МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6– 05– 0612– 01 «Программная инженерия»

Направление специальности 6– 05– 0612– 01 «Программная инженерия»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно–ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Велопрокат»

Исполнитель

Студент (ка) 2 курса группы 10 Мандрик А.И.

(Ф.И.О.)

Руководитель работы асс. Мущук А.Н.

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

Курсовой проект защищен с оценкой

Председатель Мущук А.Н.

(подпись)

Минск 2025

Содержание

[ВВЕДЕНИЕ 4](#_Toc198501991)

[1.1. Анализ прототипов 5](#_Toc198501992)

[1.2. Требования к проекту 7](#_Toc198501993)

[1.3 Выводы по разделу: 8](#_Toc198501994)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc198501995)

[2.1. Описание средств разработки 9](#_Toc198501996)

[2.2. Спецификация функциональных требований к программному средству 11](#_Toc198501997)

[2.3. Спецификация функциональных требований 11](#_Toc198501998)

[2.4. Выводы по разделу: 12](#_Toc198501999)

[3. Проектирование программного средства 13](#_Toc198502000)

[3.1. Общая структура 13](#_Toc198502001)

[3.2. Взаимоотношения между классами 13](#_Toc198502002)

[3.3. Модель базы данных 14](#_Toc198502003)

[3.5. Проектирование последовательностей проекта 15](#_Toc198502004)

[4.1. Основные классы программного средства 16](#_Toc198502005)

[4.2. Описание классов и методов программного средства 16](#_Toc198502006)

[4.2.1. Работа с базой данных 16](#_Toc198502007)

[4.2.2. Выполнение входа 16](#_Toc198502008)

[4.2.3. Управление арендами 17](#_Toc198502009)

[4.2.4. Управление статусом пользователей 17](#_Toc198502010)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 18](#_Toc198502011)

[5.1. Тестирование авторизации 18](#_Toc198502012)

[5.2. Тестирование оформления бронирования 18](#_Toc198502013)

[5.3. Тестирование блокировки пользователя 19](#_Toc198502014)

[5.4. Тестирование добавления пользователя 20](#_Toc198502015)

[5.5. Выводы по разделу: 21](#_Toc198502016)

[6. Руководство по установке и использованию 22](#_Toc198502017)

[6.1. Использование 22](#_Toc198502018)

[6.2. Выводы поразделу: 26](#_Toc198502019)

[Заключение 26](#_Toc198502020)

[**Список литературы** 28](#_Toc198502021)

[ПРИЛОЖЕНИЕ А 29](#_Toc198502022)

[ПРИЛОЖЕНИЕ Б 30](#_Toc198502023)

[ПРИЛОЖЕНИЕ В 31](#_Toc198502024)

[ПРИЛОЖЕНИЕ Г 32](#_Toc198502025)

## ВВЕДЕНИЕ

В современном мире средства велопроката становятся все более популярным способом активного отдыха и передвижения для туристов, местных жителей, супружеских пар и любителей здорового образа жизни. Управление такими сервисами требует эффективной системы учета велосипедов, удобного и быстрого способа бронирования и аренды двухколесных транспортных средств.

В рамках данного курсового проекта предлагается разработка программного средства для управления пунктом велопроката с использованием языка программирования C# и технологии Windows Presentation Foundation (WPF). Данное программное обеспечение будет предоставлять администратору пункта проката инструменты для эффективного контроля и управления всеми аспектами его работы.

Целью данного проекта является создание интуитивно понятного и функционального приложения, способного упростить рутинные задачи администрирования велопроката, повысить эффективность работы персонала и обеспечить удобство для клиентов.

В ходе разработки будут рассмотрены основные аспекты проектирования и реализации программного средства, включая архитектуру приложения, пользовательский интерфейс, механизмы управления бронированием велосипедов, а также интеграцию с базой данных для хранения информации о клиентах, доступных велосипедах, бронированиях и комментариях.

В результате успешной реализации данного проекта ожидается создание удобного и мощного инструмента для управления велопрокатом, способного повысить качество обслуживания клиентов, оптимизировать рабочие процессы администрации и увеличить рентабельность пункта проката.

**1**.**Аналитический обзор литературы и формирование требований**

## Анализ прототипов

Были проанализированы цели, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

* + 1. **Velocat**

Velocat – это cовременное программное обеспечение для управления велопрокатом. Программный комплекс включает всё необходимое для эффективного управления: сетка с доступными велосипедами, кратким названием и фотографией, возможность накапливать и использовать бонусы, акции, скидки, промокоды, и другое. Интерфейс представлен на рисунке 1.1.

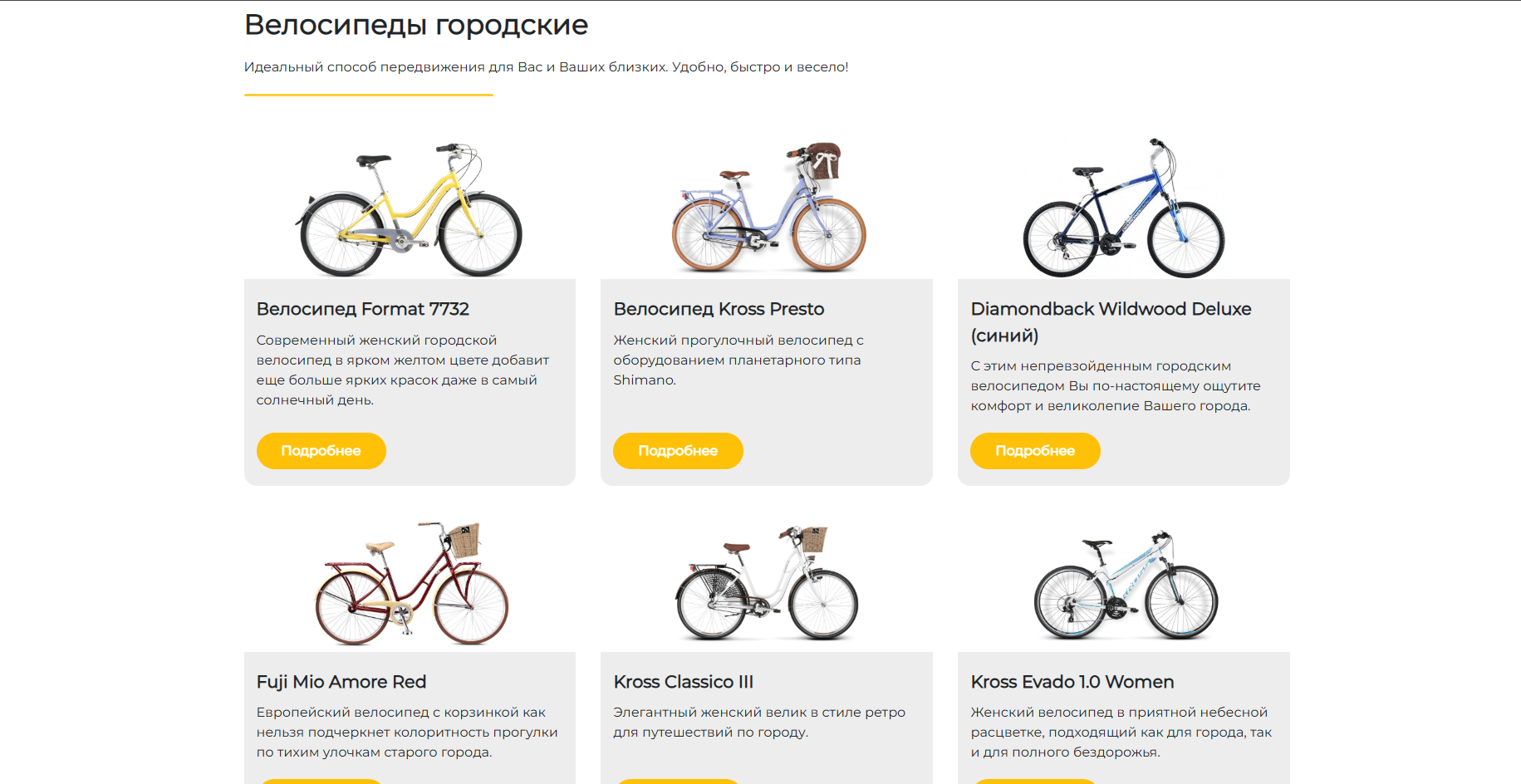


Рисунок 1.1 – Интерфейс Velocat

Явным недостатком является слишком длительный процесс аренды велосипеда и отсутствие возможность бронирования. Это является большим недостатком, так как аренду хочется оформить за несколько кликов.

* + 1. Арентер

Программа Арентер предназначена для автоматизации работы велопроката. По функционалу сервис предоставляет выбор и фильтрацию велосипедов, быструю аренду. Однако, визуально карточки товаров выглядят информационно насыщенными, что может усложнять быстрое восприятие.

Обилие фильтров также может оказаться избыточным для некоторых пользователей. Это видно на рисунке 1.2.

