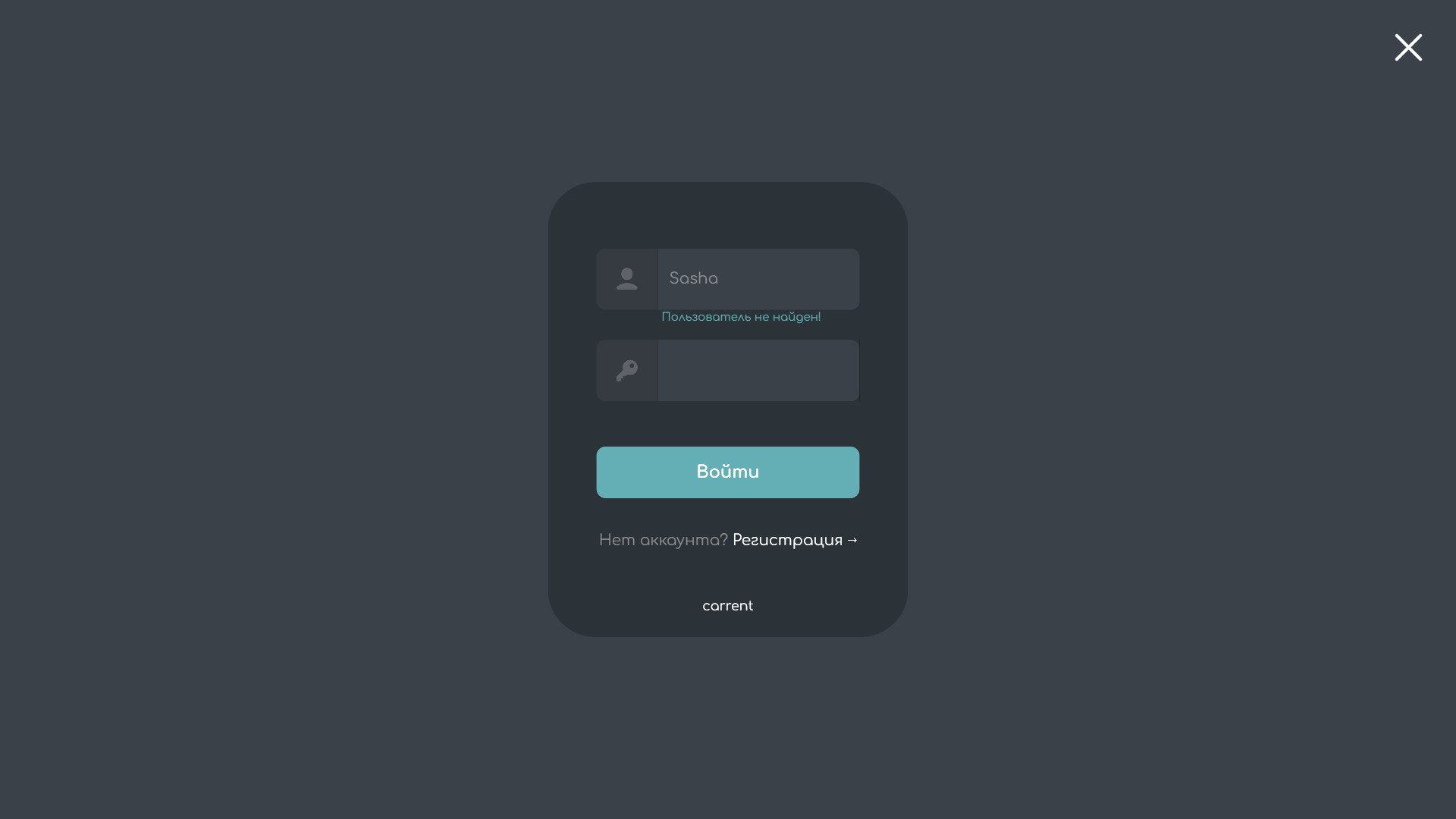


Рисунок 5.1 — Обработка несуществующего пользователя

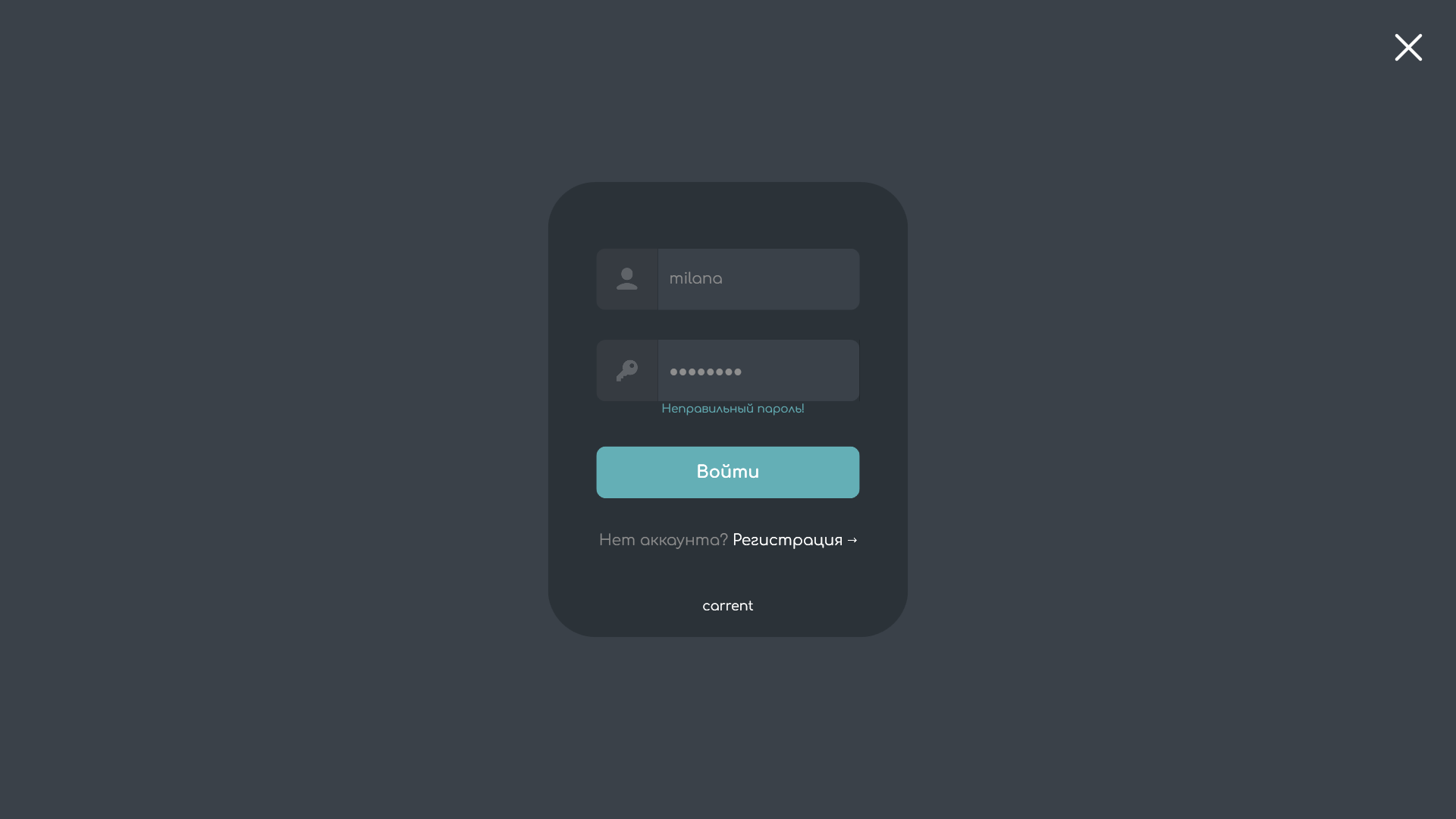


Рисунок 5.2 — Обработка неправильного пароля

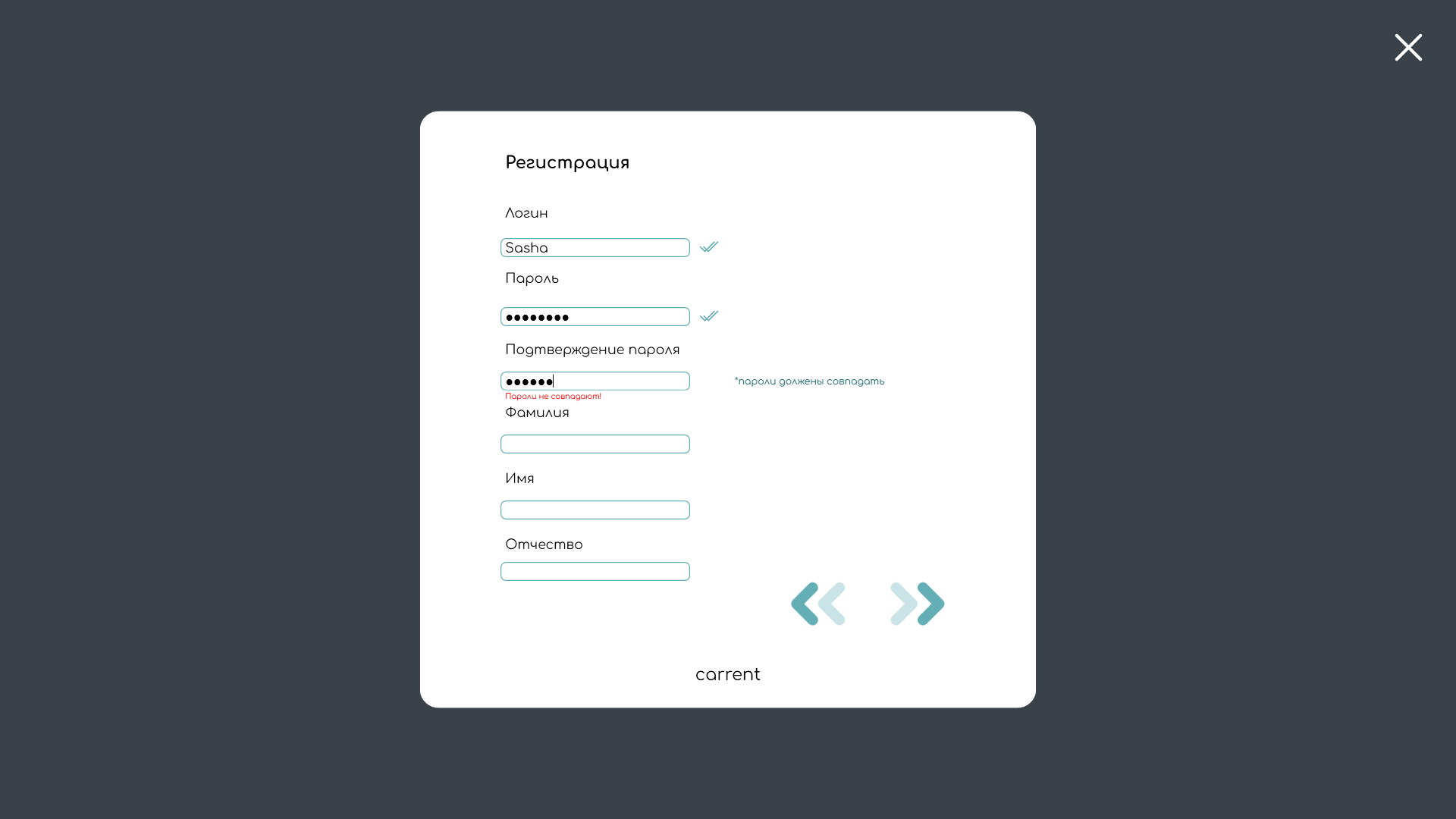


Рисунок 5.3 — Обработка корректно введенных данных, несовпадающих паролей, а также всплывающие подсказки

## Тестирование поиска и бронирования автомобиля

На странице каталога с поиском и бронированием автомобиля также могут возникнуть ошибки, например, если при поиске автомобиля с заданными параметрами не найдено ни одного автомобиля, или если при оформлении заявки на бронь пользователь вводит некорректные даты или несуществующий промокод. Эти исключения будут обработаны и пользователю выведется сообщение об ошибке. Примеры обработки исключений представлены на рисунках 5.4 – 5.5.

