МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное проектирование и программирование»

Тема: Программное средство «Велопрокат»

Исполнитель

студент 3 курса группы 7 Костюкевич Виктор Петрович

(Ф.И.О.)

Руководитель работы преподаватель-стажер Якунович А.В.

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

Курсовой проект защищен с оценкой

Председатель

(подпись)

Минск 2023

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc154059396)

[1 Аналитический обзор литературы и постановка задачи 5](#_Toc154059397)

[1.1 Обзор прототипов 5](#_Toc154059398)

[1.2 Постановка задачи 9](#_Toc154059399)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc154059400)

[2.1 Описание средств разработки 10](#_Toc154059401)

[2.2 Описание разрабатываемой функциональности программного средства 11](#_Toc154059402)

[3 Проектирование программного средства 12](#_Toc154059403)

[3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства 12](#_Toc154059405)

[3.2 Разработка общей схемы приложения 13](#_Toc154059406)

[3.3 Проектирование архитектуры приложения 14](#_Toc154059407)

[3.4 Проектирование базы данных 16](#_Toc154059408)

[4 Создание (реализация) программного средства 18](#_Toc154059409)

[4.1 Регистрация 18](#_Toc154059411)

[4.2 Авторизация 18](#_Toc154059412)

[4.3 Функции администратора: Добавление велосипеда 18](#_Toc154059413)

[5. Тестирование и анализ работоспособности и полученных результатов 20](#_Toc154059414)

[5.1 Тестирование авторизации и регистрации 20](#_Toc154059415)

[5.2 Тестирование основного функционала 21](#_Toc154059416)

[6. Руководство по использованию 23](#_Toc154059417)

[ЗАКЛЮЧЕНИЕ 26](#_Toc154059418)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 27](#_Toc154059419)

[Приложение А 28](#_Toc154059420)

[Приложение Б 29](#_Toc154059421)

[Графический материал 32](#_Toc154059422)

[Диаграмма вариантов использования 32](#_Toc154059423)

[Логическая схема базы данных 33](#_Toc154059424)

[Диаграммы классов 35](#_Toc154059425)

[Диаграммы последовательностей 36](#_Toc154059426)

ВВЕДЕНИЕ

Шеринг и прокат - это формы совместного потребления, которые становятся все более популярными в наше время. Они позволяют людям экономить деньги, ресурсы и пространство, а также получать доступ к различным товарам и услугам без необходимости их покупки. Одной из причин, почему люди не стремятся покупать себе что-то, но готовы пользоваться услугами временного пользования, является изменение ценностей и предпочтений. Многие люди, особенно молодые, больше ценят свободу, мобильность и разнообразие, чем владение вещами. Они хотят иметь возможность выбирать, что, когда и как им использовать, а не быть привязанными к одному предмету или месту. Кроме того, многие люди осознают, что покупка вещей не всегда приносит удовлетворение и счастье, а иногда даже создает дополнительные проблемы и заботы.

Информационные технологии играют важную роль в развитии шеринга и проката, так как облегчают взаимодействие между поставщиками и потребителями. С помощью интернета, мобильных приложений и платформ люди могут легко находить, заказывать и оплачивать различные товары и услуги, а также давать и получать отзывы и рекомендации. Информационные технологии также помогают повышать доверие и безопасность в сфере шеринга и проката, так как предоставляют различные механизмы идентификации, верификации и защиты данных.

Приложение Велопрокат представляет собой механизм для бронирования велосипедов, просмотра отзывов о велосипедах, а также предлагает ряд возможностей для администрирования.

Приложение «Велопрокат» предназначено для аренды велосипедов. Оно предоставляет клиентам возможность оформления подписки на услуги, оплаты велопроката. Данное приложение предназначено для пользования людьми, желающими пользоваться велосипедом, но не имеющими желания или возможности купить велосипед.

«Велопрокат» предназначено для использования в качестве десктоп-приложения. Сервис обеспечивает простой и удобный способ управления платежами, бронированием, информацией о велосипедах для администраторов

Приложение «Велопрокат» позволяет пользователям выбирать велосипед для бронирования, оплачивать услугу велопроката, в функции администратора входит управление данными о велосипедах, модерация отзывов и т. п.

1 Аналитический обзор литературы и постановка задачи

В качестве источников литературы были использованы материалы с сайта <https://metanit.com>, лекционные материалы по предметам «Объектно-ориентированное программирование», «Базы данных» и «Паттерны проектирования»

1.1 Обзор прототипов

Большинство сервисов для аренды велосипедов находится в Интернете, ниже будут описаны самые популярные русскоязычные сайты.

Велобайк - это сеть веломагазинов, расположенная в городе Минск. Это сайт, где вы можете купить велосипеды, аксессуары, запчасти и сопутствующие товары по доступным ценам. Данная сеть магазинов предоставляет возможность велопроката, и на её сайте расположен подробный прейскурант, который предлагает пользователю ознакомиться с ценами за сутки использования велосипеда.

Интерфейс приложения показан на рисунке 1.1.