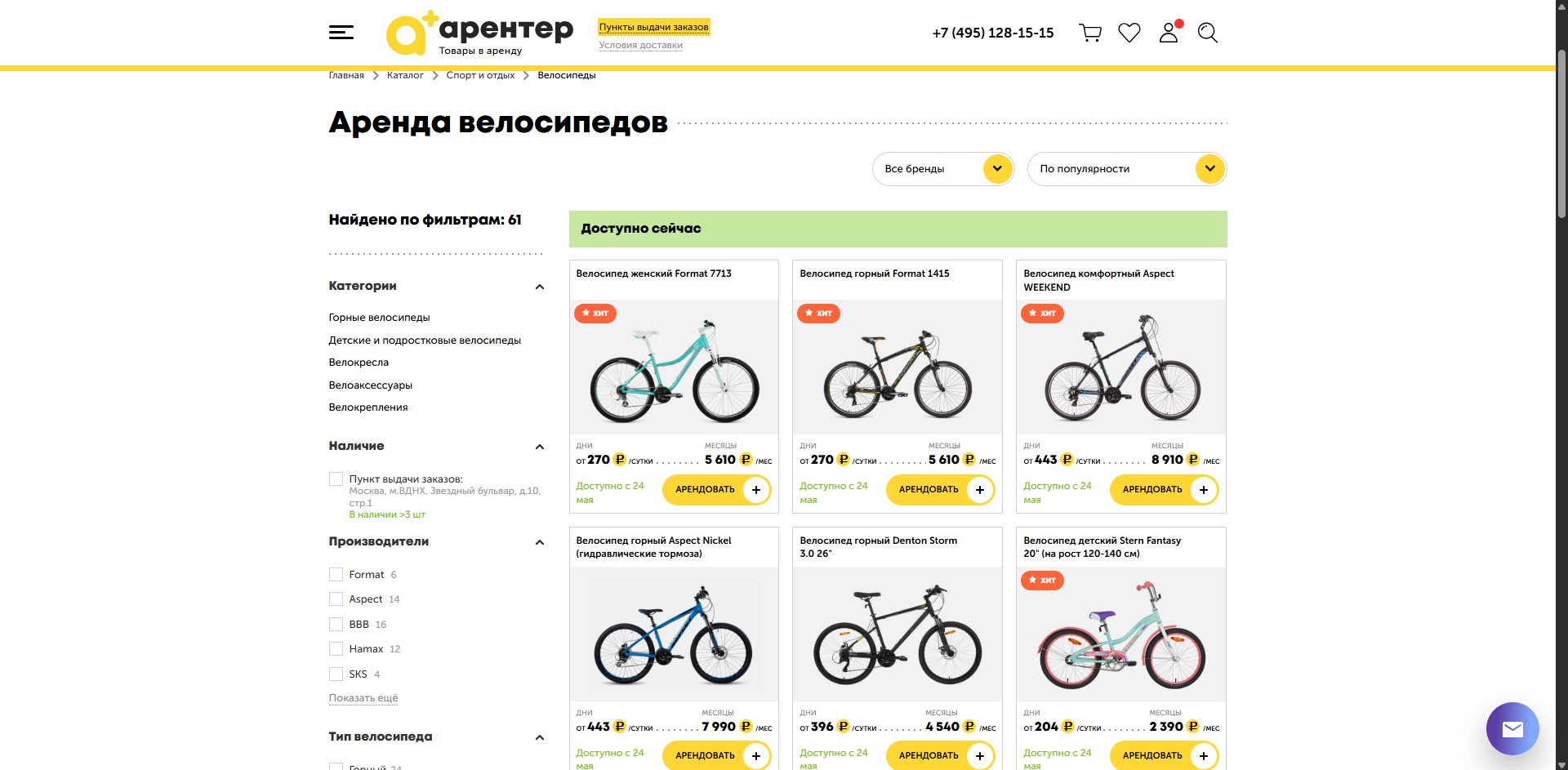


Рисунок 1.2 – Интерфейс Арентер

Кроме того, программное средство обеспечивает быстрый доступ к бронированию велосипеда, полную поддержку Windows 11.

* + 1. Крути педали

Крути педали представляет собой инновационное программное обеспечение, разработанное с учетом потребностей велопроката.

В основе Крути педали лежит идея управления категориями клиентов. Это позволяет вам легко классифицировать ваших клиентов и предлагать им наиболее подходящие услуги. Крути педали также предлагает уникальные возможности для управления пакетами услуг. Вы можете настраивать как сгораемые, так и не сгораемые пакеты услуг, чтобы удовлетворить потребности каждого клиента.

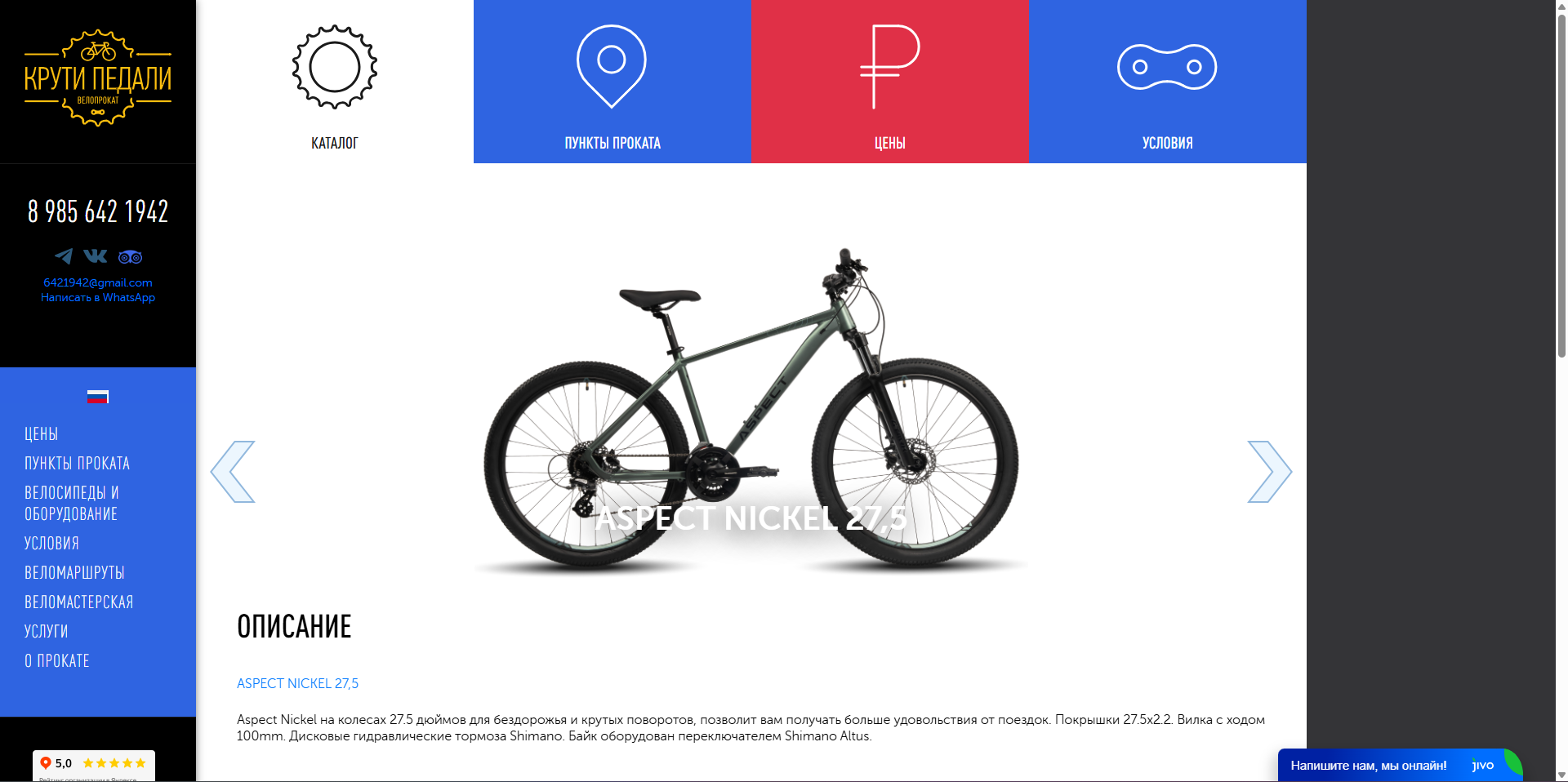
****

Рисунок 1.3 – Интерфейс Крути педали

Основным недостатком Крути педали является узкий выбор доступных для аренды велосипедов и отсутствие возможности оставлять комментарии после аренды.

* + 1. Speedy Go

Speedy Go – это быстрый, удобный и надёжный велопрокат для тех, кто ценит своё время и свободу передвижения. Современные и лёгкие велосипеды обеспечивают комфортную езду по городу, а мгновенное бронирование через приложение позволяет арендовать велосипед всего за пару кликов. Удобные тарифы – поминутные, почасовые и суточные – подойдут для любых нужд.

Интерфейс программы представлен на рисунке 1.5.