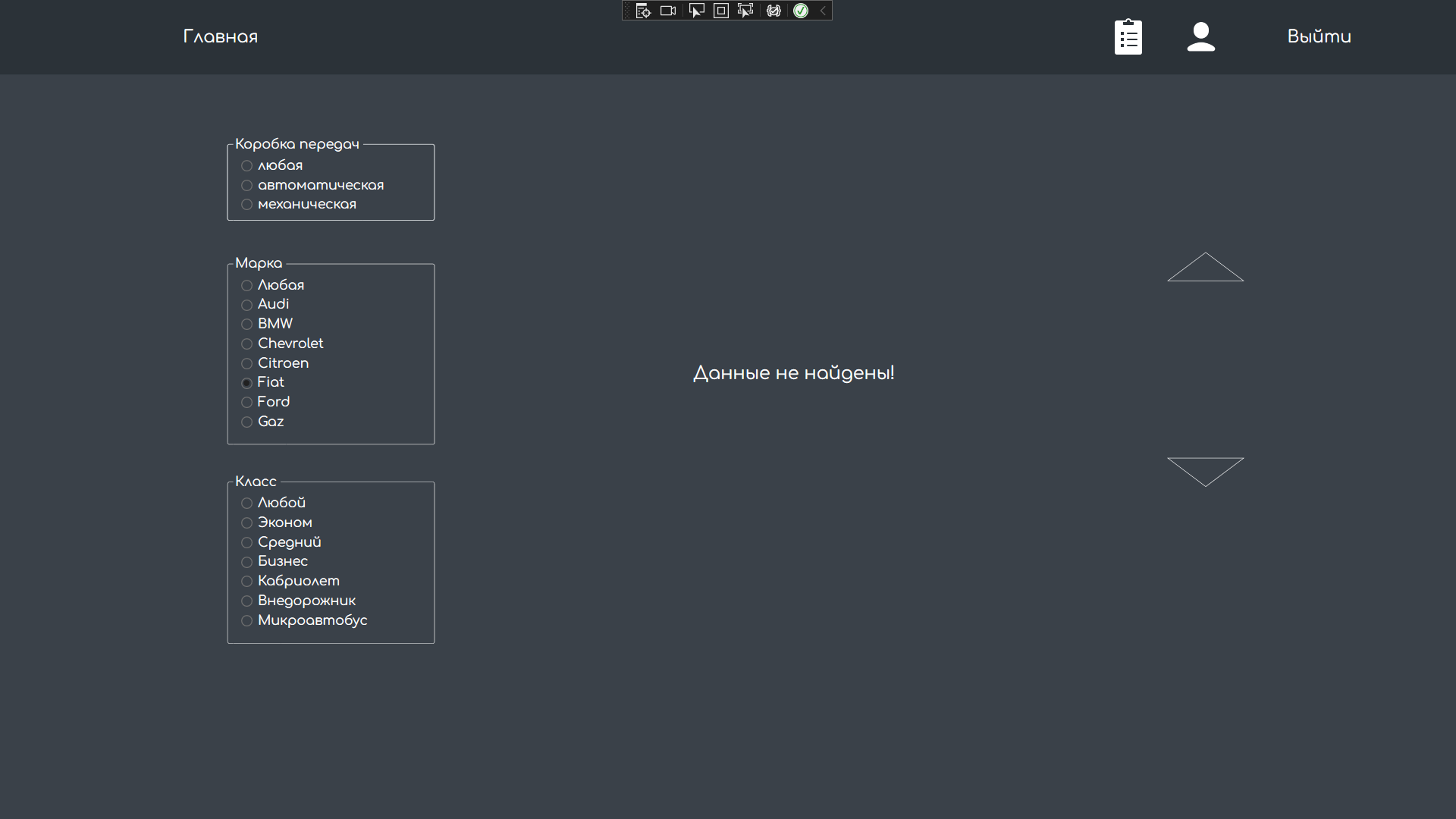


Рисунок 5.4 — Обработка пустого результата поиска

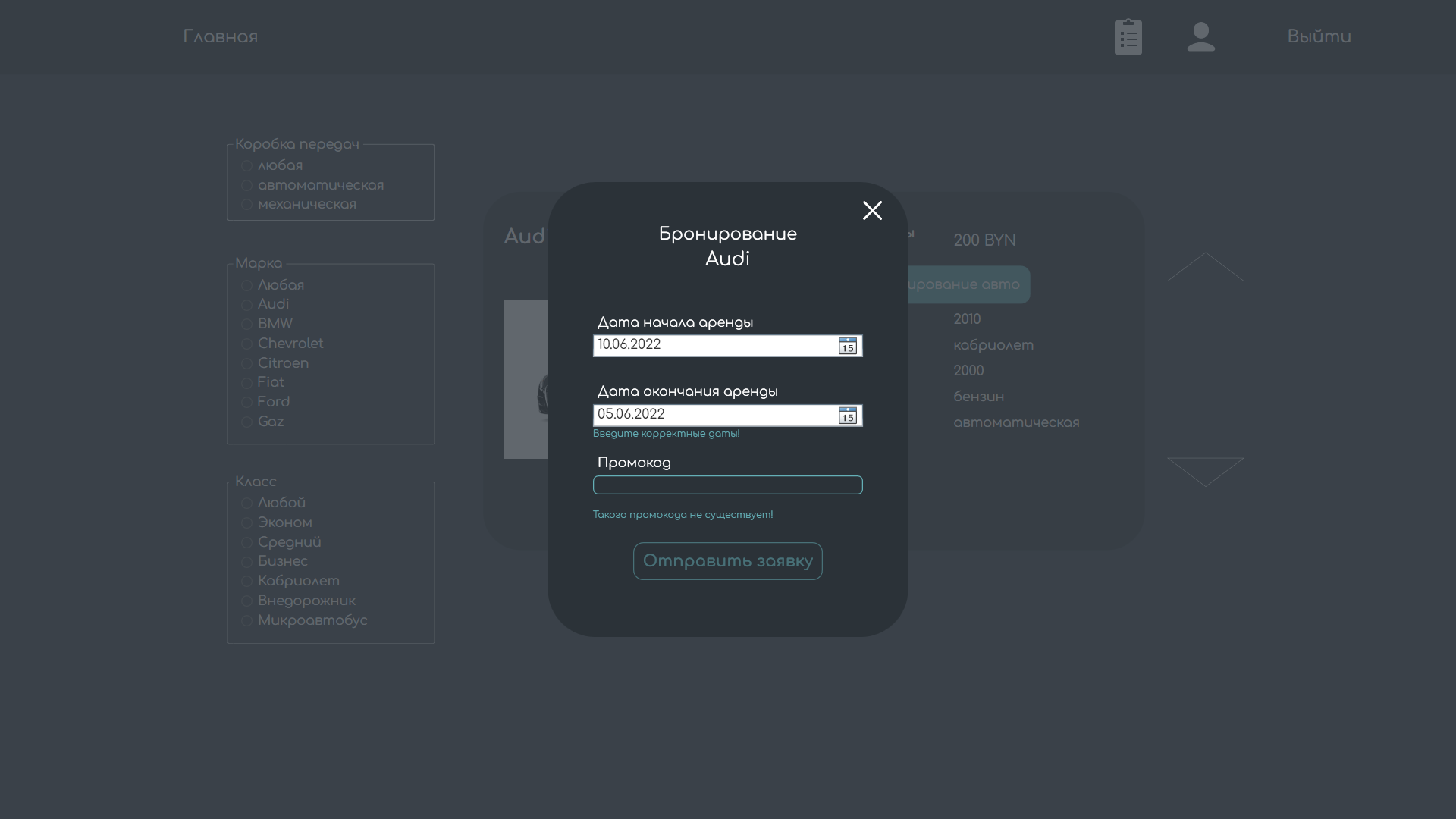


Рисунок 5.5 — Обработка некорректно введенных данных и отсутствие введенного промокода

## Тестирование изменения информации в личном кабинете

Форма в личном кабинете имеет текстовые поля, поэтому необходимо проверять введённую информацию на корректность. При вводе некорректных данных личная информация не будет изменена, а исключения будут обработаны и пользователю высветятся соответствующие подсказки, как на рисунке 5.6.