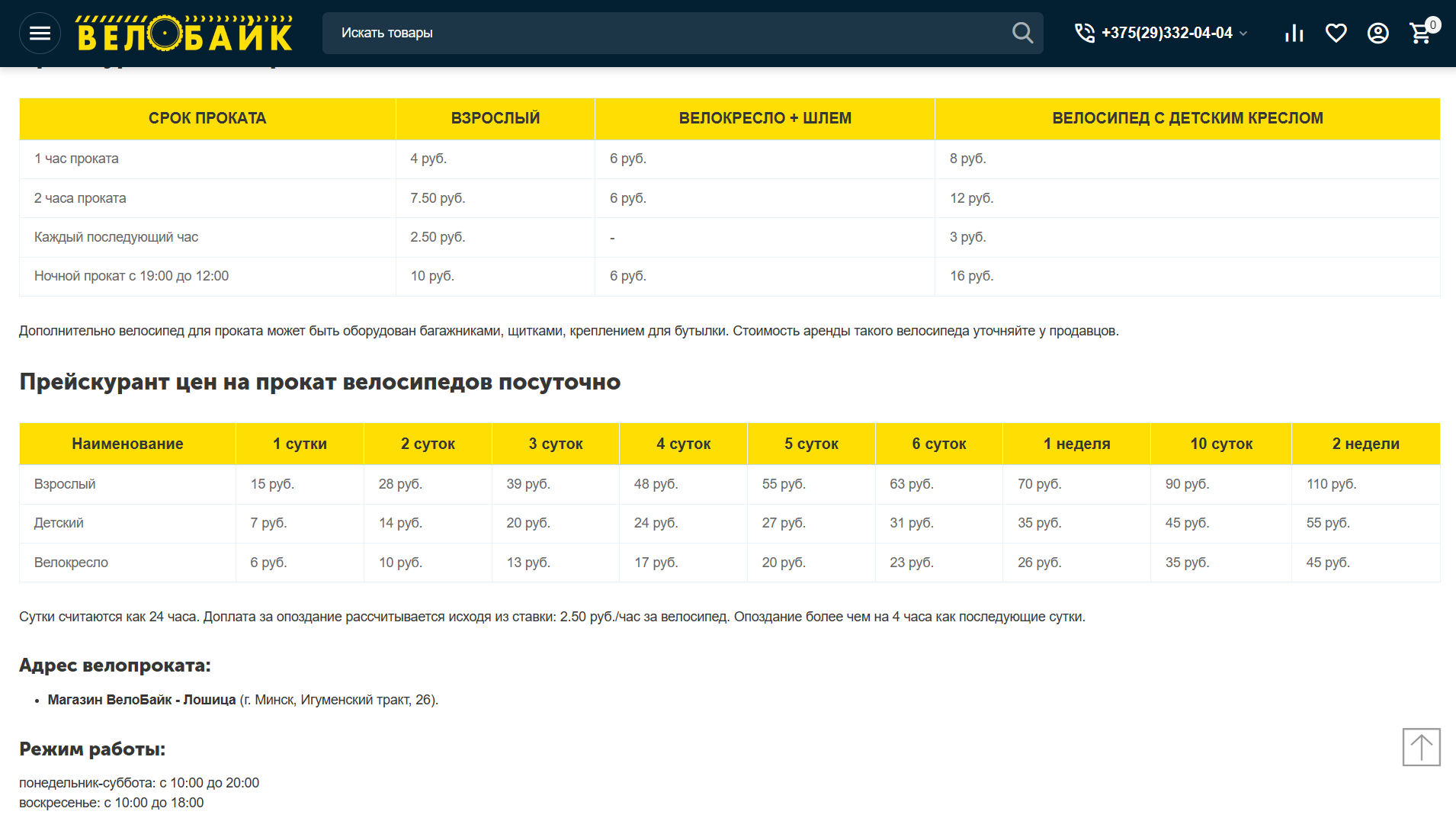


Рисунок 1.1 – Интерфейс приложения Велобайк

Велобайк это универсальный сайт для велосипедистов, поскольку он предоставляет большое количество услуг, связанных с продажей, обслуживанием, ремонтом и прокатом велосипедов.

Недостатком данного сервиса является отсутствие онлайн-заказа. Вы не можете сделать заказ через сайт, а только по телефону или в магазине.

Как и другие веб-сервисы, сайт Велобайк использует HTML, CSS и JavaScript. Сайт построен на основе этих трех основных технологий веб-разработки, которые позволяют создавать динамичные и интерактивные веб-страницы. Сайт адаптирован для разных разрешений экранов и браузеров, а также поддерживает мультиязычность.

Другой популярный сервис – R24.by.

R24.by - это сайт, который предлагает услуги проката и аренды различных товаров в Минске. На сайте можно найти более 16 тысяч товаров от более 800 компаний, которые можно взять в прокат на сутки, часы или даже минуты. Сайт помогает пользователям экономить деньги, время и место, а также решать разные задачи, не покупая ненужные вещи.