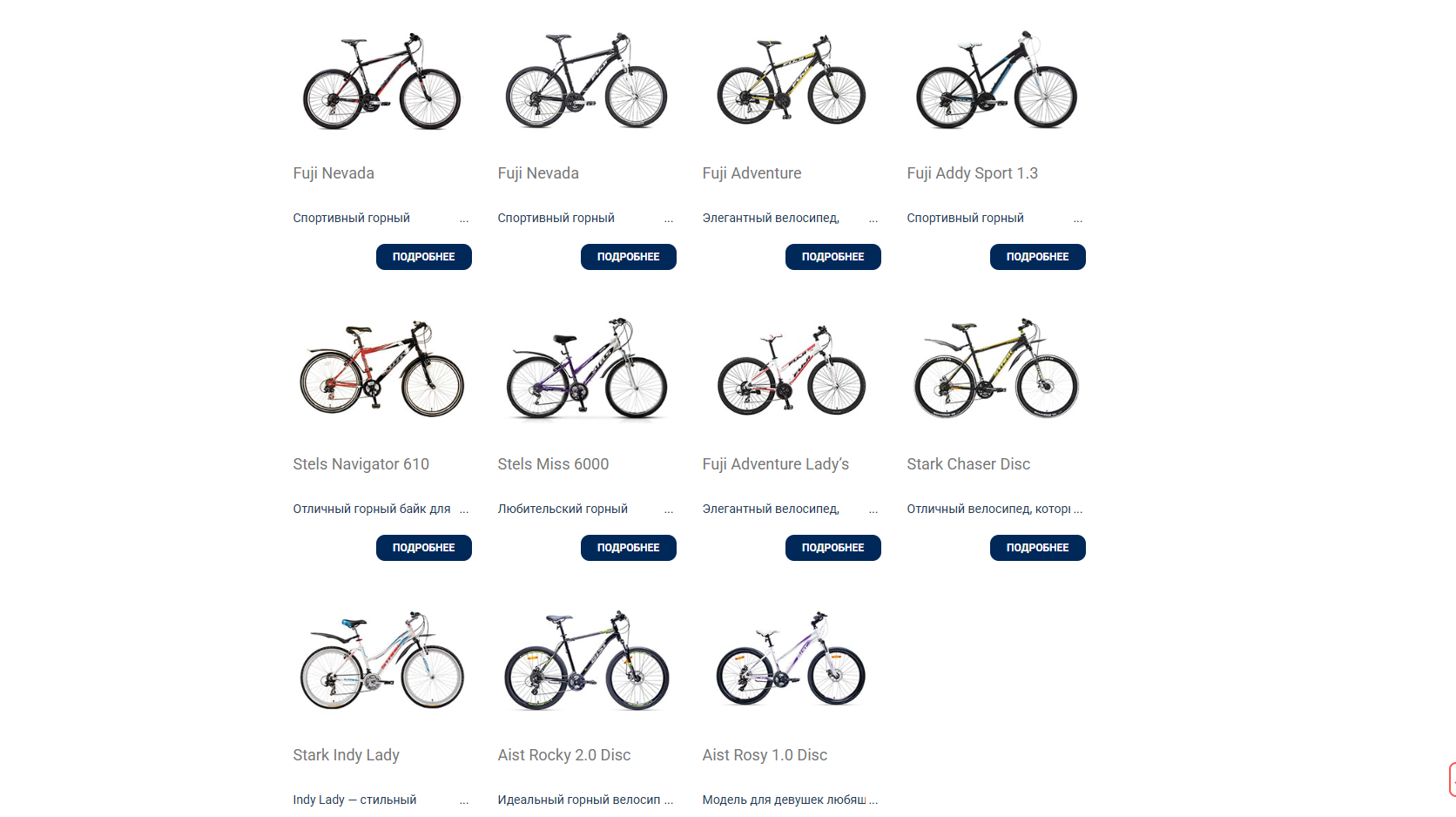
****

Рисунок 1.4 – Интерфейс Speedy Go

Данная система является слишком нагруженной, а потому она не очень хорошо подходит для небольших велопрокатов. Так же в ней отсутствует возможность добавления комментариев.

## Требования к проекту

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* поддержка бронирования в пару кликов;
* возможность оплаты бонусами;
* возможность следить за активными заказами;
* возможность следить за историей аренды;
* возможность оставлять отзывы после аренды;
* ведение подсчета стоимости на основе тарифа и времени.

## Выводы по разделу:

В данном разделе были рассмотрены и проанализированы существующие программные решения в сфере велопроката: Velocat, Арентер, Крути педали и Speedy Go. Каждое из решений обладает своими сильными и слабыми сторонами. Так, Velocat предлагает широкий функционал и систему бонусов, но страдает от неудобного процесса аренды. Арентер обеспечивает быструю аренду и фильтрацию, однако перегруженность интерфейса может затруднить восприятие. Программа Крути педали выгодно выделяется возможностью управления клиентскими категориями и пакетами услуг, но имеет ограниченный выбор велосипедов и не поддерживает обратную связь от пользователей. Speedy Go обеспечивает простоту и скорость аренды, однако система слишком ресурсоёмка для небольших прокатов и также не поддерживает возможность оставления отзывов.

Анализ этих решений позволил выявить наиболее важные функции, которые должны быть реализованы в разрабатываемом программном средстве: простота и скорость бронирования, использование бонусов, отслеживание текущих и прошлых заказов, возможность оставлять отзывы, а также автоматический расчёт стоимости аренды. Формулировка данных требований закладывает основу для разработки более эффективного и удобного сервиса велопроката, учитывающего современные потребности пользователей и недостатки существующих решений.

## 2. 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 Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* база данных MS SQL Server.

1. **Microsoft Visual Studio 2022**

Microsoft Visual Studio 2022 — это интегрированная среда разработки для написания, отладки и сборки кода, а также последующей публикации приложений. Данный продукт позволяет разрабатывать не только консольные, но и десктопные приложения, с использованием таких технологий, как WinForms или WPF.

1. **Программная платформа .NET Core 8.0**

Платформа .NET Core — это созданная Microsoft в 2016 году кроссплатформенная, с открытым исходным кодом технология, предназначенная для создания и выполнения современных приложений. Она поддерживает разработку различных типов приложений, включая веб– приложения, облачные сервисы, десктопные приложения и микросервисы, работая на операционных системах Windows, macOS и Linux.

Основой .NET Core является общеязыковая среда выполнения Common Language Runtime (CLR), которая обеспечивает выполнение кода, написанного на различных поддерживаемых языках программирования, таких как C#, F#, и Visual Basic. .NET Core предлагает модульную архитектуру, где разработчики могут выбирать и использовать только необходимые пакеты, что позволяет создавать более легкие и производительные приложения.

1. **Язык программирования C#**

В качестве языка программирования используется C# – основной язык разработки в .NET Framework. Язык объектно – ориентированный, имеет строгую статическую типизацию, поддерживает перегрузку операторов, указатели на функции– члены классов, атрибуты, события, свойства, исключения. Используется как основной язык в технологии WPF.

1. **Технология WPF**

Для предоставления пользовательского интерфейса и разграничения дизайна и бизнес– логики используется технология Microsoft WPF – аналог WinForms, система для построения клиентских приложений Windows с возможностями взаимодействия с пользователем и графическая подсистема в составе .NET Framework, использующая язык разметки XAML.

1. **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. XAML представляет собой язык декларативного описания интерфейса, основанный на XML.

1. **Технология Entity Framework Core**

Для взаимодействия с базой данных в приложении WPF используется технология Entity Framework Core — кроссплатформенная и расширяемая версия технологии Entity Framework с открытым исходным кодом, которая активно развивается и дополняется. В то время как традиционные средства ADO.NET обеспечивают создание подключений, выполнение команд и работу с другими объектами для взаимодействия с базами данных, Entity Framework предлагает более высокий уровень абстракции. Эта технология позволяет разработчикам абстрагироваться от конкретной базы данных и работать с данными независимо от типа хранилища. Entity Framework Core является последней версией Entity Framework 7, которая совместима с .NET Core и продолжает активно развиваться.

Entity Framework Core предоставляет три основных способа взаимодействия с базой данных:

* database First: Entity Framework Core создает набор классов, отражающих модель существующей базы данных;
* model First: разработчик сначала создает модель данных, на основе которой Entity Framework Core затем генерирует реальную базу данных на сервере;
* code First: разработчик создает классы моделей данных, которые будут храниться в базе данных, а затем Entity Framework Core генерирует базу данных и ее таблицы на основе этих моделей.

1. **MS SQL Server**

Для организации баз 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. Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос 10 на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения. Основной используемый язык запросов – Transact-SQL – реализован на структурированном языке запросов (SQL) с расширениями.

# **Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* авторизация;
* просмотр доступных велосипедов;
* управление состоянием бронирования;
* просмотр состояния бронирования;
* оформление бронирования
* просмотр истории заказов;
* добавление комментариев.

Для администратора:

* добавление новых велосипедов;
* редактирование информации о велосипедах;
* удаление велосипедов;
* просмотр всех бронирований;
* изменение статуса бронирования;
* управление статусом пользователя;
* модерация отзывов.

# **Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно база данных описана в следующем разделе.

В программном средстве необходимо реализовать авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Введенные данные, успешно прошедшие валидацию, сверяются с данными в базе данных.

Пользователь может управлять арендой и комментариями. Все сведения должны автоматически обновляться в базе данных.

Администратор обладает возможностью добавлять, изменять и удалять велосипеды, управлять статусом бронирования и модерировать комментарии.

## 2.4. Выводы по разделу:

В разделе были определены инструменты, которые будут использоваться в программном средстве. К таким относятся:

* программная платформа .NET Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* база данных MS SQL Server.

## 3. Проектирование программного средства

# **Общая структура**

Программное средство использует архитектуру MVVM. Программное средство имеет следующую структуру, представленную на рисунке 3.1.

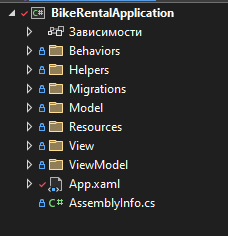


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| App.config | Файл с параметрами проекта. |
| App.xaml | Файл с общими ресурсами проекта. |
| Папка Behaviors | Папка, описывающая поведение при нажатии. |
| Папка Helpers | Папка, содержащая классы, необходимые в работе других классов. |
| Папка Migrations | Папка, содержащая классы миграций базы данных Entity Framework Core. |
| Папка Model | Папка, содержащая классы, которые являются сущностями базы данных. |
| Папка Resources | Папка, содержащая статические данные (фото). |
| Папка View | Папка, содержащая файлы разметок и классы соответствующих страниц. |
| Папка ViewModel | Папка, содержащая классы являющимися контекстами для страниц. |

# **Взаимоотношения между классами**

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в Приложении А.