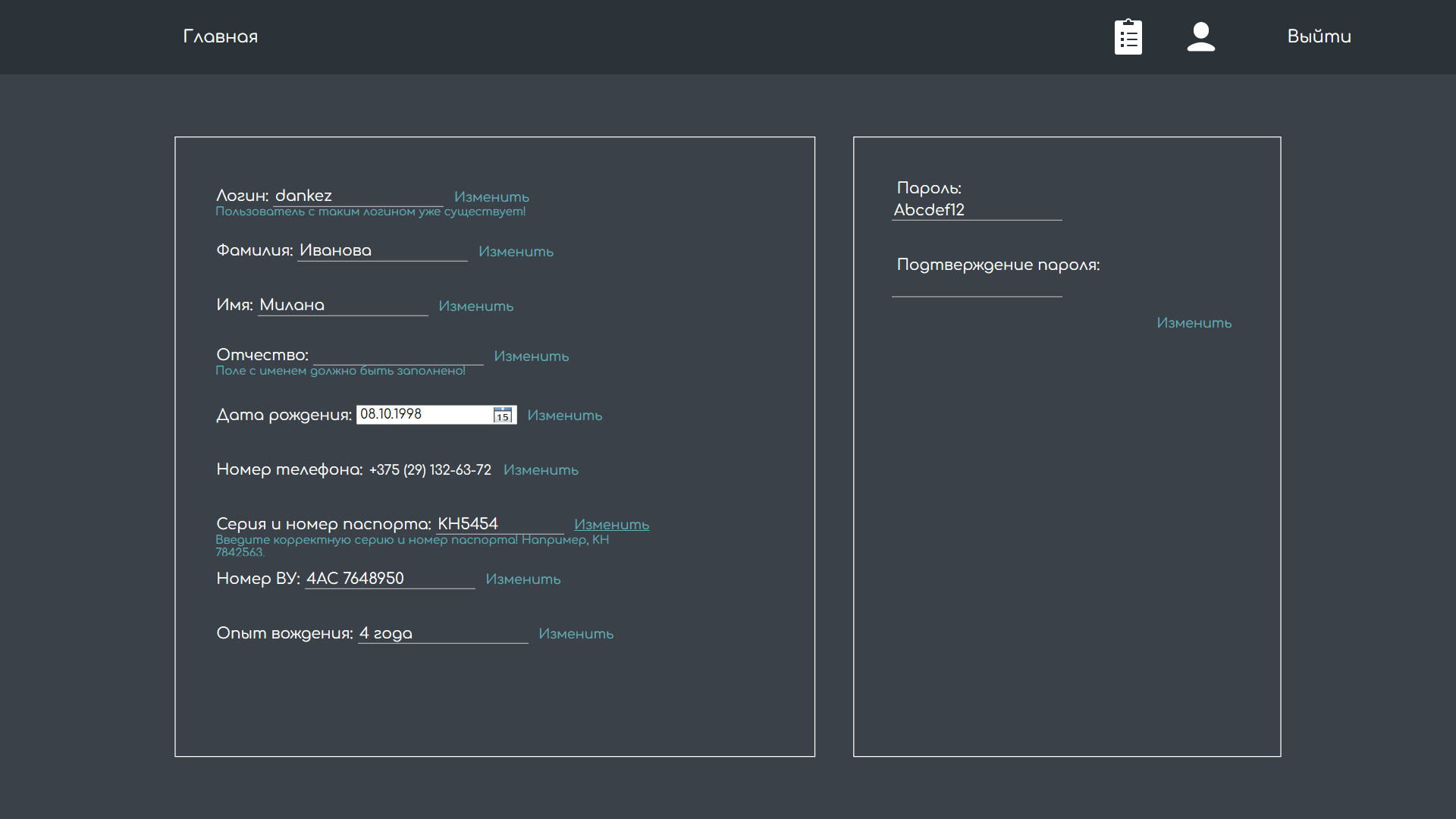


Рисунок 5.6 — Обработка некорректных данных при изменении

## Тестирование форм для администратора

При работе с таблицами у администратора возможно генерирование таких исключительных ситуаций, как: нажатие на кнопку «Удалить» или «Редактировать» при отсутствии выбранной записи. Для всех вкладок, с которыми работает администратор, вышеописанные исключительные ситуации генерируют всплывающее окно с текстом ошибки, представленные на рисунках 5.7 – 5.8.

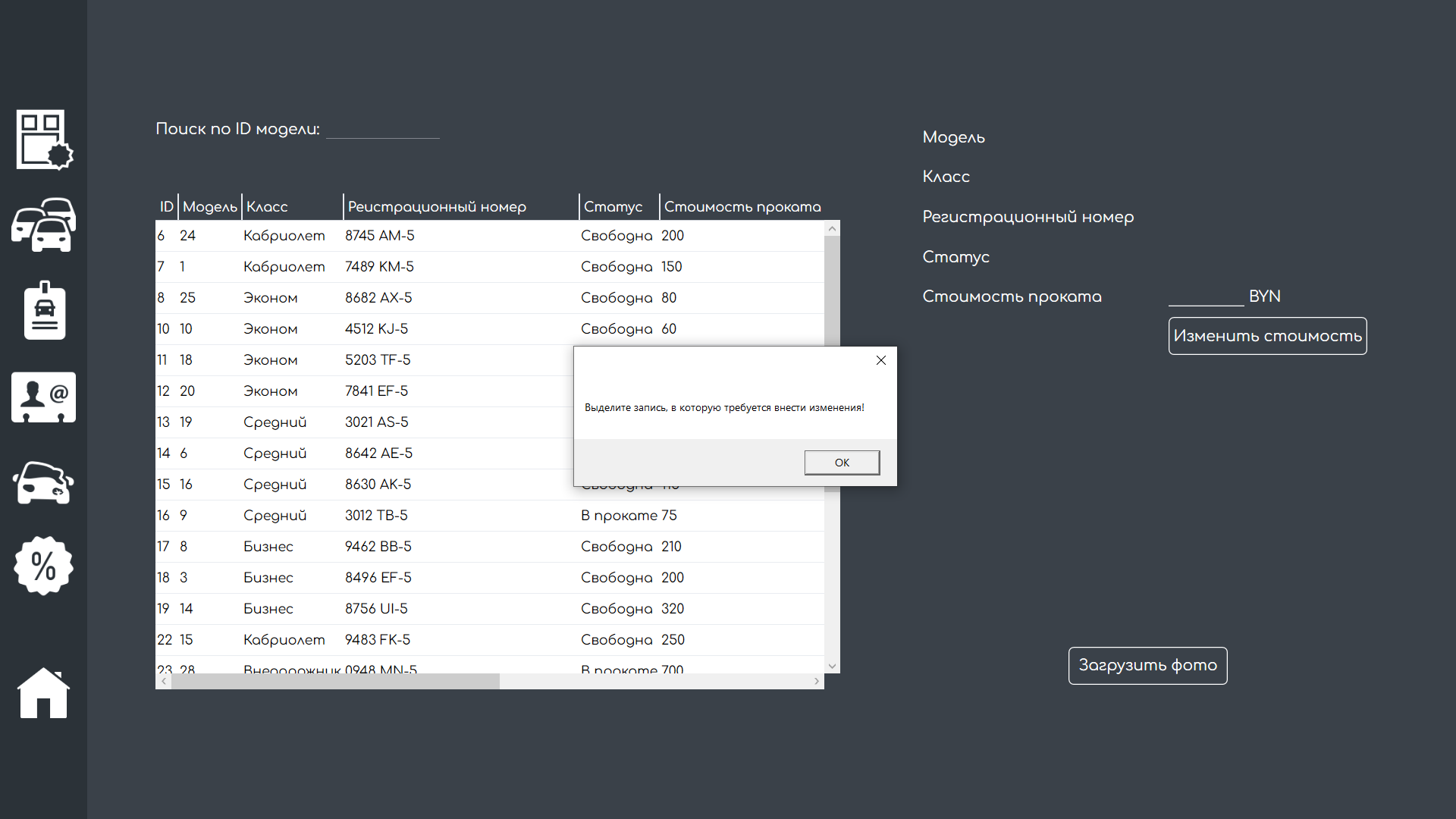


Рисунок 5.7 — Обработка нажатия на кнопку «Изменить» при отсутствии выбранной записи

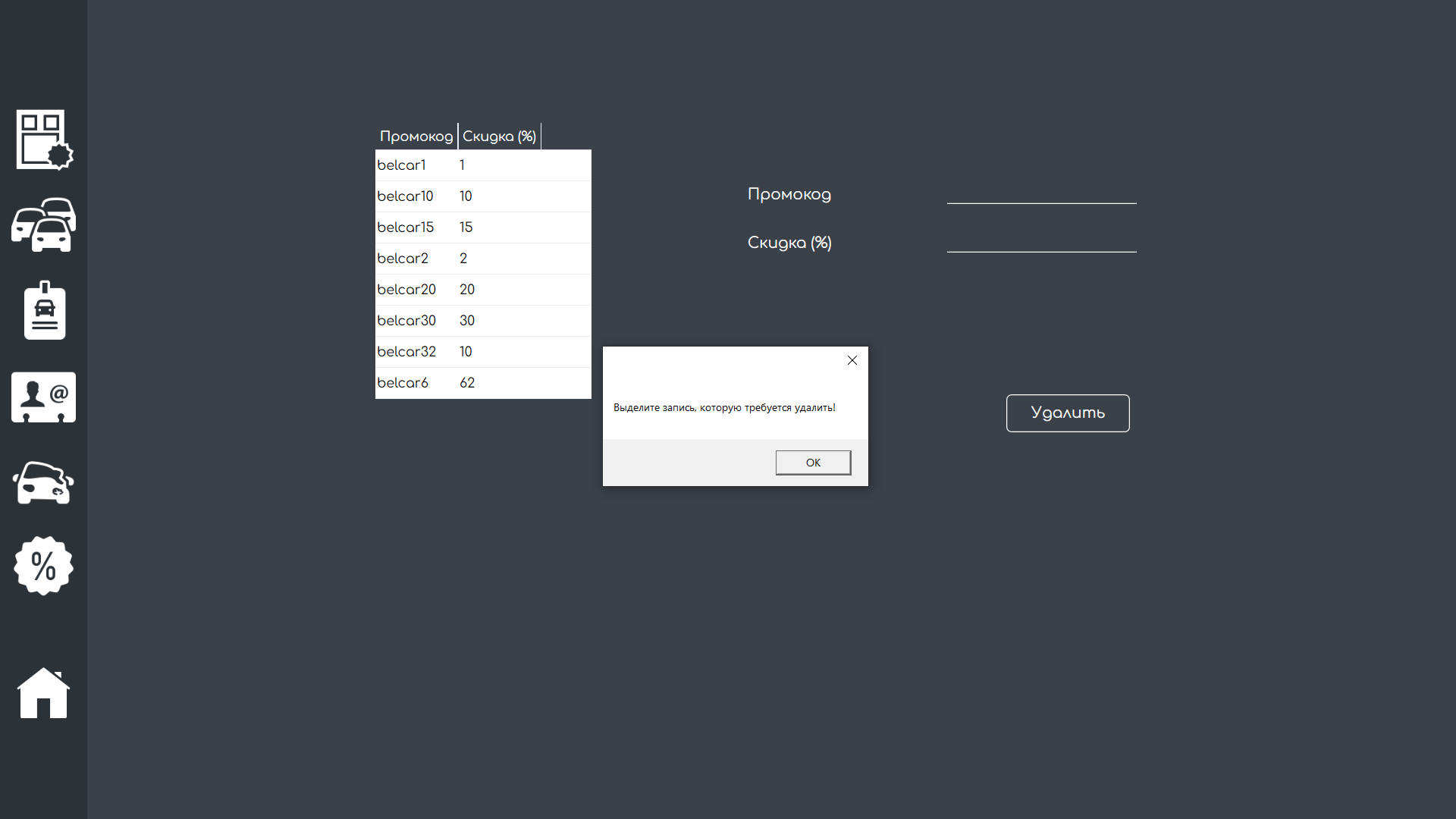


Рисунок 5.8 — Обработка нажатия на кнопку «Удалить» при отсутствии выбранной записи

# Руководство по установке и использованию

При запуске программного средства «belcar» пользователь попадает на страницу авторизации, содержащую формы входа и регистрации.

Стартовая страница позволяет зарегистрированному пользователю войти в аккаунт, а не зарегистрированному – перейти на страницы регистрации и заполнить данными все необходимые поля.

По умолчанию новому зарегистрированному клиенту присваивается роль «Пользователь». Роль «Администратор» при регистрации не присваивается. После регистрации пользователь будет перенаправлен на главную страницу приложения. На главной странице находится краткое описание сервиса по указанию услуг, а также кнопки, позволяющие перемещаться по приложению. Страница входа, регистрации и главная страницы представлены на рисунках 6.1 – 6.4.