На сайте R24.by можно выбрать товары из разных категорий, таких как автомобили, строительное оборудование, бытовая техника, детские товары, спорт и активный отдых, звуковое оборудование и многое другое. Сайт предоставляет подробную информацию о каждом товаре, включая цену, контакты компании, условия аренды, фотографии и описание. Также на сайте можно почитать статьи о преимуществах проката, советы по выбору товаров и отзывы других пользователей. Сайт R24.by - это удобный и полезный ресурс для тех, кто хочет арендовать товары в Минске.

Интерфейс приложения показан на рисунке 1.2.

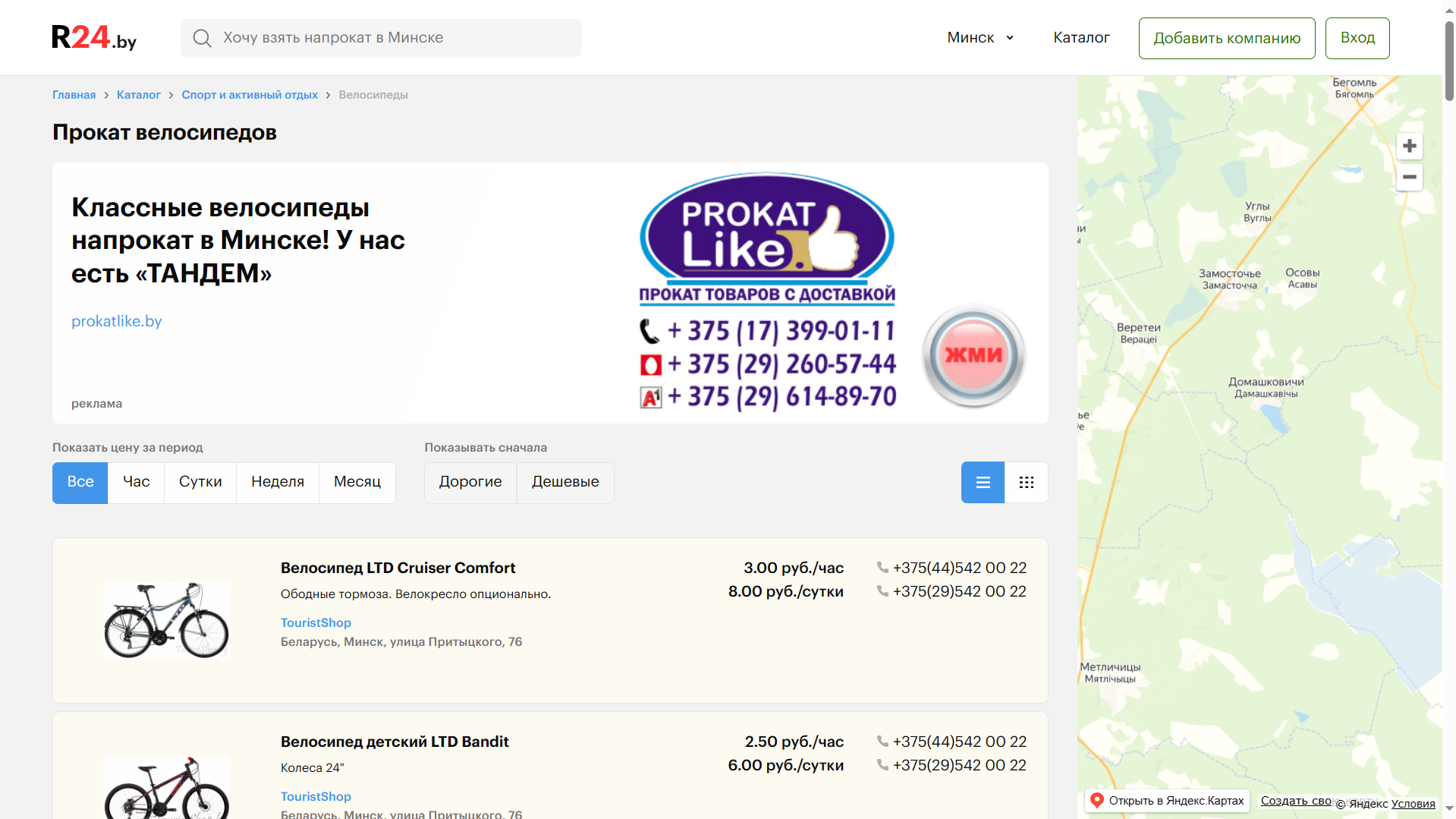


Рисунок 1.2 – Интерфейс приложения R24.by

Сервис имеет веб-сайт, который адаптирован для разных устройств и браузеров. На сайте можно просматривать каталог товаров, фильтровать их по категориям, цене, рейтингу и другим параметрам, а также читать описание, фотографии и условия аренды каждого товара. На сайте также можно оформить заказ, оплатить его онлайн или наличными при получении, а также отслеживать статус заказа и связаться с компанией-поставщиком.