# **Модель базы данных**

Для реализации поставленной задачи была создана база данных BikeRentalApplication. Для ее создания использовалась система управления реляционными базами данных MSQL Server. База данных состоит из таблиц, представленных на рисунке 3.3.

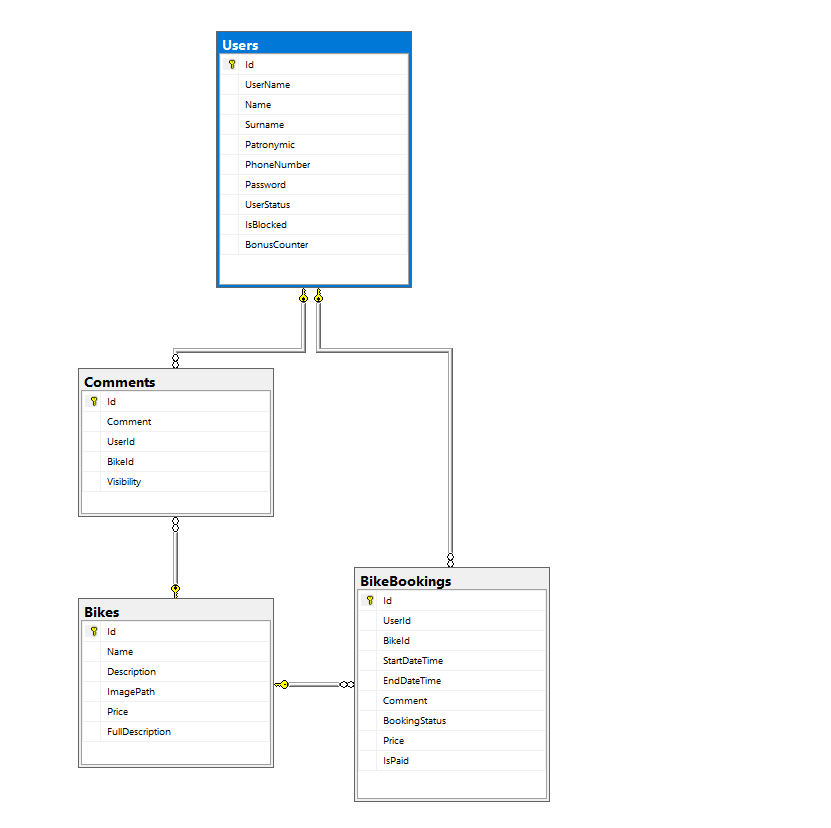


Рисунок 3.3 – Логическая модель базы данных BikeRentalApplication

В базе данных находится 4 таблицы: Users, Bikes, BikeBooking, Comments. При этом таблицы BikeBooking и Comments являются сводными таблицами, между таблицами Users и Bikes . Таблицы связаны между собой следующим образом:

* таблица Users содержит информацию о пользователях;
* таблица Bikes содержит информацию о велосипедах;
* таблица BikeBooking содежрит информацию о бронировании, связывая пользователя с велосипедом;
* таблица Comments содержит данные о комментариях, связывая пользователя с велосипедом.

**3.4. Проектирование функционала проекта**

Для общего представления функционального назначения системы используется диаграмма использования, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. На диаграмме использования применяются два типа основных сущностей: варианты использования и группы пользователей.

Разные группы пользователей в диаграмме называются актёрами, и обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействии с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в приложении Б.

# **3.5. Проектирование последовательностей проекта**

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML– диаграмма – диаграмма последовательностей. Данная 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.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками. Диаграмма последовательностей представлена в приложении В.

**3.6.** Выводы по разделу:

В разделе была выбрана архитектура проекта, которая задаёт общий план проектирование продукта. Также для программного средства были спроектированы сущности базы данных, на основе которых получены их аналоги в языке программирования C#.

**4. Реализация программного средства**

# **Основные классы программного средства**

Для выполнения технических задач программного средства должны быть реализованы следующие функции, соответствующие им классы и методы:

* работа с базой данных;
* выполнение входа;
* управление арендами;
* управление статусом пользователей;

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы, их функционал и реализация.

# **Описание классов и методов программного средства**

## Работа с базой данных

Независимо от того, в какой части приложения находится пользователь, ViewModel всегда связан с базой данных. Для реализации методов работы с базой данных в программе существует класс DataWorker.

Этот класс содержит методы для четырех направлений:

* пользователи – методы CreateUserAsync(), DeleteUser(), GetAllUsers(), GetUserById();
* велосипеды – методы CreateBike(), DeleteBike(), EditBike(), GetAllBikes();
* бронирования – методы CreateBikeBooking(), DeleteBikeBooking(), GetAllBookings();
* комментарии – методы AddComment(), DeleteComment(), GetAllComments().

Код класса DataWorker представлен в приложении Г.

## Выполнение входа

Вход в приложение осуществляется в отдельном окне – AuthWindow. В своём контексте данное окно ссылается на ViewModel авторизации AuthVM. В данном классе содержится:

* конструктор AuthVM(), в котором идёт привязка команд и вызывается метод UpdateTexts(), загружающий необходимый текст;
* метод Login(), отвечающий за авторизацию пользователя. В нём вызываются методы валидации и получения информации, такие как AuthenticateUser(), CheckBloking(), GetUserRole(), GetUserByUserName();
* метод Register(), отвечающий за регистрацию пользователя. В нем проводится валидация на допустимость значений и вызывается метод создания пользователя CreateUserAsync();
* методы OpenAdminBikeWindowMethod(), OpenMainWindowMethod(), предназначенные для открытия соответствующих окон.

Код класса AuthVM представлен в приложении Г.

## Управление арендами

После успешной авторизации пользователь попадет на окно MainWindow, где доступны все велосипеды. После выбора понравившегося велосипеда пользователь может нажать на кнопку Забронировать и откроется окно BikeBookingWindow, контекст которого связан с классом BikeBookingVM.

Класс BikeBookingVM представляет собой ViewModel, управляющую логикой окна бронирования велосипеда. В конструкторе происходит инициализация выбранного велосипеда и идентификатора пользователя, установка начальных значений даты и времени бронирования, привязка команд подтверждения и отмены, а также вызов метода расчета стоимости бронирования.

Метод ExecuteConfirmBooking() отвечает за создание бронирования: формируются дата и время начала и окончания, обрабатывается оплата (обычная или бонусными баллами), сохраняются данные через DataWorker, отображается сообщение и закрывается окно.

Метод UpdateTotalCost() выполняет динамический расчет стоимости на основе введённых дат, времени и цены велосипеда.

Метод CanExecuteConfirmBooking() проверяет, что дата и время окончания бронирования корректны.

Метод ExecuteCancel() обрабатывает отмену бронирования без сохранения.

## Управление статусом пользователей

После успешной авторизации в роли admin существует возможность управлять статусом пользователей. Для этого в классе AdminVM определен метод SetIsBlockedExecute(), который передает выбранного пользователя в метод работы с базой данных ChangeIsBlockedUser(), в котором происходит изменение статуса.

**4.3.** Выводы по разделу:

В разделе описан процесс успешной реализации программного средства, спроектированного в предыдущий разделах. Были описаны классы и методы этих классов, которые задавали работу программного средства.

## 5. Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1. Тестирование авторизации

При авторизации программное средство учитывает многие факторы. Одним из таких является проверка корректности заполнения полей. Например, поля логина и пароля не могут быть пустыми и не могут содержать информацию более 50 символов. Пример попытки ввода некорректных данных представлен на рисунке 5.1.

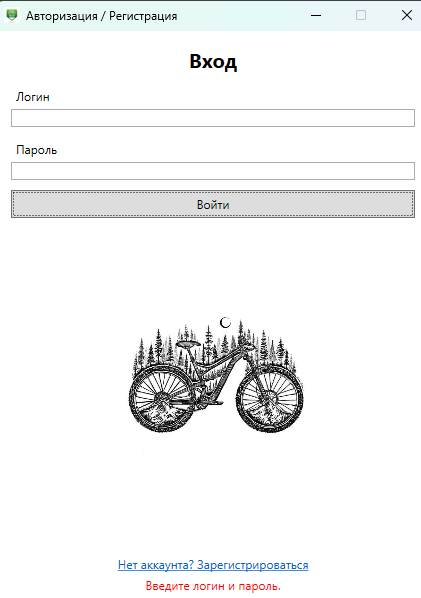
****

Рисунок 5.1 – Обработка пустых полей

Таким образом программное средство подсказывает пользователю, что ему необходимо изменить чтобы успешно войти в систему.

## 5.2. Тестирование оформления бронирования

В форме оформления бронирования применяются следующие правила валидации:

* все поля обязательны к заполнению;
* дата начала не должна быть позже даты конца;
* бонусов должно быть достаточно для оплаты при оплате бонусами.

Пример тестирования системы валидации данной формы приведен на рисунке 5.3.