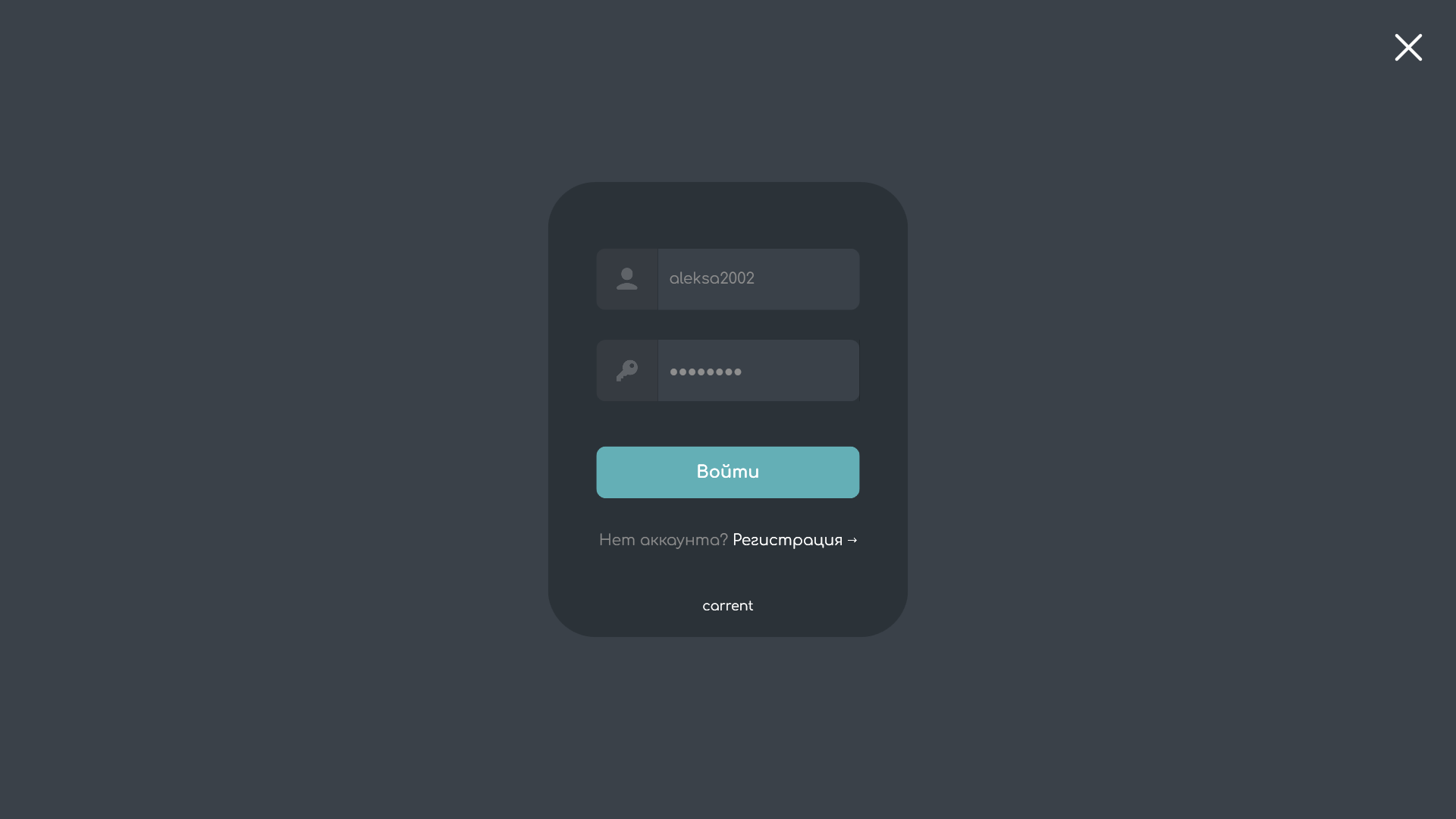


Рисунок 6.1 — Страница входа в аккаунт



Рисунок 6.2 — Первая страница авторизации



Рисунок 6.3 — Вторая страница авторизации

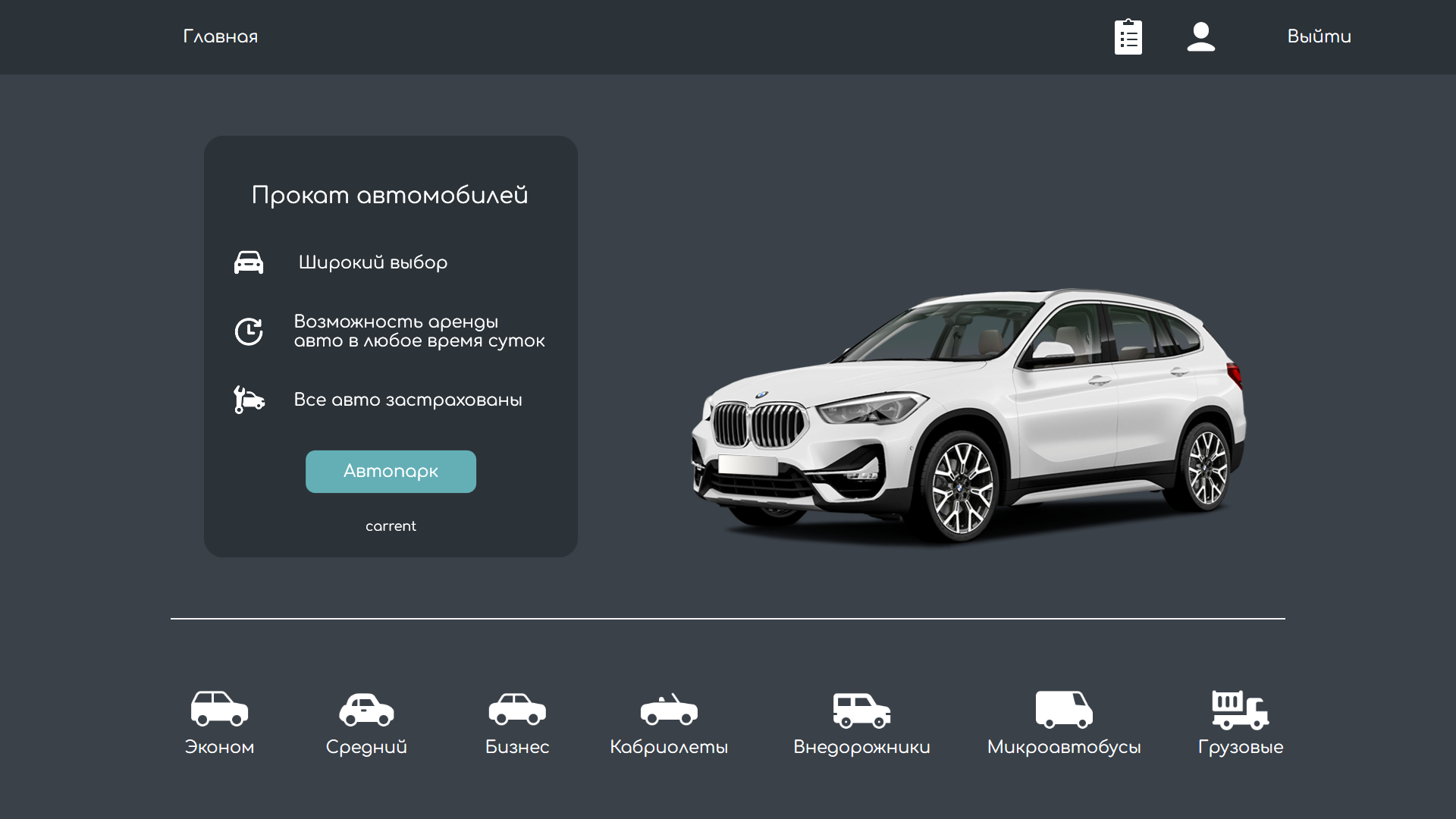


Рисунок 6.4 — Главная страница

Кнопка «Автопарк» и кнопки с названиями классов открывают соответствующие страницы с перечнем автомобилей и возможностью подбора автомобиля по заданным параметрам (Рисунок 6.5).

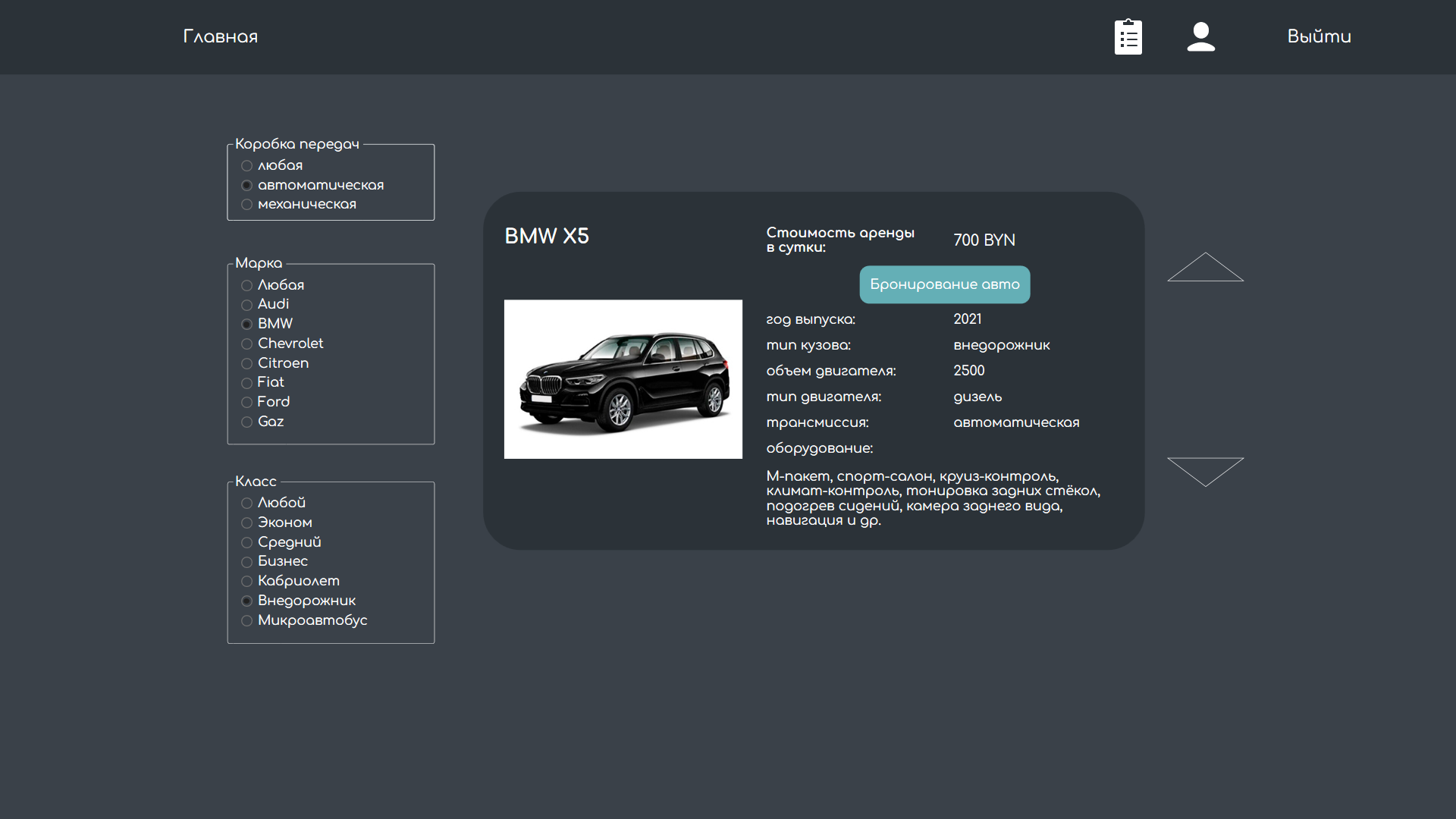


Рисунок 6.5 — Страница подбора авто и результат поиска

Если не было найдено ни одного автомобиля, соответствующего заданным параметрам, то выводится надпись «Данные не найдены!».