Сервис также имеет мобильное приложение для Android и iOS, которое позволяет делать то же самое, что и на веб-сайте, но с удобством использования смартфона или планшета. Приложение имеет простой и интуитивный интерфейс, быструю загрузку и синхронизацию данных, а также возможность получать уведомления о новых товарах, акциях и скидках.

“Зелёный город” - это сервис аренды велосипедов, который работает во многих городах России, включая Краснодар, Казань, Тверь, Курск, Воронеж, Иваново и другие. Этот сервис предназначен для людей, которые ищут удобный и экологически чистый способ передвижения по городу. Он также может быть полезен для тех, кто хочет арендовать велосипед для развлечения или быстрого перемещения по городу.

.Интерфейс приложения показан на рисунке 1.3.

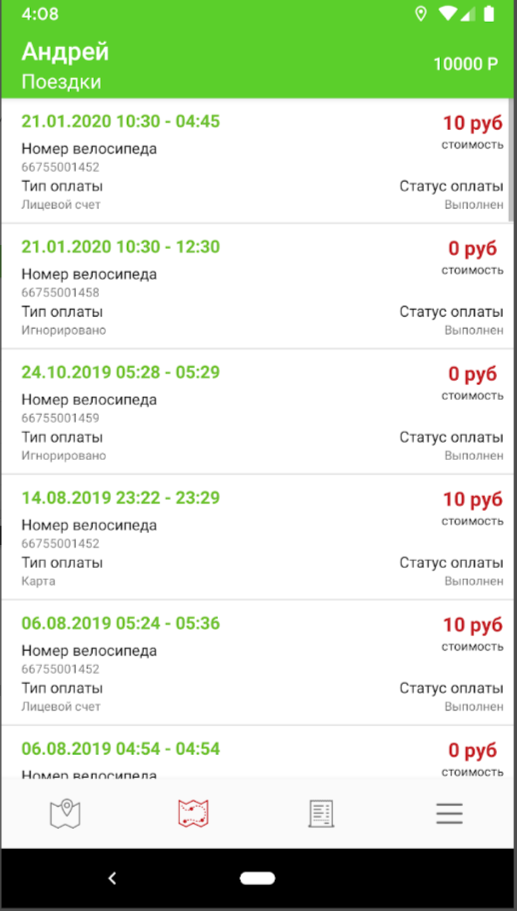


Рисунок 1.3 – Интерфейс Зелёный город

Данное мобильное приложение имеет ряд технических особенностей, таких как: доступность для скачивания бесплатно в Play Market и App Store, простой и удобный доступ к государственным и частным организациям из одного приложения, cостоит из набора типовых модулей, доступных для заполнения всем участникам, в приложении реализованы модули онлайн-записи, онлайн-оплаты.

1.2 Постановка задачи

Проанализировав прототипы, были выделены основная задача и функциональные требования проекта.

Целью курсового проекта стало написание десктопного приложения для аренды велосипедов, а также базы данных и сервера для него. В приложении возможно выполнять авторизацию и регистрацию пользователей, смотреть и оставлять отзывы на велосипеды, производить бронирование велосипедов, изменять ключевую информацию с аккаунта администратора.

Функциональные требования описаны в главе 2.

2 Анализ требований к программному средству и разработка функциональных требований

2.1 Описание средств разработки

Visual Studio - это интегрированная среда разработки (IDE), предоставляемая компанией Microsoft. Она используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-служб и мобильных приложений. Visual Studio поддерживает множество языков программирования, включая, но не ограничиваясь, C++, C#, JavaScript, TypeScript и Python. Это мощное средство разработчика, которое позволяет записывать, редактировать, отлаживать и собирать код, а затем развертывать приложение. Касательно объектно-ориентированного программирования (ООП), Visual Studio предлагает полную поддержку, включая инкапсуляцию, наследование и полиморфизм.

C# - это объектно-ориентированный язык программирования, созданный Microsoft, который работает на платформе .NET. C# произошел от семейства языков C и близок к таким популярным языкам, как C++ и Java. C# используется для разработки мобильных приложений, настольных приложений, веб-приложений, веб-сервисов, веб-сайтов, игр, VR, баз данных и многого другого. C# - это простой, современный, объектно-ориентированный и типобезопасный язык программирования [1].

C# - это объектно-ориентированный язык программирования, разработанный компанией Microsoft. C# поддерживает статическую типизацию, полиморфизм, перегрузку операторов, делегацию, атрибуты, события, обобщенные типы и анонимные функции. C# популярен за счет своей простоты. Программисты, хорошо знакомые с C#, востребованы в индустрии.

Windows Presentation Foundation (WPF) - это библиотека для создания настольных приложений на платформе .NET. WPF использует язык XAML для определения пользовательского интерфейса и поддерживает двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции3.

XAML - это декларативный язык разметки, используемый для определения пользовательского интерфейса в приложениях WPF4. XAML позволяет разрабатывать приложения, используя как разметку, так и код программной части4. Разметка XAML обычно используется для создания окон, страниц и пользовательских элементов управления

Entity Framework (EF) Core - это простая, кроссплатформенная и расширяемая версия популярной технологии доступа к данным Entity Framework с открытым исходным кодом. EF Core может использоваться в качестве объектно-реляционного модуля сопоставления (O/RM), который позволяет разработчикам .NET работать с базой данных с помощью объектов.