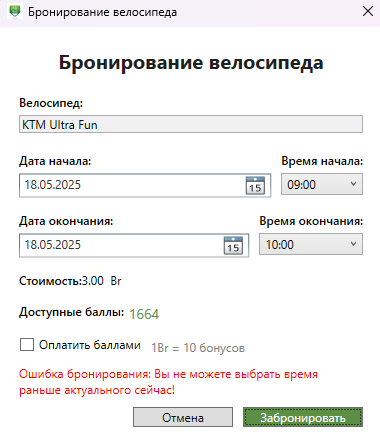


Рисунок 5.2 – Обработка полей бронирования

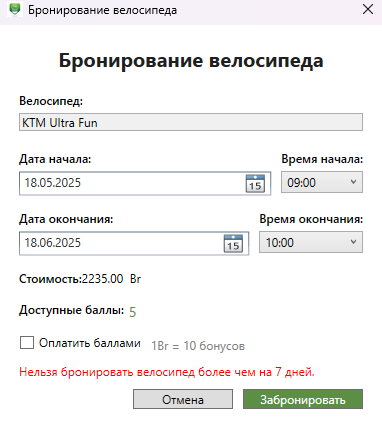


Рисунок 5.3 – Обработка полей бронирования

Таким образом, пользователь не сможет указать дату конца бронирования раньше даты начала, так как такое поведение подозрительно и, предположительно, не соответствует ожидаемому результату.

## 5.3. Тестирование блокировки пользователя

В случае недобросовестных действий админ имеет возможность заблокировать конкретного пользователя. В этом случае программное средство изменит статус пользователя на заблокирован и при следующей авторизации пользователь увидит сообщение, которое представлено на рисунке 5.4.

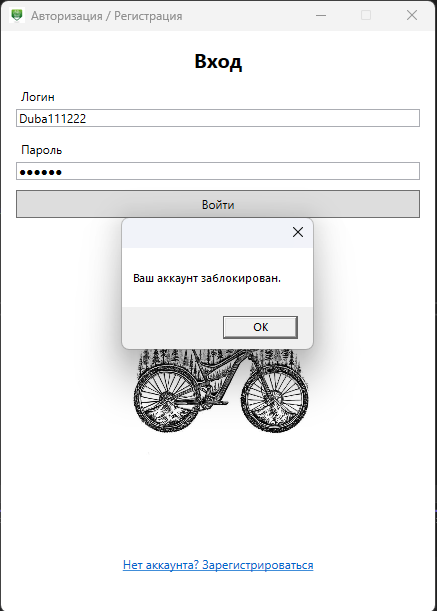


Рисунок 5.4 – Сообщение о блокировке

Благодаря такого рода проверке, программное средство позволяет поддерживать порядок и обеспечивать безопасность использования.

## 5.4. Тестирование добавления пользователя

При регистрации пользователь должен соблюдать множество правил, так как процесс регистрации – это основа взаимодействия с приложением для новых пользователей. К таковым правилам можно отнести:

* заполнить все поля;
* минимальная длина каждого поля 2 символа;
* максимальная длина каждого поля 50 символов;
* минимальная длина пароля 5 символов;
* номер должен соответствовать формату;
* пароль и повтор пароля должны совпадать;
* имя, фамилия, отчество должны содержать только буквы;
* в базе данных не должно быть пользователей с таким же логином.

В случае несоблюдения хотя бы одного правила, программное средство будет выводить сообщения об ошибках, которое представлено на рисунке 5.5.

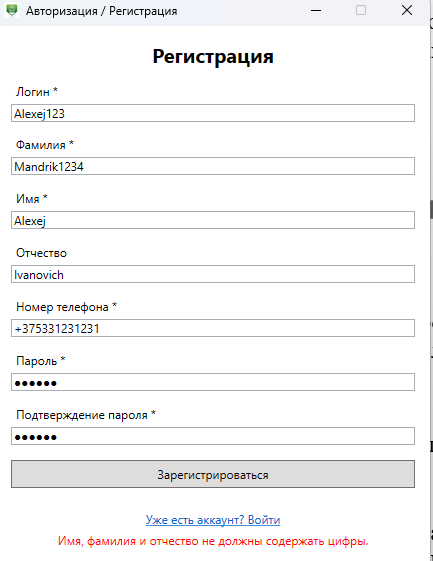


Рисунок 5.5 – Обработка полей добавления сотрудника

## 5.5. Выводы по разделу:

В разделе описан процесс тестирования продукта. Были проверены случаи негативного и положительного тестирования. Были проверены все случаи валидации вводимых пользователем данных.

## Руководство по установке и использованию

## Использование

При запуске программного средства пользователь попадает на страницу авторизации, содержащая форму входа.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, для этого ему необходимо ввести корректные логин и пароль, после чего пользователь будет перенаправлен на главную страницу.

Окно авторизации представлено на рисунках 6.1.

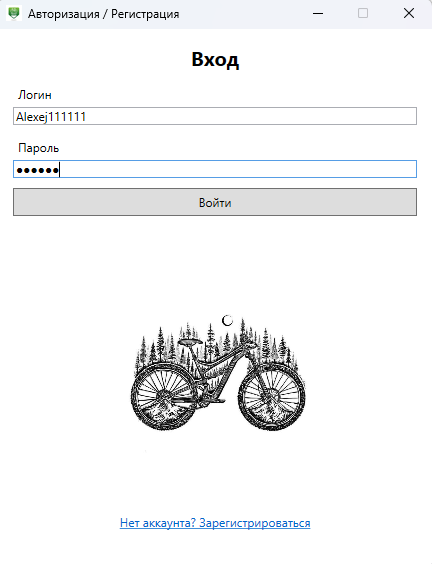


Рисунок 6.1 – Окно аворизации

Находясь на главной странице пользователь получает доступ к каталогу велосипедов велопроката. Главная страница представлена на рисунке рисунке 6.2.

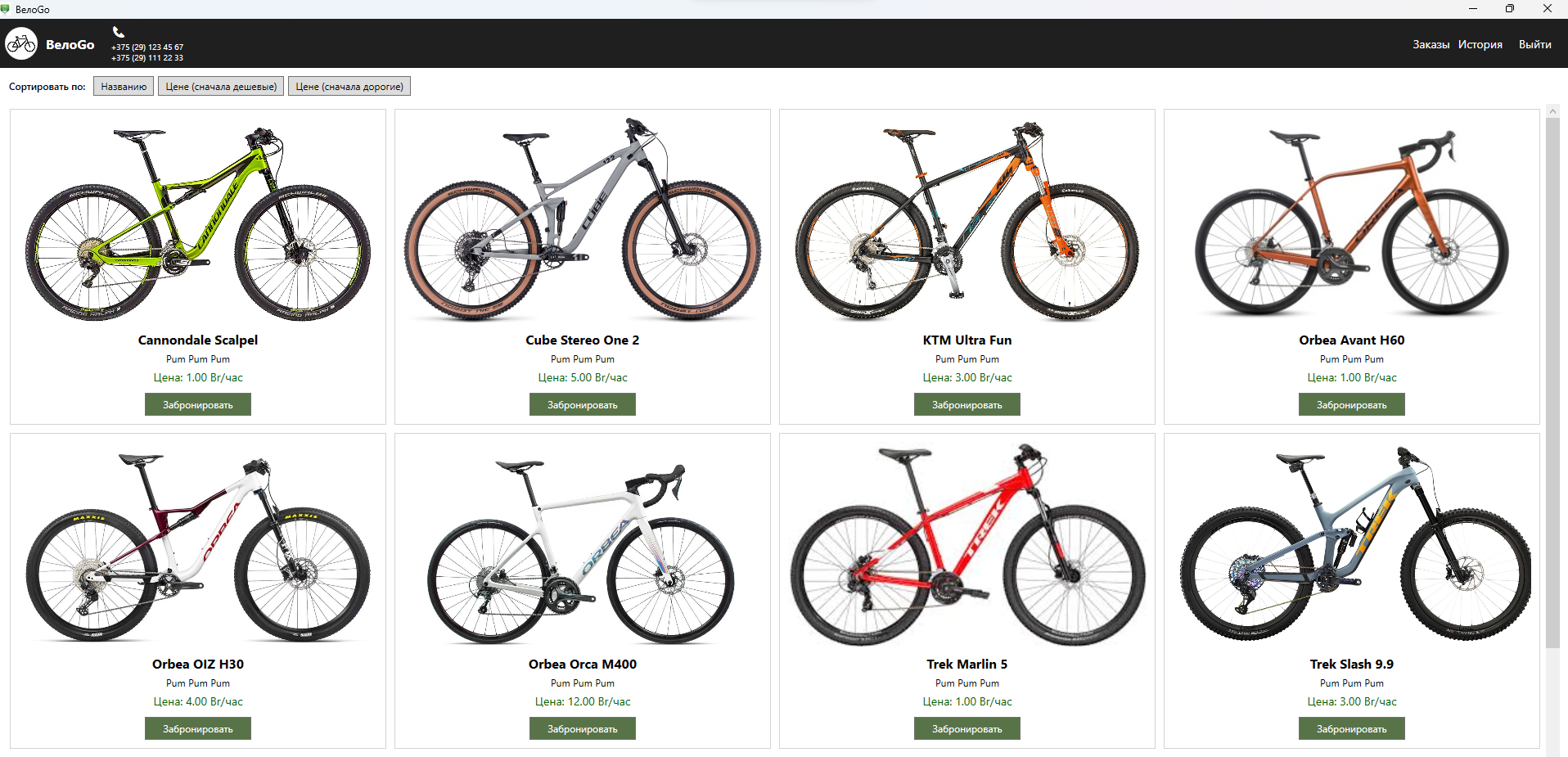


Рисунок 6.2 – Главная страница пользователя

Вверху страницы находятся критерии для фильтрации, по которым можно сортировать доступные велосипеды. В правом верхнем углу находятся кнопки маршрутизации, которые предоставляют возможность перехода в окна Заказы, История, или же предоставляют возможность выйти из приложения.