Далее, чтобы забронировать авто, необходимо нажать по кнопке «Бронирование авто». После чего откроется окно, необходимое для заполнения нужных данных для формирования заявки (Рисунок 6.6).

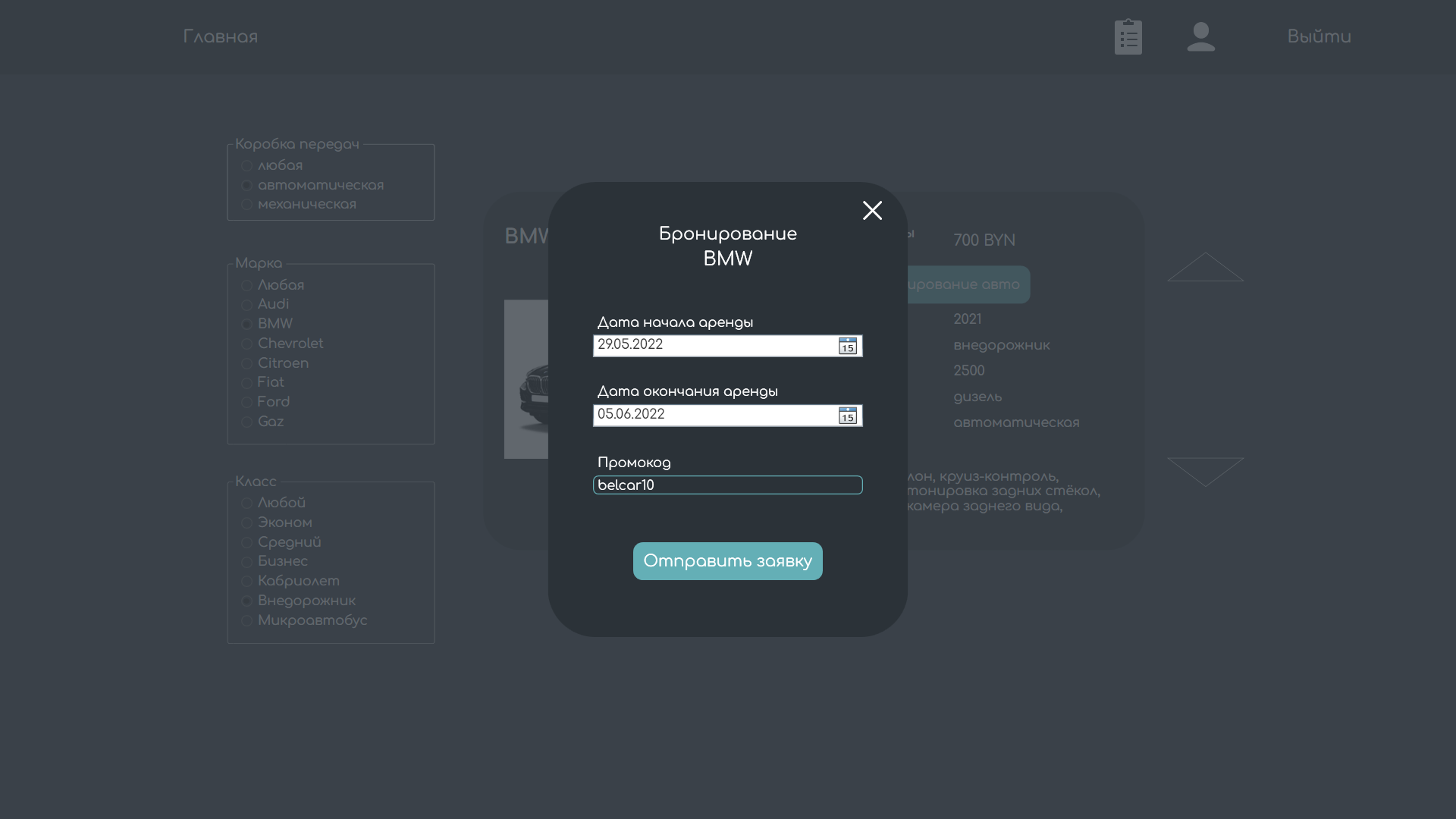


Рисунок 6.6 — Окно заполнения и отправления заявки

Далее при успешном отправлении заявки пользователь перенаправляется на страницу с историей прокатов (Рисунок 6.7). На данной странице пользователь может просмотреть уже завершенные прокаты или отменить еще не подтвержденные заявки. Для удаления заявки необходимо нажать на кнопку «Отменить заявку». После удаления заявки, информация о статусе заявки изменится (Рисунок 6.8).