Для хранения данных приложения была выбрана база данных Microsoft SQL Server. Microsoft SQL Server - это система управления базами данных (СУБД), разработанная компанией Microsoft. Она предназначена для хранения, обработки и анализа данных, и может функционировать на ПК, ноутбуках, серверах, виртуальных машинах или в облаке. SQL Server поддерживает как локальные, так и сетевые запросы, и может обрабатывать большой объем запросов благодаря своей масштабируемости.

Преимущества Microsoft SQL Server включают: масштабируемость, тесная интеграция с Windows, поддержка многопользовательской среды, работа с удалённым подключением.

2.2 Описание разрабатываемой функциональности программного средства

В программном средстве присутствуют 2 роли: клиент и администратор.

У роли клиента следующие функциональные возможности:

* Выполнение регистрации и авторизации;
* Просмотр доступных велосипедов;
* Бронирование велосипедов;
* Оплата услуг;
* Оставление отзывов;

У роли администратора имеются следующие функциональные возможности:

* Управление информацией о велосипедах;
* Управление бронированием;
* Управление платежами;
* Выполнение модерации отзывов.

В завершение разработки продукта, его следует проверить и отладить, изучить его возможности и, при необходимости, внести изменения или усовершенствования проекта, с учетом возможности дальнейшего расширения проекта.

В графической части записки показана UML-диаграмма функциональности, которая отражает функциональность программного средства с точки зрения достижения целевого результата для пользователя.

3 Проектирование программного средства

Проектирование программного продукта - это критический этап в процессе разработки ПО, который определяет архитектуру, функции, интерфейс и другие аспекты программного продукта. Проектирование ПО основывается на анализе исходных данных задачи, которые были определены в начальной фазе проекта. Анализ исходных данных задачи помогает определить требования к программному продукту, а также ограничения и риски, связанные с его разработкой и внедрением.

3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства

Для построения приложения было принято решение использовать паттерн MVVM (Model-View-ViewModel) - это архитектурный шаблон проектирования, который позволяет отделить логику приложения от визуальной части (представления). Этот паттерн был представлен Джоном Госсманом в 2005 году. MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View):.

* Model описывает данные, используемые в приложении, и может содержать логику, связанную с этими данными.
* View определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.
* ViewModel связывает модель и представление через механизм привязки данных.

Структура проекта, реализующего паттерн MVVM в WPF приложении, обычно включает следующие компоненты: Models, Views, ViewModels, Services, Helpers. Структура проекта «Велопрокат» представлена в таблице 3.1.

Таблица 3.1 – Структура проекта «Babble»

|  |  |
| --- | --- |
| Имя папки | Содержание |
| Helpers | Классы-помошники |
| Interfaces | Интерфейсы, используемые в приложении |
| Repositories | Репозитории данных приложения |
| Services | Сервисы приложения, представляют слой Model и служат для доступа к данным через сервер |
| ViewModels | ViewModel классы приложения |
| Views | XAML представления |
| Commands | Реализация |
| Model | Классы модели приложения |

Таким образом, проект имеют структуру, которая типична для большинства WPF приложений, разработанных с использованием паттерна MVVM, при этом учитывая потребности разрабатываемого продукта.

3.2 Разработка общей схемы приложения

Общая схема окон приложения для пользователя представлена на рисунке 3.1.

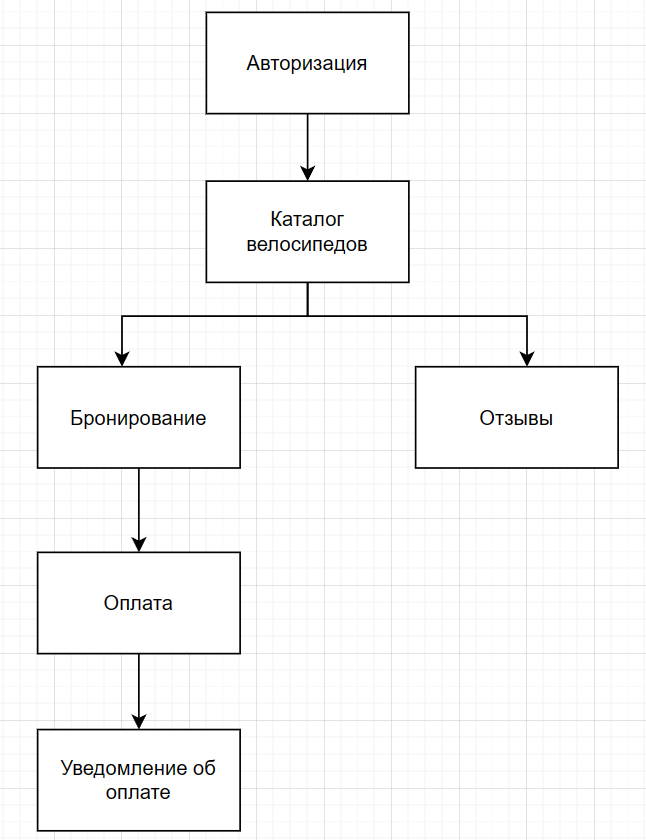


Рисунок 3.1 – Схема окон пользователя

Приложение имеет несколько основных окон. Работа с приложением начинается с окна авторизации, после чего пользователь может перейти в каталог, а оттуда – в бронирование или отзыв.