В шаговой доступности находится кнопка Забронировать, при нажатии на которую можно в пару кликов забронировать велосипед. Форма бронирования представлена на рисунке 6.3.

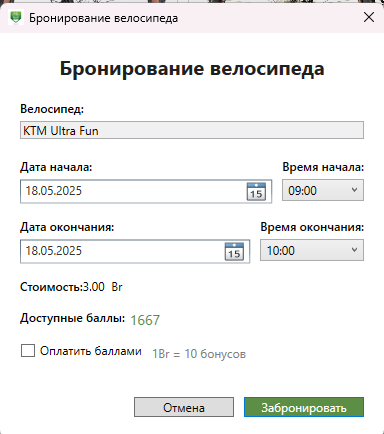


Рисунок 6.3 – Окно бронирования

После успешного бронирования пользователь может перейти в окно заказы, где будут находиться все забронированные и активные заказы. При желании в этом окне пользователь может отменить бронирование определенного велосипеда. Окно заказы предсталвено на рисунке 6.4.

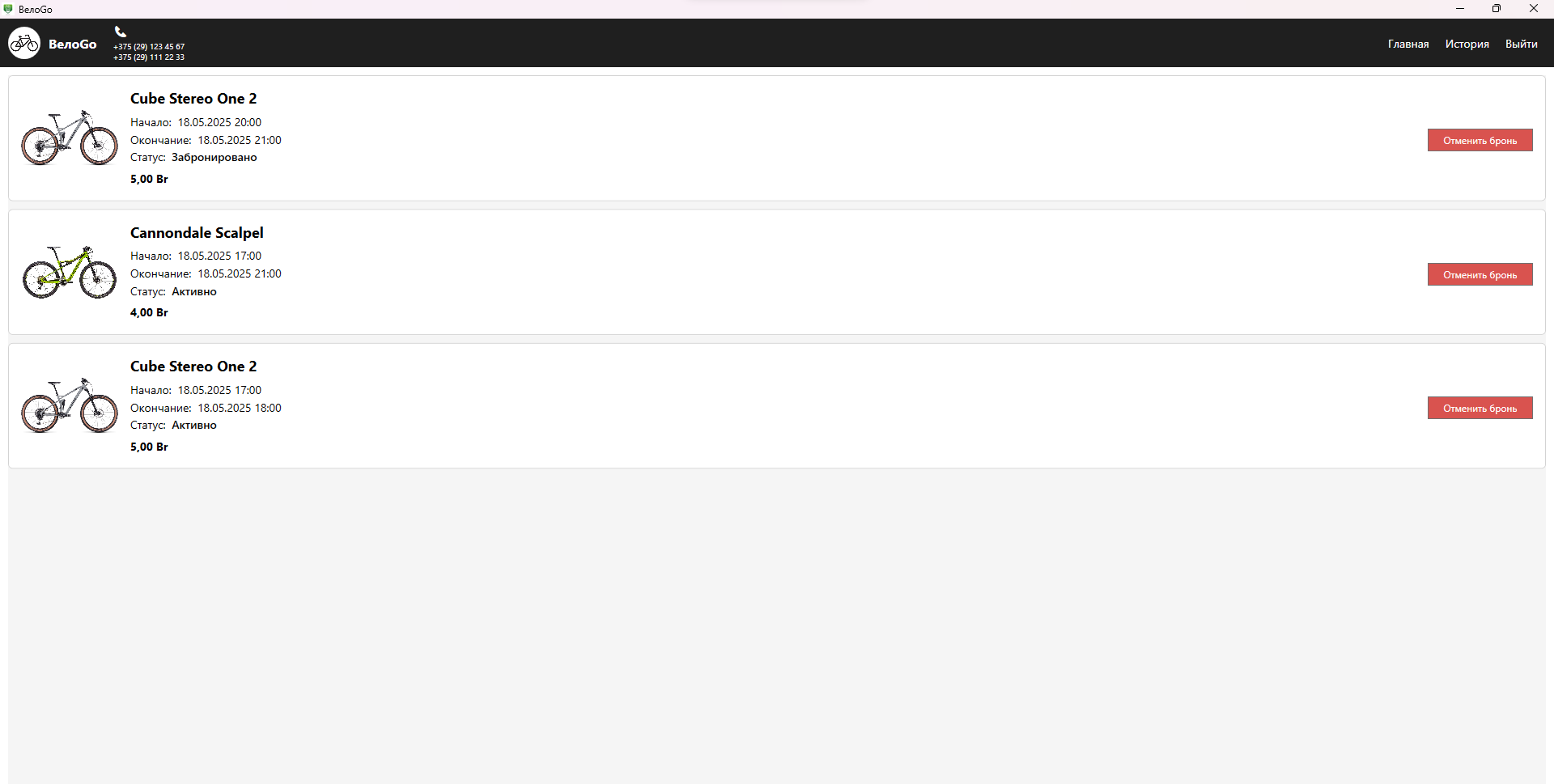


Рисунок 6.4 – Окно заказы

Так же пользователь может перейти в окно истроия, в котором будут находиться только завершенные заказы. Пример окна история представлен на рисунке 6.5.



Рисунок 6.5 – Окно история

В этом же окне пользователь может открыть окон комментариев, где можно описать своё впечатление о велосипеде. Окно комментариев представлено на рисунке 6.6.

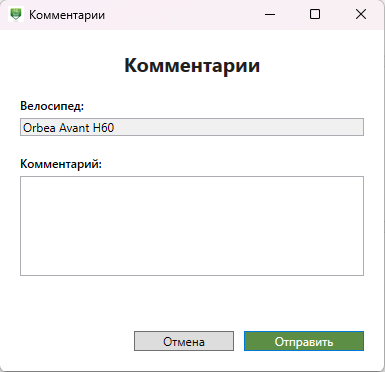


Рисунок 6.6 – Окно комментарии

Окно адниминстратора не сильно отличается от окна пользователя. Вверху окна содержатся кнопки добавления, удаления и изменения выбранного администратором велосипеда. Также в правом верхнем углу пристуствуют кнопки маршрутизации. С их помощью можно перейти в окно Пользоваетели и окно Заказы, или же покинуть приложение.

Внешний вид страницы аднимистратора представлен на рисунке 6.3.

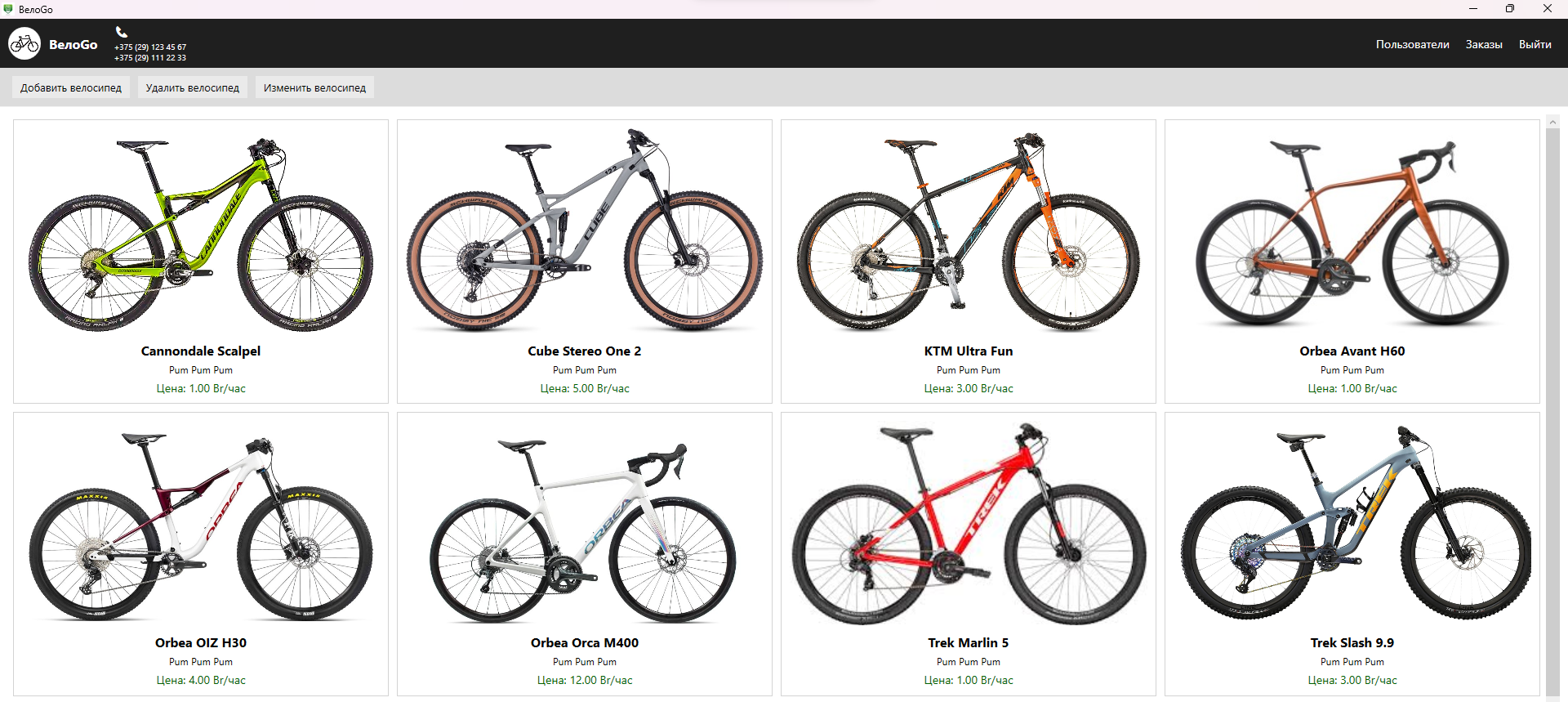


Рисунок 6.7 – Страница администратора

Администратор имеет возможность модерировать закзаы. В окне заказы выводятся все бронирования всех пользователей. Администратор может изменить статус заказа, если это необходимо. Окон заказы предсталвено на рисунке 6.8.