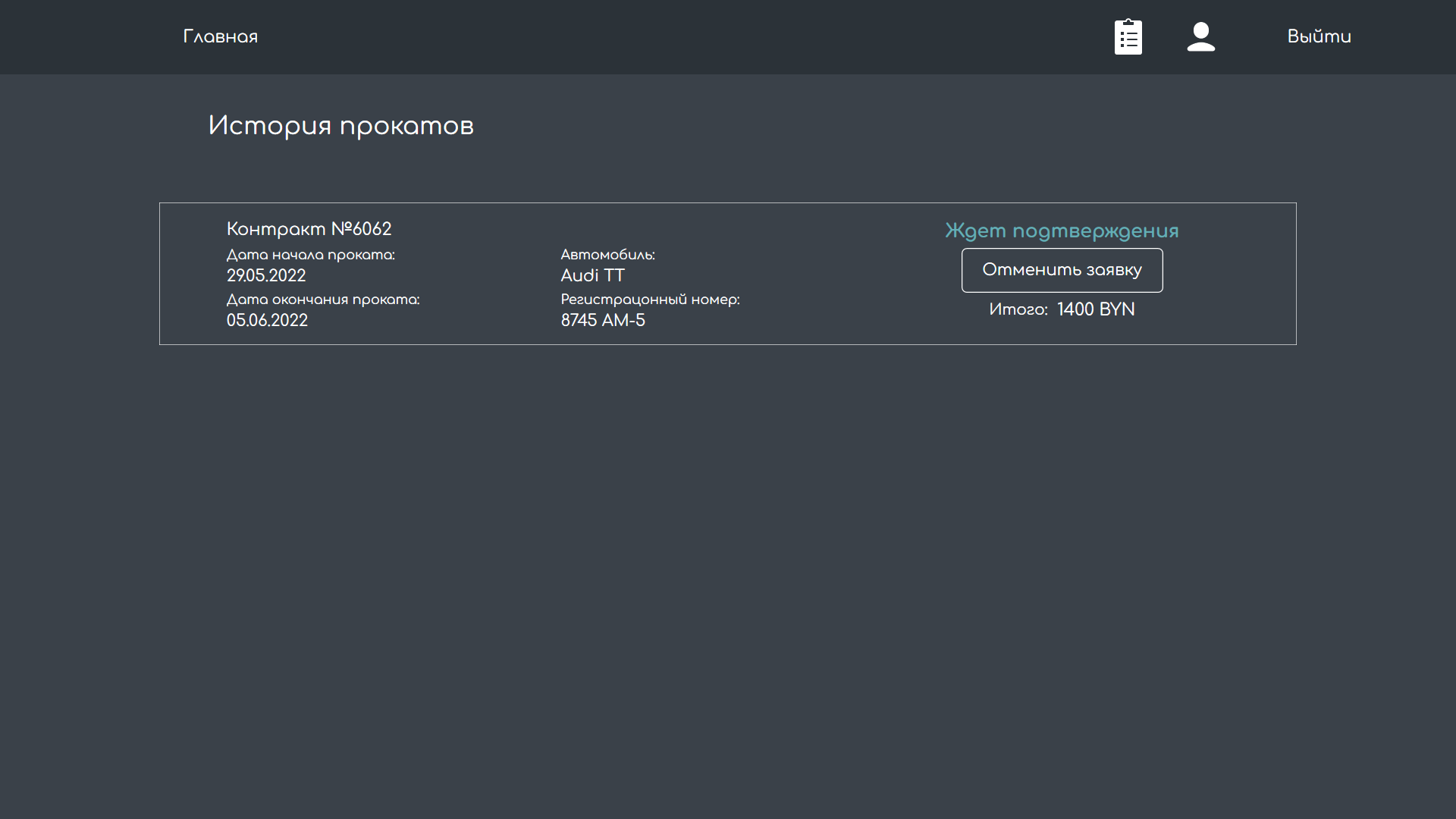


Рисунок 6.7 — История прокатов

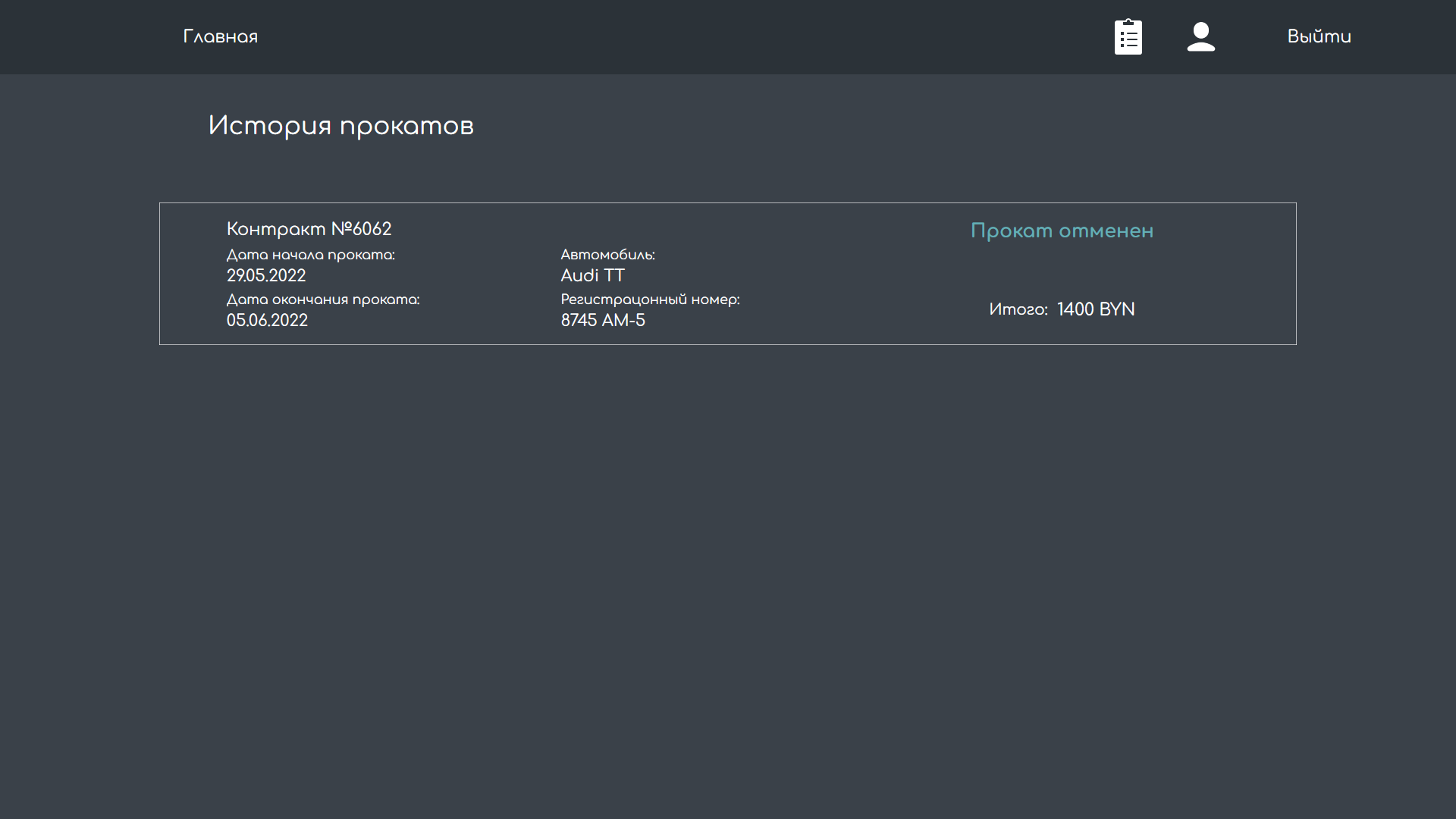


Рисунок 6.8 — Отмена заявки на бронь

Для пользователя также доступна страница с личной информацией, в которой он может изменить свою личную информацию. Для этого достаточно просто внести изменения в форму и нажать на кнопку «Изменить». При сохранении информации происходит валидация данных и при вводе некорректных данных программное средство сообщает пользователю об ошибках появлением соответствующих подсказок.

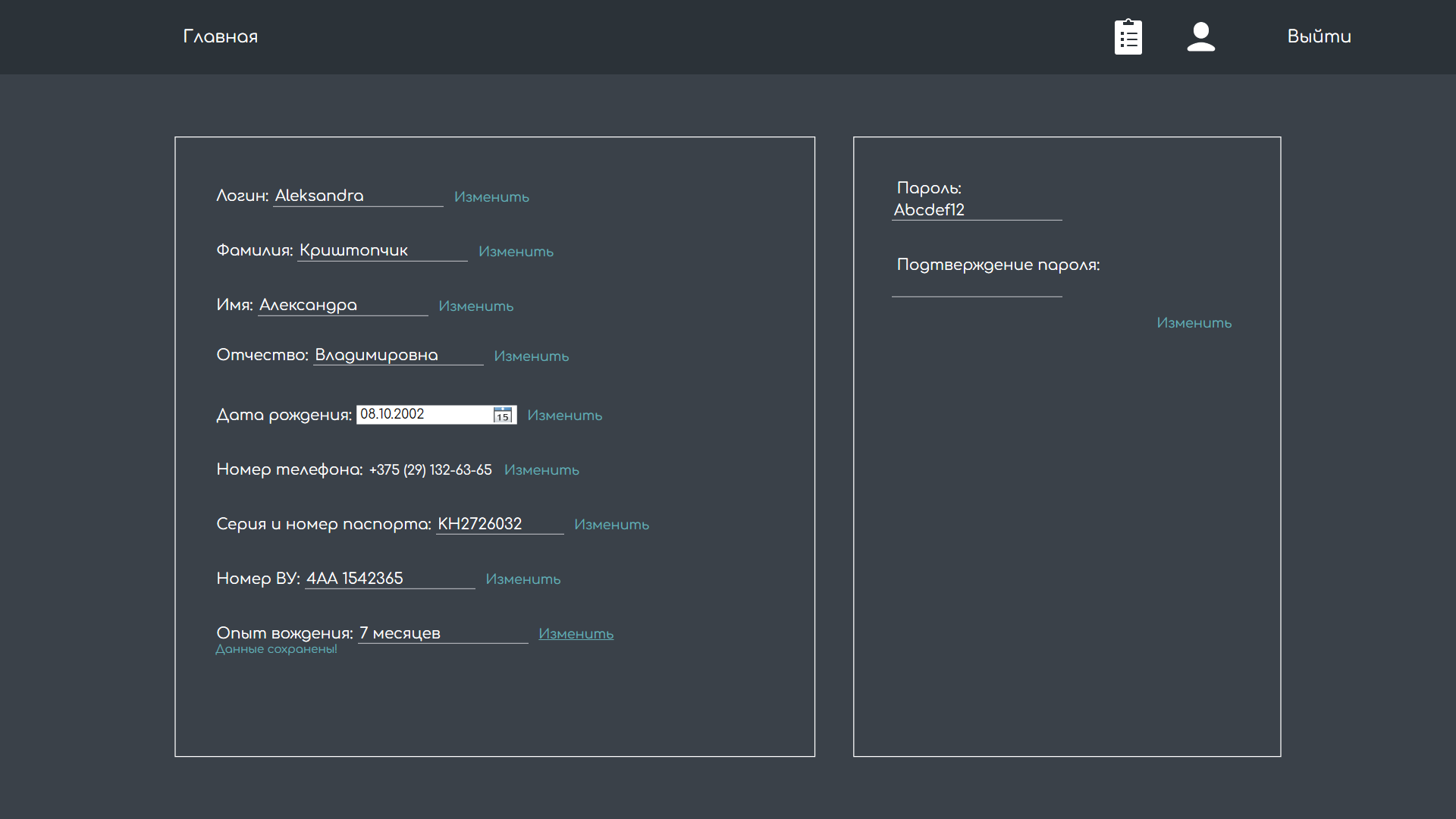


Рисунок 6.9 — Успешно внесенные изменения

Для выхода из учетной записи необходимо нажать на кнопку «Выйти». Если требуется полностью выйти из приложения, то при переходе на страницу входа, следует нажать на крестик в правом верхнем углу.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано программное средство «belcar» с целью аренды автомобиля. При выполнении были изучены: подсистема для построения графических интерфейсов WPF, фреймворк доступа к данным Entity Framework, унифицированный язык моделирования UML. Так же были получены практические навыки в работе с языком верстки XAML и объектно-ориентированным языком программирования С#.

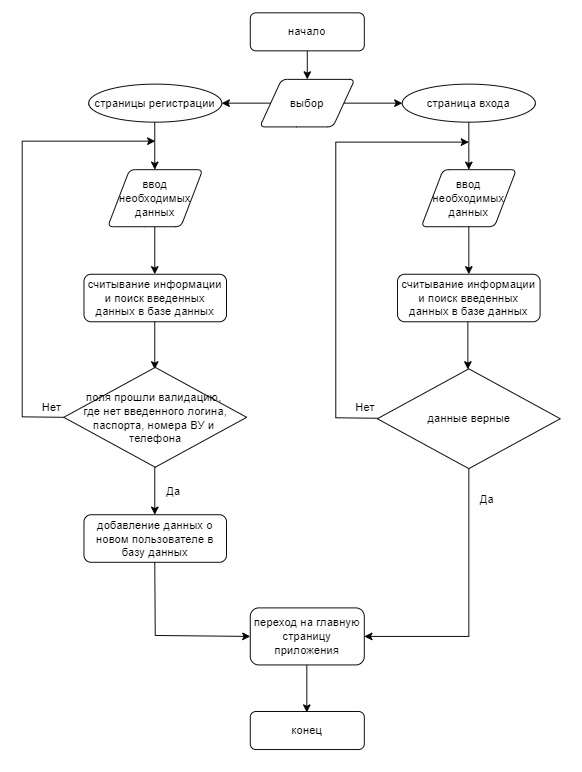
Тестирование программы показало, что она работает корректно и выполняет все свои функции, описанные в требованиях, а именно:

* управление администратором базой данных;
* уведомление администратора о поступлении новой заявки на бронь;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность просмотра каталога автомобилей;
* возможность поиска подходящего авто по заданным параметрам
* возможность оформления заявки на бронь;
* возможность пользователя просматривать информацию о всех его прокатах;
* возможность просмотра личной информации пользователя и ее изменение.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Microsoft [Электронный ресурс] –Общие сведения о Visual Studio– Режим доступа: <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> – Дата доступа 17.05.2022.
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 14.03.2022
3. Microsoft [Электронный ресурс] –Тур по С#– Режим доступа: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/> – Дата доступа: 18.05.2022.
4. Microsoft [Электронный ресурс] – Что такое Windows Presentation Forms(WPF) – Режим доступа: [https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022 –](https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022%20–) Дата доступа: 19.05.2022.
5. Metanit [Электронный ресурс]. – C# и WPF | XAML – Режим доступа: https://metanit.com/sharp/wpf/2.php/– Дата доступа: 18.05.2022.
6. Metanit [Электронный ресурс]. – Работа с EF. – Режим доступа: https://metanit.com/sharp/wpf/2.php/– Дата доступа: 18.05.2022.
7. ProfessorWeb [Электронный ресурс] – веб-ресурс с современными уроками по C# и платформе .NET Framework. – Режим доступа: https://professorweb.ru/. – Дата доступа: 20.04.2022.
8. Metanit [Электронный ресурс] – Сайт о программировании. – Режим доступа: https://metanit.com/sharp/wpf/. – Дата доступа: 17.05.2022.