Общая схема окон приложения для модератора представлена на рисунке 3.2.



Рисунок 3.2 – Схема окон модератора

Когда приложение запускается, пользователь видит окно для входа в систему. После того как авторизация прошла успешно, открывается главное меню приложения, где пользователь может взаимодействовать с различными страницами.

3.3 Проектирование архитектуры приложения

Как и было описано ранее, приложение будет построено с использованием паттерна MVVM. Паттерн MVVM (Model-View-ViewModel) - это шаблон проектирования архитектуры приложения, который позволяет отделить логику приложения от визуальной части (представления). Паттерн состоит из трех компонентов: модели, представления и модели представления.

На рисунке 3.3 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

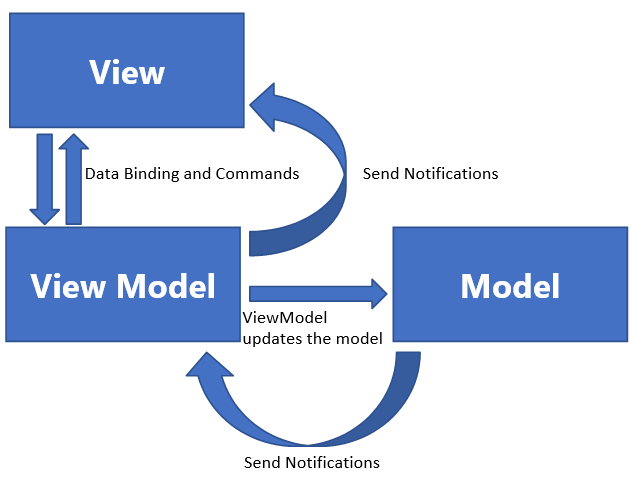


Рисунок 3.3 – Структура паттерна MVVM

Использование паттерна MVVM имеет следующие преимущества:

* Разделение ответственности: MVVM обеспечивает четкое разделение между пользовательским интерфейсом и бизнес-логикой, что упрощает управление кодом.
* Повторное использование кода: ViewModel не имеет прямой связи с View, что позволяет повторно использовать одну и ту же модель представления с различными представлениями.
* Привязка данных: MVVM поддерживает двустороннюю привязку данных между View и ViewModel, что автоматически обновляет View при изменении данных в ViewModel.
* Тестируемость: Благодаря разделению ответственности, бизнес-логику можно легко тестировать независимо от пользовательского интерфейса.

Привязка данных – важная часть паттерна MVVM. В WPF для привязки данных используется интерфейс INotifyPropertyChanged. INotifyPropertyChanged - это интерфейс, предоставляемый .NET, который класс может реализовать для сигнализации об изменении значения свойства. Это полезно в сценариях привязки данных, где автоматическое обновление пользовательского интерфейса может быть инициировано после изменения данных, к которым он привязан.

Частым решением для имплементации INotifyPropertyChanged является базовая ViewModel, от которой будут наследоваться все модели представления приложения, что упрощает работу, так как интерфейс нужно реализовать лишь один раз. В разрабатываемом приложении будет использоваться такой же подход.

Таким образом, был выбран подход для создания приложения с разделением ответственности между слоями программы.

3.4 Проектирование базы данных

Процесс проектирования базы данных (БД) — это процесс определения структуры, схемы и ограничений данных, которые будут храниться и обрабатываться в приложении. Процесс проектирования БД зависит от типа и сложности приложения, а также от требований к производительности, безопасности и масштабируемости.

Один из способов проектирования БД — это использование Entity Framework (EF), который является современным фреймворком для работы с различными базами данных на платформе .NET. EF позволяет создавать модели данных, которые представляют сущности и связи в БД, а также выполнять операции CRUD (создание, чтение, обновление и удаление) над данными с помощью LINQ (Language Integrated Query).

Созданная база данных содержит в себе 5 таблиц: UserAccount, Bike, Review, Booking, Payment.

В таблице 3.3 представлена структура таблицы Users.

Таблица 3.3 – Структура таблицы UserAccount

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Nullable |
| userId | INT | Нет |
| user\_login | NVARCHAR(50) | Нет |
| salt | NVARCHAR(128) | Нет |
| first\_name | NVARCHAR(50) | Нет |
| last\_name | NVARCHAR(50) | Нет |
| phone\_number | VARCHAR(15) | Нет |
| email | VARCHAR(100) | Нет |
| role | NVARCHAR(50) | Нет |
| image | NVARCHAR(255) | Нет |
| password | NVARCHAR(128) | Нет |

В данной таблице содержится информация о пользователях платформы.