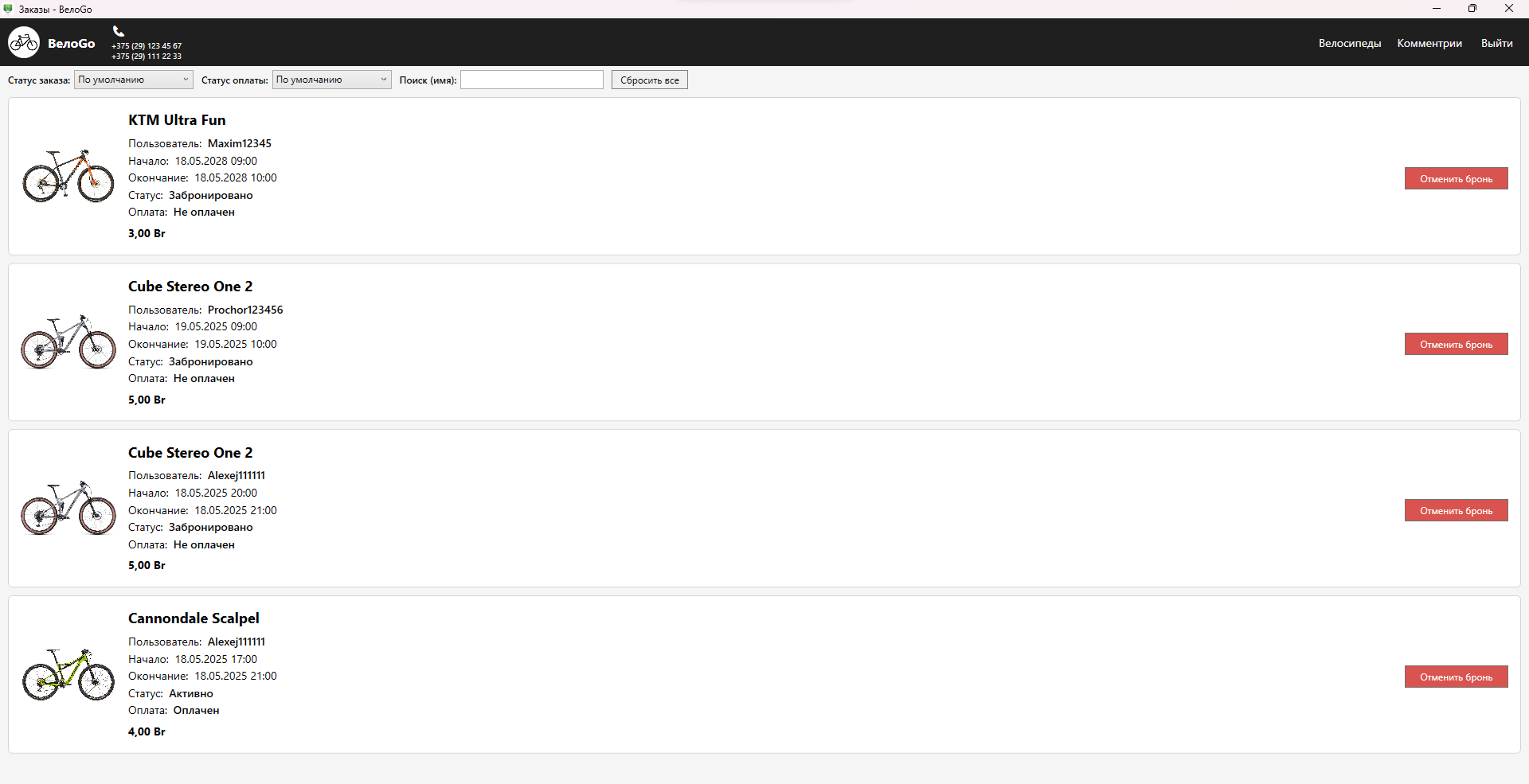


Рисунок 6.7 – Окно заказы

Администратор ответственен в том числе и за модерацию отзывов. Он может изменить статус видимости или же вовсе удалить комментарий из базы данных. Окно комментариев представлено на рисунке 6.8.

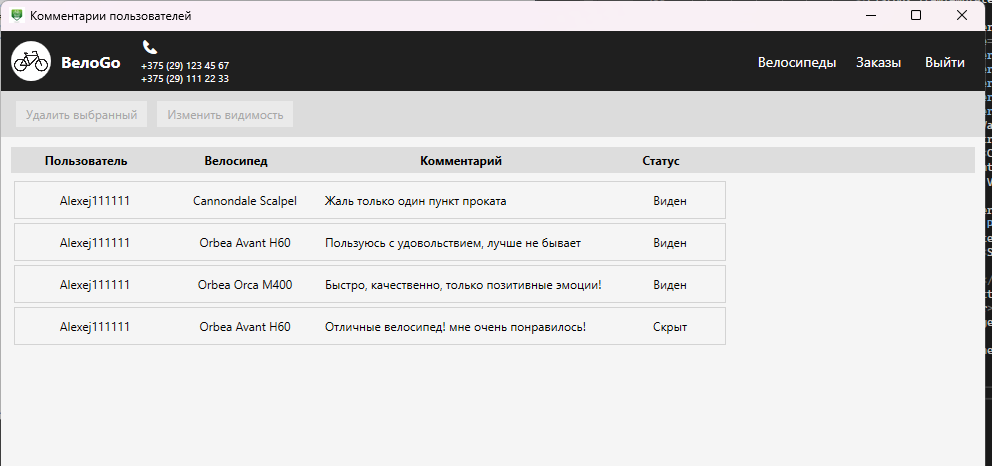


Рисунок 6.8 – Окно заказы

Администрирование комментариев позволяет поддерживать порядок в приложении.

## Выводы по разделу:

В разделе описан процесс использования программного средства. В нем пользователя инструктирую, как воспользоваться каждой из функции, которые предоставляет программное средство.

## Заключение

В ходе выполнения курсового проекта было разработано программное средство для управления компьютерным клубом на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке программного средства были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

* создана база данных;
* реализована авторизация и регистрация;
* просмотр доступных велосипедов;
* управление состоянием бронирования;
* просмотр состояния бронирования;
* оформление бронирования
* просмотр истории заказов;
* добавление комментариев.
* редактирование информации о велосипедах;
* удаление велосипедов;
* просмотр всех бронирований;
* изменение статуса бронирования;
* управление статусом пользователя;
* модерация отзывов.

Тестирование программы показало, что она работает корректно и выполняет все свои функции.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список литературы**

1. Онлайн– ресурс “smartshell.gg” [Электронный ресурс] – Режим доступа smartshell.gg – Дата доступа: 25.04.2025
2. Онлайн– ресурс “astrumsoft.com” [Электронный ресурс] – Режим доступа astrumsoft.com – Дата доступа: 25.04.2025
3. Онлайн– ресурс “lightshell.ru” [Электронный ресурс] – Режим доступа lightshell.ru – Дата доступа: 27.04.2025
4. Онлайн– ресурс “ggLeap” [Электронный ресурс] – Режим доступа esportsentertainmentgroup.com – Дата доступа: 26.04.2025
5. Онлайн– ресурс “smartlaunch” [Электронный ресурс] – Режим доступа smartlaunch.com – Дата доступа: 26.04.2025
6. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
7. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.04.2025
8. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity– framework/6/level1/ – Дата доступа 23.04.2025