# ПРИЛОЖЕНИЕ А



# ПРИЛОЖЕНИЕ Б

Листинг 1: методы Come\_Click и Hyperlink\_Click

|  |
| --- |
| private void Come\_Click(object sender, RoutedEventArgs e)  {  {  if (login.Text.Length != 0 && password.Password.Length != 0)  {  loginMessage.Text = "";  passwordMessage.Text = "";  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  string log = login.Text.Trim();  string pass = password.Password.Trim();    CLIENT admin = db.CLIENTS.Where(a => a.LOGIN.Trim() == log && a.USER\_TYPE.ToString().Trim() == "1").AsEnumerable().Where(a => a.LOGIN.Trim() == log && a.USER\_TYPE.ToString().Trim() == "1").FirstOrDefault();  if (admin != null)  {  if (admin.PASSWORD.Trim() == pass && admin.LOGIN.Trim() == log)  {  AdminWindow adminWindow = new AdminWindow();  App.admin = admin;  adminWindow.Show();  MainWindow.Close();  }  else  {  if(admin.PASSWORD.Trim() != pass && admin.LOGIN.Trim() == log)  passwordMessage.Visibility = Visibility.Visible;  passwordMessage.Text = "Неправильный пароль!";  return;  }  }  else  {  loginMessage.Visibility = Visibility.Visible;  loginMessage.Text = "Пользователь не найден!";  }  passwordMessage.Visibility = Visibility.Hidden;  loginMessage.Visibility = Visibility.Hidden;  CLIENT client = db.CLIENTS.Where(u => u.LOGIN.Trim() == log && u.USER\_TYPE.ToString().Trim() == "0").AsEnumerable().Where(u => u.LOGIN.Trim() == log && u.USER\_TYPE.ToString().Trim() == "0").FirstOrDefault();  if (client != null)  {  if (client.PASSWORD.Trim() == pass && client.LOGIN.Trim() == log)  {  CatalogWindow catalog = new CatalogWindow();  App.currentClient = client;  catalog.Show();  Application.Current.MainWindow.Close();  }  else  {  if (client.PASSWORD.Trim() != pass && client.LOGIN.Trim() == log)  passwordMessage.Visibility = Visibility.Visible;  passwordMessage.Text = "Неправильный пароль!";  }  }  else  {    loginMessage.Visibility = Visibility.Visible;  loginMessage.Text = "Пользователь не найден!";      }  }  }  else  {  loginMessage.Text = "";  passwordMessage.Text = "";  if (login.Text.Length == 0)  {  loginMessage.Visibility = Visibility.Visible;  loginMessage.Text = "Введите логин!";  }  if (password.Password.Length == 0)  {  passwordMessage.Visibility = Visibility.Visible;  passwordMessage.Text = "Введите пароль!";  }  }  }  private void Hyperlink\_Click(object sender, RoutedEventArgs e)  {  AuthorizationPage authorizationPage = new AuthorizationPage();  NavigationService.Navigate(authorizationPage);  } |

Листинг 2: регистрация

|  |
| --- |
| private void Next(object sender, RoutedEventArgs e)  {  string log = login.Text.Trim();  string pass = password.Password.Trim();  string passConf = passwordConfirmation.Password.Trim();  string surn = surname.Text.Trim();  string n = name.Text.Trim();  string patron = patronymic.Text.Trim();  string patternLogin = @"^[a-zA-Zа-яА-Я0-9\_-]{3,16}$";  string patternPassword = @"^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])[A-Za-z0-9]{8,12}$";  string patternName = @"^[А-Я][а-я'-]+$";  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  CLIENT client = db.CLIENTS.Where(u => u.LOGIN.Trim() == log).AsEnumerable().Where(u => u.LOGIN.Trim() == log).FirstOrDefault();  if (client != null)  {  busy.Visibility = Visibility.Visible;  }  else  {  if (Regex.IsMatch(log, patternLogin)  && Regex.IsMatch(pass, patternPassword)  && Regex.IsMatch(passConf, patternPassword)  && Regex.IsMatch(surn, patternName)  && Regex.IsMatch(n, patternName)  && Regex.IsMatch(patron, patternName)  && logDone.Visibility == Visibility.Visible  && passDone.Visibility == Visibility.Visible  && passConfDone.Visibility == Visibility.Visible  && surnDone.Visibility == Visibility.Visible  && nDone.Visibility == Visibility.Visible  && patronDone.Visibility == Visibility.Visible)  {  this.NavigationService.Navigate(new AuthorizationPageDataInfo(log, pass, surn, n, patron));  }  else  {  if (pass != passConf)  {  notmatch.Visibility = Visibility.Visible;  }  else if (pass == passConf)  {  fillAll.Visibility = Visibility.Visible;  }  }  }  }  }  private void login\_SelectionChanged(object sender, RoutedEventArgs e)  {  string log = login.Text.Trim();  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  CLIENT client = db.CLIENTS.Where(c => c.LOGIN.Trim() == log).FirstOrDefault();  string patternLogin = @"^[a-zA-Zа-яА-Я0-9\_-]{3,16}$";  if (client != null)  {  busy.Visibility = Visibility.Visible;  }  if (client == null && Regex.IsMatch(log, patternLogin))  {  logDone.Visibility = Visibility.Visible;  busy.Visibility = Visibility.Hidden;  }  else  {  logDone.Visibility = Visibility.Hidden;  }  }  }  private void password\_SelectionChanged(object sender, RoutedEventArgs e)  {  string patternPassword = @"^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])[A-Za-z0-9]{8,12}$";  string pass = password.Password.Trim();  if (Regex.IsMatch(pass, patternPassword))  {  passDone.Visibility = Visibility.Visible;  }  else  {  passDone.Visibility = Visibility.Hidden;  }  }  private void passwordConfirmation\_SelectionChanged(object sender, RoutedEventArgs e)  {  string pass = password.Password.Trim();  string passConf = passwordConfirmation.Password.Trim();  if (pass == passConf)  {  passConfDone.Visibility = Visibility.Visible;  notmatch.Visibility = Visibility.Hidden;  }  if (passConf.Length == 0)  {  passConfDone.Visibility = Visibility.Hidden;  }  else if (pass != passConf)  {  notmatch.Visibility = Visibility.Visible;  passConfDone.Visibility = Visibility.Hidden;  }  } private void surname\_SelectionChanged(object sender, RoutedEventArgs e)  {  string surn = surname.Text.Trim();  string patternName = @"^[А-Я][а-я'-]+$";  if (Regex.IsMatch(surn, patternName))  {  surnDone.Visibility = Visibility.Visible;  }  else  {  surnDone.Visibility = Visibility.Hidden;  }  }  private void Button\_Click(object sender, RoutedEventArgs e)  {  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  DateTime date = new DateTime();  DateTime.TryParse(bday.Text.Trim(), out date);  CLIENT client = new CLIENT();  client.LOGIN = login;  client.PASSWORD = password;  client.SURNAME = surname;  client.NAME = name;  client.PATRONYMIC = patronymic;  client.BDAY = date;  client.TELEPHONE = telephone.Text.Trim();  client.PASSPORT = passport.Text;  client.DRIVER\_LICENSE\_ID = licenseID.Text.Trim();  client.DRIVING\_EXPERIENCE = experience.Text.Trim();  if (telephDone.Visibility == Visibility.Visible  && bdayDone.Visibility == Visibility.Visible  && passportDone.Visibility == Visibility.Visible  && numberDone.Visibility == Visibility.Visible  && numDone.Visibility == Visibility.Visible)  {  db.CLIENTS.Add(client);  db.SaveChanges();  App.currentClient = client;  AuthorizattionIsDone authorizattionIsDone = new AuthorizattionIsDone();  NavigationService.Navigate(authorizattionIsDone);  }  else  {  if (passport.Text != "")  {  MessageBox.Show("Необходимо заполнить все поля!");  }  }  }  }  private void bday\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)  {  DateTime date = new DateTime();  DateTime.TryParse(bday.Text.Trim(), out date);  DateTime today = DateTime.Today;  if ((today.Year - date.Year) < 18)  {  MessageBox.Show("Извините! Вам нет 18 лет! Мы не сможем предоставить вам автомобиль!");  return;  }  if ((today.Year - date.Year) > 90)  {  MessageBox.Show("Извините! Некорректно введенные данные!");  return;  }  else  {  if (bday.Text.Trim() != null && bday.Text.Trim().Length == 10)  {  bdayDone.Visibility = Visibility.Visible;  }  else  {  bdayDone.Visibility = Visibility.Hidden;  }  }  }  private void telephone\_TextChanged(object sender, TextChangedEventArgs e)  {  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  string patternTelephone = @"^\+375 \((25|29|33|44)\) [0-9]{3}-[0-9]{2}-[0-9]{2}$";  string teleph = telephone.Text.Trim();  CLIENT clientTelephone = db.CLIENTS.FirstOrDefault(u => u.TELEPHONE.Trim() == telephone.Text.Trim());  if (clientTelephone != null)  {  telephMessage.Visibility = Visibility.Visible;  }  else  {  if (Regex.IsMatch(teleph, patternTelephone) && clientTelephone == null)  {  telephDone.Visibility = Visibility.Visible;  telephMessage.Visibility = Visibility.Hidden;  }  else  {  telephDone.Visibility = Visibility.Hidden;  }  }  }  }  9))  { |

Листинг 3: метод Search() и класс Car()

|  |
| --- |
| private void Search()  {  StackPanel.Items.Clear();  string transmission = CheckedTransmission;  string brand = CheckedBrand;  string classs = CheckedClass;  if (transmission == "любая")  {  transmission = null;  }  if (brand == "Любая")  {  brand = null;  }  if (classs == "Любой")  {  classs = null;  }  var regexClass = new Regex(@"\w\*" + classs + @"\w\*", RegexOptions.IgnoreCase);  var regexBrand = new Regex(@"\w\*" + brand + @"\w\*", RegexOptions.IgnoreCase);  var regexTrans = new Regex(@"\w\*" + transmission + @"\w\*", RegexOptions.IgnoreCase);  search.Visibility = Visibility.Visible;  mainInfo.Visibility = Visibility.Hidden;  mainCar.Visibility = Visibility.Hidden;  classes.Visibility = Visibility.Hidden;  line.Visibility = Visibility.Hidden;  Classes.Visibility = Visibility.Visible;  using (CAR\_RENTEntities db = new CAR\_RENTEntities())  {  var carsCatalog = from cars in db.CARS  join models in db.MODEL\_INFO  on cars.MODEL equals models.ID  select new  {  CurrentName = models.BREND + " " + models.MODEL,  Price = cars.RENT\_PRICE,  Year = models.YEAR\_OF\_ISSUE,  BodyType = models.BODY\_TYPE,  EngineCapacity = models.ENGINE\_CAPACITY,  EngineType = models.ENGINE\_TYPE,  Transmission = models.TRANSMISSION,  Equipment = models.EQUIPMENT,  Image = cars.IMAGE,  Id = cars.ID,  Class = cars.CLASS,  Trans = models.TRANSMISSION,  CurName = models.BREND  };  foreach (var car in carsCatalog)  {  var buf = new Car(car.CurrentName, car.Price.ToString(),  car.Year.ToString().Remove(0, 6).Remove(4), car.BodyType, car.EngineType, car.EngineCapacity,  car.Transmission, car.Equipment, car.Image, car.Id.ToString(), car.Class.ToString(), car.CurName.ToString(), car.Trans.ToString());  buf.Width = 740;  buf.Height = 420;  string a = buf.Transm.ToString() + buf.CarName.ToString() + buf.Clas.ToString();  var matchesClass = regexClass.Matches(a);  var matchesBrand = regexBrand.Matches(a);  var matchesTrans = regexTrans.Matches(a);  if ((matchesClass.Count > 0 || classs == null) && (matchesBrand.Count > 0 || brand == null) && (matchesTrans.Count > 0 || transmission == null))  {  StackPanel.Items.Add(buf);    }  if (StackPanel.Items.Count == 0)  {  result.Text = "Данные не найдены!";  result.Visibility = Visibility.Visible;  }  }  }  } public partial class Car : UserControl  {  public string ID { get; set; }  public string CarName { get; set; }  public string Clas { get; set; }  public string Transm { get; set; }  public string CurName { get; set; }  public Car(string CurrentName, string Price, string Year,string BodyType, string EngineType,  string EngineCapacity, string Transmission, string Equipment, string Image, string Id, string Class, string Name, string Trans)  {  InitializeComponent();  currentName.Text = CurrentName;  price.Text = Price + " BYN";  year.Text = Year;  bodyType.Text = BodyType;  engineType.Text = EngineType;  engineCapacity.Text = EngineCapacity;  transmission.Text = Transmission;  equipment.Text = Equipment;  BitmapImage myBitmapImage = new BitmapImage(new Uri(Image));  myBitmapImage.CacheOption = BitmapCacheOption.OnLoad;  currentImage.Source = myBitmapImage;  ID = Id;  CarName = Name;  Clas = Class;  Transm = Trans;    }    private void Rent(object sender, RoutedEventArgs e)  {  rent rent = new rent(ID, CarName);  rent.Show();    }  } |