В таблице 3.4 представлена структура таблицы Bike.

Таблица 3.4 – Структура таблицы Bike

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Nullable |
| id | INT | Нет |
| model | NVARCHAR(50) | Нет |
| price\_per\_day | DECIMAL(10, 2) | Нет |
| description | NVARCHAR(255) | Да |
| image | NVARCHAR(255) | Да |

В данной таблице содержится информация о велосипедах.

В таблице 3.5 представлена структура таблицы Messages.

Таблица 3.5 – Структура таблицы Review

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Nullable |
| Id | INT | Нет |
| client\_id | INT | Нет |
| bike\_id | INT | Нет |
| review\_header | NVARCHAR(100) | Нет |
| review\_text | NVARCHAR (255) | Нет |
| score | INT | Нет |
| answer\_text | NVARCHAR(255) | Да |

В данной таблице содержится информация об отзывах.

В таблице 3.6 представлена структура таблицы Booking.

Таблица 3.6 – Структура таблицы Booking

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Nullable |
| Id | INT | Нет |
| client\_id | INT | Нет |
| bike\_id | INT | Нет |
| booking\_date | DATE | Нет |
| status | NVARCHAR(50) | Нет |
| rental\_start\_date | DATE | Нет |
| rental\_end\_date | DATE | Нет |

В данной таблице содержится информация о бронированиях.

В таблице 3.7 представлена структура таблицы Payment.

Таблица 3.7 – Структура таблицы Payment

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Nullable |
| id | INT | Нет |
| booking\_id | INT | Нет |
| amount | DECIMAL(10, 2) | Нет |
| payment\_date | DATE | Нет |

В данной содержится информация об оплатах бронирований.

4 Создание (реализация) программного средства

Следующим этапом является реализация спроектированного ПО. Создание продукта включает разработку всех компонентов программы, следую выбранным принципам построения ПО, архитектуре и паттернам.

4.1 Регистрация

Для регистрации пользователем нового аккаунта был реализован класс SignUpView, в котором представлены все необходимые поля для создания аккаунта пользователя. Также данное окно содержит две кнопки: кнопку подтверждения создания аккаунта и кнопку перехода на форму входа в учетную запись. Рядом с кнопкой входа в учётную запись располагается подпись, поясняющая, что эта кнопка предназначена для тех, у кого уже есть аккаунт данного приложения.

Поля ввода окна регистрации привязаны к свойствам экземпляра UserAccount, хранимого в SignUpAndInViewModel. При нажатии на кнопку Sign Up, выполняется SignUpCommand. В ней происходит проверка на пустые строки в полях ввода и на длину пароля. Если проверка пройдена, то пароль, введённый пользователем при регистрации, хэшируется, его хэш и salt помещаются в экземпляр нового пользователя.

Далее вызывается метод сервиса модели аккаунта AddItem(), который с помощью класса UserAccountRepository добавляет новую строку в таблицу UserAccount базы данных.

4.2 Авторизация

SignInView содержит два поля для ввода: логин и пароль. Логин является уникальным столбцом в базе данных, поэтому мы можем однозначно идентифицировать пользователя по введённому им логину. Для ввода пароля, как и в SignUpView был использован элемент PasswordBox, позволяющий скрыть символы, вводимые в поле. Также в окне присутствует кнопка, нажатие на которую перенаправляет обратно на окно регистрации.

При нажатии на кнопку SignIn выполняется команда SignInCommand. Внутри этой команды из базы данных с помощью класса-сервиса извлекается аккаунт с логином, введённым в поле Login. Далее хэш введённого пароля сравнивается с хэшем пароля учётной записи из базы данных. Если они совпадают, то заполняется глобальный объект сессии и открывается окно каталога.

4.3 Функции администратора: Добавление велосипеда

Администратор может добавлять велосипед в базу данных и приложение, определяя значения всех его свойств. Находясь в главном каталоге, оперируя такими объектами, как велосипед, администратору доступны 3 действия: создать, удалить, изменить велосипед.

При нажатии на кнопку Добавить выполняется код команды AddCommand. Эта команда создаёт новое окно для ввода данных о новом велосипеде NewBikeView. NewBikeView имеет поля ввода, связанные с объектом NewItem. Есть возможность выбрать изображение с помощью OpenFileDialog, а после загрузить его на сервер. При нажатии на кнопку Save, выполняется команда SaveCommand, которая добавляет в базу данных новый велосипед.

**4.4 Функции администратора: Редактирование и удаление объектов**

Администратор способен редактировать такие объекты, как велосипеды, бронирования, платежи отзывы. Для этого CatalogViewModel и NewItemViewModel были сделаны обобщёнными, чтобы работать с объектами любого типа.

Для редактирования объектов у администратора в CatalogView есть кнопка редактировать, которая связана с командой EditCommand и может быть нажата, только если администратор выбрал один из объеков в ListBox. Эта команда может работать с любым типом объекта и открывает соответствующую типу объекта New(TypeOfObject)View, которые представляют пользовательский интерфейс для редактирования объекта определённого типа.