## ПРИЛОЖЕНИЕ А

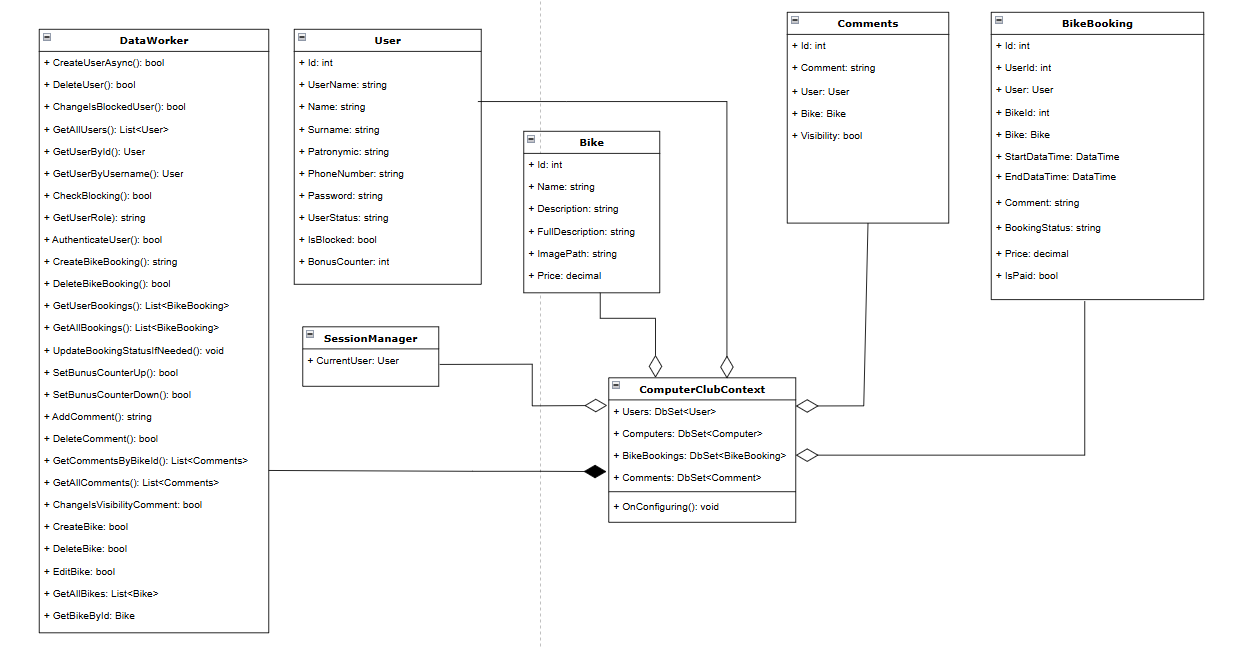


Рисунок А.1 – диаграмма классов

## ПРИЛОЖЕНИЕ Б

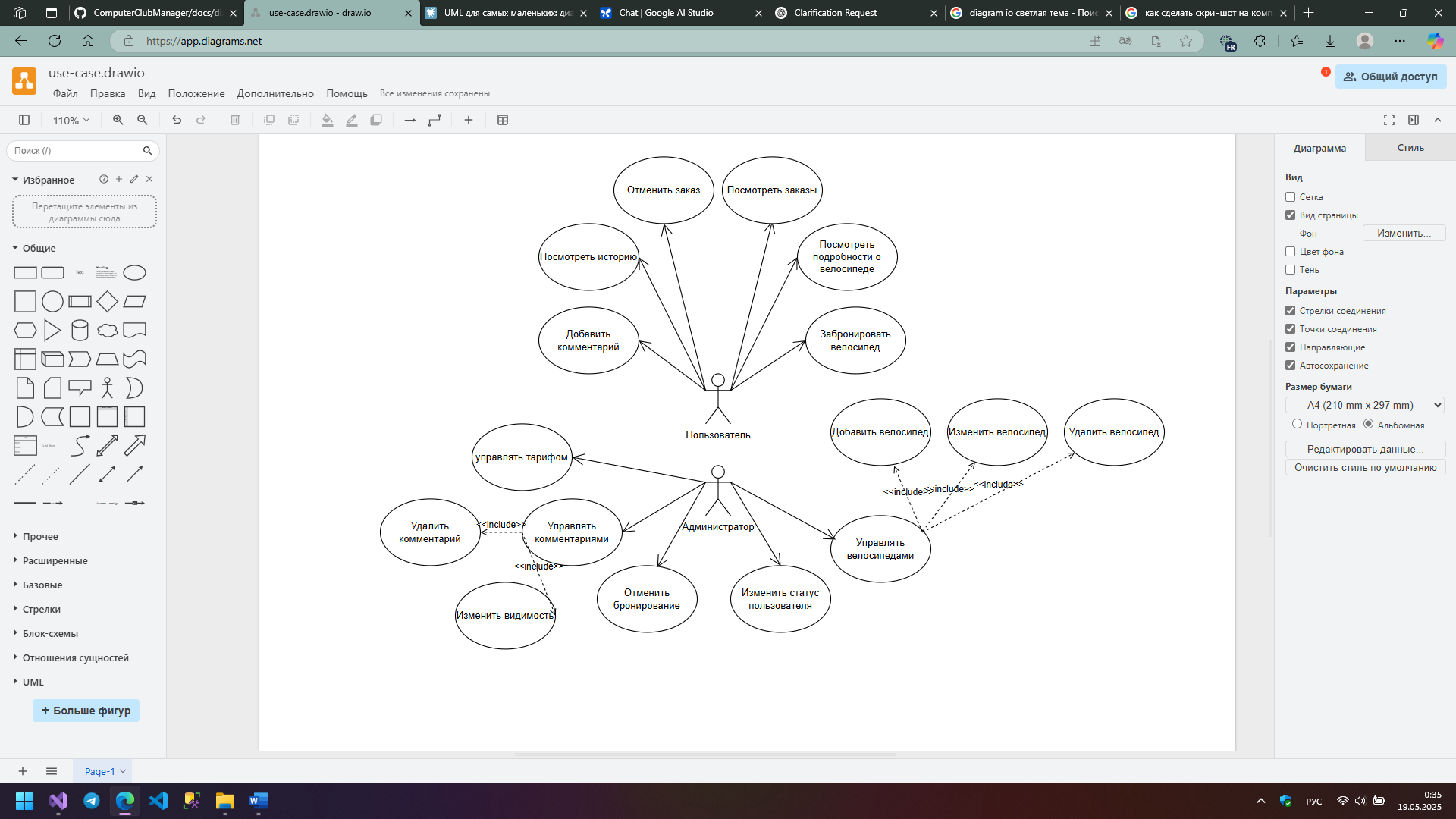
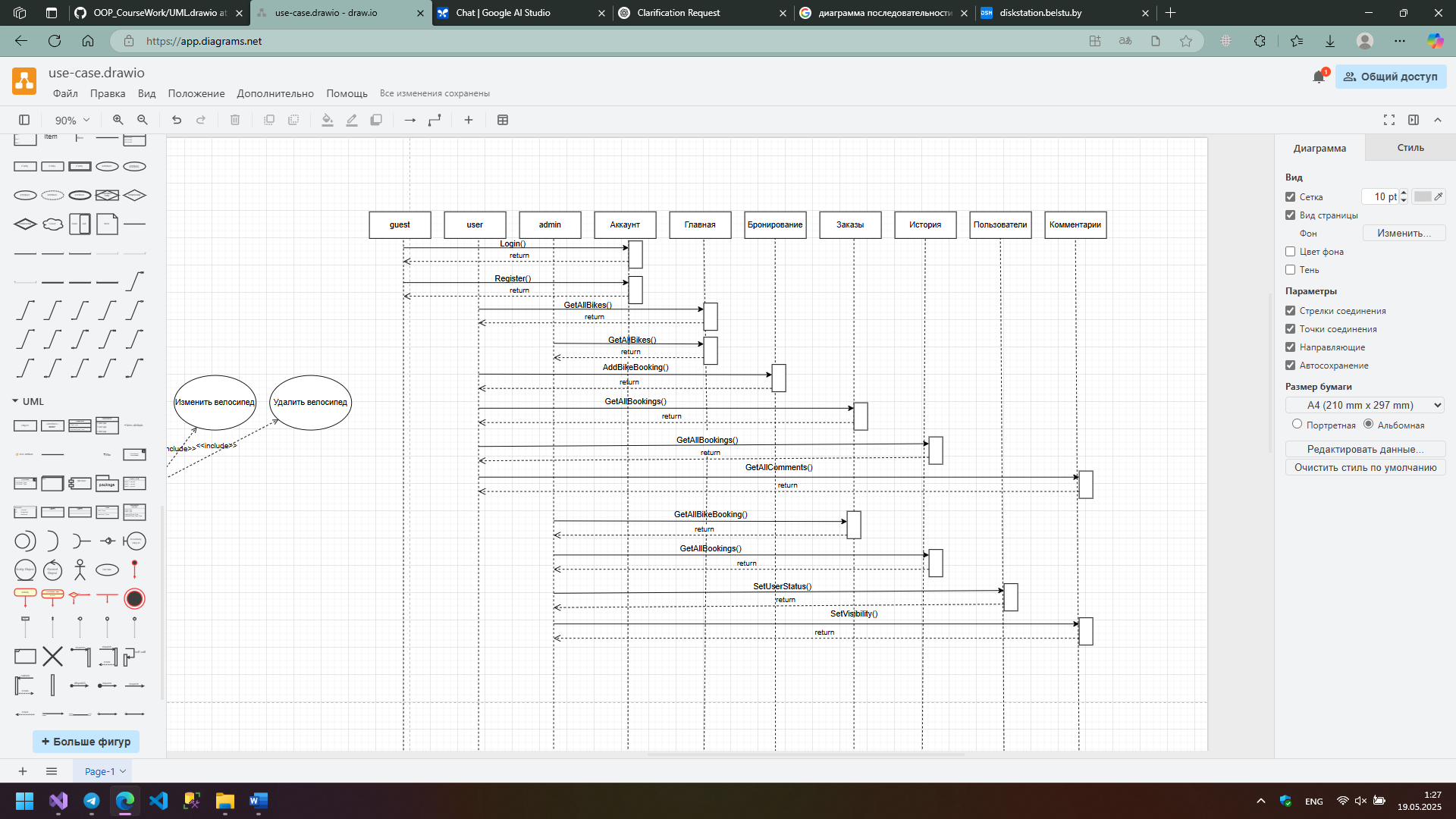


Рисунок Б.1 – диаграмма вариантов использования

ПРИЛОЖЕНИЕ В

  
Рисунок В.1 – диаграмма последовательности

## ПРИЛОЖЕНИЕ Г