Листинг 4: метод send\_Click()

|  |
| --- |
| private void send\_Click(object sender, RoutedEventArgs e)  {  CONTRACT contract = new CONTRACT();  StringBuilder errors = new StringBuilder();  DateTime contract\_start = new DateTime();  DateTime.TryParse(CONTRACT\_START.Text, out contract\_start);  DateTime contract\_end = new DateTime();  DateTime.TryParse(CONTRACT\_END.Text, out contract\_end);  TimeSpan days = contract\_end - contract\_start;  if (string.IsNullOrWhiteSpace(CONTRACT\_START.Text))  {  errors.AppendLine();  }  else  {  }  if (string.IsNullOrWhiteSpace(CONTRACT\_END.Text))  {  errors.AppendLine();  }  double promocode = 0;  PROMO\_CODE promo\_code = null;  if (PROMO\_CODE.Text.Length != 0)  {  contract.PROMO\_CODE = PROMO\_CODE.Text;  promo\_code = CAR\_RENTEntities.GetContext().PROMO\_CODE.Where(c => c.PROMO\_CODE1 == PROMO\_CODE.Text).FirstOrDefault();  if (promo\_code != null)  {  promocode = Convert.ToDouble(promo\_code.DISCOUNT\_AMOUNT) \* 0.01;  }  else  {  promocodeMessage.Visibility = Visibility.Visible;  promocodeMessage.Text = "Такого промокода не существует!";  PROMO\_CODE.Clear();  errors.AppendLine();  }  }  else contract.PROMO\_CODE = null;  if (days.Days < 0)  {  dateMessage.Visibility = Visibility.Visible;  dateMessage.Text = "Введите корректные даты!";  errors.AppendLine();  }  if (errors.Length > 0 && days.Days > 0)  {  }  if (errors.Length > 0)  {  return;  }  contract.CLIENT\_ID = App.currentClient.ID;  contract.CAR\_ID = Convert.ToInt32(ID);  contract.CONTRACT\_START = Convert.ToDateTime(CONTRACT\_START.Text);  contract.CONTRACT\_END = Convert.ToDateTime(CONTRACT\_END.Text);  CAR car = CAR\_RENTEntities.GetContext().CARS.Where(c => c.ID.ToString() == ID.ToString()).FirstOrDefault();  double total\_price=0;  if(days.Days > 0)  {  total\_price = Convert.ToInt32(car.RENT\_PRICE) \* days.Days - Convert.ToInt32(car.RENT\_PRICE) \* days.Days \* promocode;  }  else if (days.Days == 0)  {  total\_price = Convert.ToInt32(car.RENT\_PRICE) \* 1 - Convert.ToInt32(car.RENT\_PRICE) \* 1 \* promocode;  }  contract.TOTAL\_COST = Convert.ToInt32(total\_price);  contract.STATUS = "Новая заявка";  contract.CONTRACT\_STATUS = "Ждет подтверждения";  CAR\_RENTEntities.GetContext().CONTRACTS.Add(contract);  using(CAR\_RENTEntities db= new CAR\_RENTEntities())  {  try  {  MailAddress from = new MailAddress("aleksa-vesna@mail.ru", "Aleksandra");  // кому отправляем  MailAddress to = new MailAddress("aleksa-vesna@mail.ru");  // создаем объект сообщения  MailMessage m = new MailMessage(from, to);  // тема письма  m.Subject = "Belcar";  // текст письма  m.Body = "<h2>Вам поступила новая заявка на бронь авто!</h2>";  // письмо представляет код html  m.IsBodyHtml = true;  //адрес smtp-сервера и порт, с которого будем отправлять письмо  SmtpClient smtp = new SmtpClient("smtp.mail.ru", 587);  //логин и пароль  smtp.UseDefaultCredentials = false;  smtp.Credentials = new NetworkCredential("aleksa-vesna@mail.ru", "mQYMHqTfzpsjDzbnZ59b");  smtp.EnableSsl = true;  smtp.DeliveryMethod = SmtpDeliveryMethod.Network;  smtp.Send(m);  CAR\_RENTEntities.GetContext().SaveChanges();  MessageBox.Show("Заявка отправлена!");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  return;  }  this.Close();  CatalogWindow.Frame.Navigate(new UserContracts());  }    } |

Листинг 5: класс HistoryContract

|  |
| --- |
| public partial class HistoryContract : UserControl  {  private string ID { get; set; }    public HistoryContract(string ContractId, string ContractStart, string ContractEnd, string Car, string RegisterNum, string Status, string TotalCost)  {  InitializeComponent();  CONTRACT\_ID.Text += ContractId;  ID = ContractId;  CONTRACT\_START.Text = ContractStart;  CONTRACT\_END.Text = ContractEnd;  CAR.Text = Car;  REGISTRATION\_NUMBER.Text = RegisterNum;  STATUS.Text = Status;  TOTAL\_COST.Text = TotalCost;  if (Status == "Ждет подтверждения")  {  cancel.Visibility = Visibility.Visible;  }  else cancel.Visibility = Visibility.Hidden;  }  private void cancel\_Click(object sender, RoutedEventArgs e)  {  CONTRACT contract=CAR\_RENTEntities.GetContext().CONTRACTS.Where(c=>c.ID.ToString()==ID).FirstOrDefault();  if (contract != null)  {  contract.STATUS = "Отменена";  contract.CONTRACT\_STATUS = "Прокат отменен";  CAR\_RENTEntities.GetContext().SaveChanges();  CatalogWindow.Frame.Navigate(new UserContracts());    }  }  } |

Листинг 6: окно администратора

|  |
| --- |
| private void id\_TextChanged(object sender, TextChangedEventArgs e)  {  using(CAR\_RENTEntities db=new CAR\_RENTEntities())  {  var contracts=(from contract in db.CONTRACTS  where contract.CLIENT\_ID.ToString()==id.Text  join accident in db.ACCIDENTS  on contract.ID equals accident.CONTRACT\_ID  select accident).ToList();  DGridAccidents.ItemsSource=contracts;  if (contracts.Count == 0)  {  DGridAccidents.ItemsSource = CAR\_RENTEntities.GetContext().ACCIDENTS.ToList();  }  }  }  private void Edit\_Click(object sender, RoutedEventArgs e)  {  if (ID.Text.Equals(""))  {  MessageBox.Show("Выделите запись, в которую требуется внести изменения!");  return;  }  CAR currentCar = CAR\_RENTEntities.GetContext().CARS.Where(m => m.ID.ToString() == ID.Text.ToString()).FirstOrDefault();  currentCar.RENT\_PRICE = Convert.ToInt32(RENT\_PRICE.Text.Trim());  if (currentCar != null)  {  try  {  CAR\_RENTEntities.GetContext().SaveChanges();  DGridCars.ItemsSource = CAR\_RENTEntities.GetContext().CARS.ToList();  Clear();  MessageBox.Show("Цена успешно обновлена!");  }  catch  {  MessageBox.Show("Необходимо выбрать запись для редактирования!");  }  }  }  }  private void Add\_Click(object sender, RoutedEventArgs e)  {    try  {  StringBuilder errors = new StringBuilder();  if(string.IsNullOrWhiteSpace(PROMO\_CODE.Text.Trim())  && string.IsNullOrWhiteSpace(DISCOUNT\_AMOUNT.Text.ToString().Trim()))  {  errors.AppendLine("Необходимо заполнить все поля!");  MessageBox.Show(errors.ToString());  errors.Clear();  return;  }  if (string.IsNullOrWhiteSpace(PROMO\_CODE.Text.Trim()))  {  errors.AppendLine("Введите промокод!");  }  if (string.IsNullOrWhiteSpace(DISCOUNT\_AMOUNT.Text.ToString().Trim()))  {  errors.AppendLine("Введите величину скидки!");  }  PROMO\_CODE currentPromocode = CAR\_RENTEntities.GetContext().PROMO\_CODE.Where(u => u.PROMO\_CODE1.Trim() == PROMO\_CODE.Text.Trim()).FirstOrDefault();  if(currentPromocode != null)  {  errors.AppendLine("Такой промокод уже существует!");  }  if (errors.Length > 0)  {  MessageBox.Show(errors.ToString());  errors.Clear();  return;  }  PROMO\_CODE currentPromo\_code = new PROMO\_CODE();  currentPromo\_code.PROMO\_CODE1 = PROMO\_CODE.Text.Trim();  currentPromo\_code.DISCOUNT\_AMOUNT = DISCOUNT\_AMOUNT.Text.Trim();  CAR\_RENTEntities.GetContext().PROMO\_CODE.Add(currentPromo\_code);  CAR\_RENTEntities.GetContext().SaveChanges();  DGridPromocode.ItemsSource = CAR\_RENTEntities.GetContext().PROMO\_CODE.ToList();  Clear();  MessageBox.Show("Данные успешно добавлены!");  }  catch (DbEntityValidationException ex)  {    foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)  {  foreach (DbValidationError err in validationError.ValidationErrors)  {  MessageBox.Show(err.ErrorMessage + " ");  }  }  }  }    private void Delete\_Click(object sender, RoutedEventArgs e)  {  if (PROMO\_CODE.Text.Equals(""))  {  MessageBox.Show("Выделите запись, которую требуется удалить!");  }  PROMO\_CODE currentPromocode = CAR\_RENTEntities.GetContext().PROMO\_CODE.Where(p => p.PROMO\_CODE1.Trim() == PROMO\_CODE.Text.Trim()).FirstOrDefault();  var contract = CAR\_RENTEntities.GetContext().CONTRACTS.Where(c => c.PROMO\_CODE.Trim() == PROMO\_CODE.Text.Trim()).ToList();  if (currentPromocode != null && contract.Count==0)  {  CAR\_RENTEntities.GetContext().PROMO\_CODE.Remove(currentPromocode);  try  {  CAR\_RENTEntities.GetContext().SaveChanges();  DGridPromocode.ItemsSource = CAR\_RENTEntities.GetContext().PROMO\_CODE.ToList();  Clear();  MessageBox.Show("Запись удалена!");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  if(currentPromocode != null && contract.Count>=1)  {  MessageBox.Show("Удаление промокода и и нформации о нем невозможно!");  }  else if (currentPromocode == null && string.IsNullOrEmpty(PROMO\_CODE.Text.Trim()) == false)  {  MessageBox.Show("Такого промокода не существует!");  }  } |