Чтобы удалить объект, администратору достаточно выбрать объект из ListBox и нажать кнопку удалить на нижней панели, которая связана с командой RemoveCommand. Эта команда вызывает сервис, репозиторий которого удаляет из базы данных нужный объект. Удалить можно следующие объекты: Отзывы, платежи. Бронирование, вместо удаления, можно отменить. В результате отмены в столбце state бронирования будет значение «Отменено».

**4.5 Функции клиента: Просмотр и добавление отзывов**

У клиента в нижней панели расположены кнопки «Забронировать» и «Отзывы». При выборе определённого велосипеда и нажатии на кнопку «Отзывы» выполняется команда ReviewCommand, которая открывает BkeReviewView. В этом окне можно увидеть все отзывы на данный велосипед, а также заполнить форму собственного отзыва и отправить её. Делается это с помощью команды SubmitReviewCommand, которая вызывает сервис и отправляет в базу данных новый отзыв.

**4.6 Функции клиента: Бронирование велосипеда**

При нажатии на кнопку «Забронировать», команда BookCommand открывает новое окно BookingWindow, в котором пользователю предлагается выбрать наальную и конечную даты бронирования. При нажатии на кнопку Book, будет открыто ещё одно окно PaymentView, в котором пользователю будет предложено ввести платежные данные для оплаты. Нажатие на кнопку Pay создаёт новый объект Payment и Booking и добавляет соответствующие строки в базу данных. После этого клиенту приходит уведомление о том, что его бронирование успешно оформлено.

**5. Тестирование и анализ работоспособности и полученных результатов**

**5.1 Тестирование авторизации и регистрации**

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе будут рассмотрены некоторые такие сценарии и посмотрим на их обработку.

В момент авторизации, возможна такая ситуация, в которой пользователь ничего не ввёл. Обработка данного сценария приведена на рисунке 5.1.

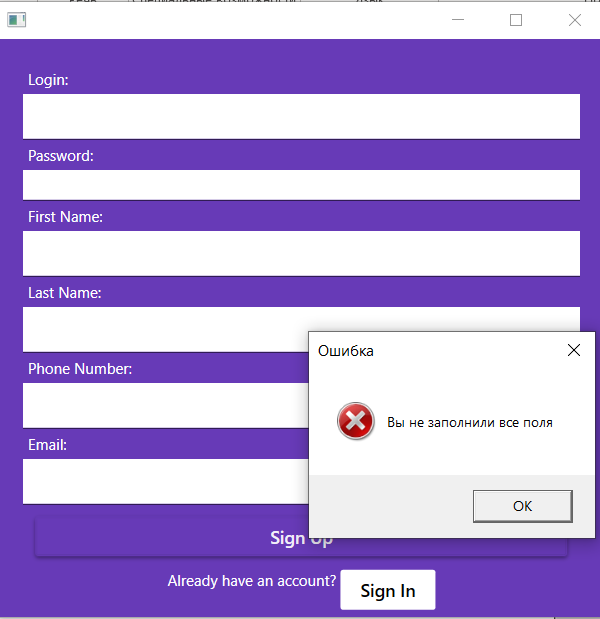


Рисунок 5.1 — Обработка пустых полей при авторизации

После нажатия на кнопку регистрации или входа, если не все поля заполнены, то возникает ошибка, в тексте которой написано, что пользователь не ввёл все поля.

При вводе некорректных данных, возникает ошибка, представленная на рисунке 5.2.

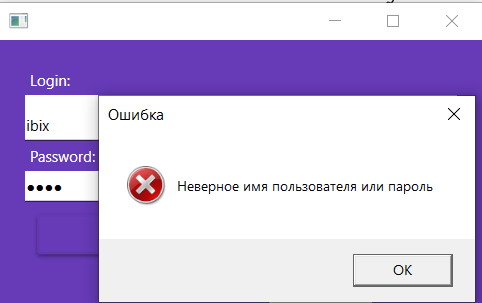


Рисунок 5.2 — Обработка некорректного ввода полей при авторизации

Как видно с рисунка вывело сообщение «Неверное имя пользователя или пароль».

Если в базе данных нет пользователя с запрашиваемым логином, будет выведена ошибка

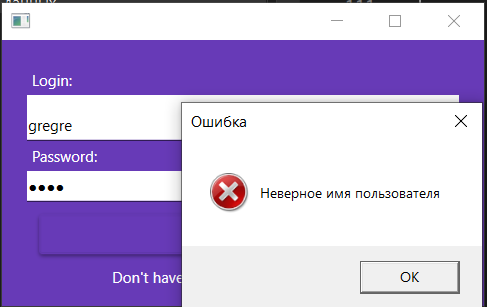


Рисунок 5.3 — Обработка ввода несуществующего пользователя

**5.2 Тестирование основного функционала**

При попытке удалить велосипед, будет возникать ошибка, в связи с тем, что 5.5.

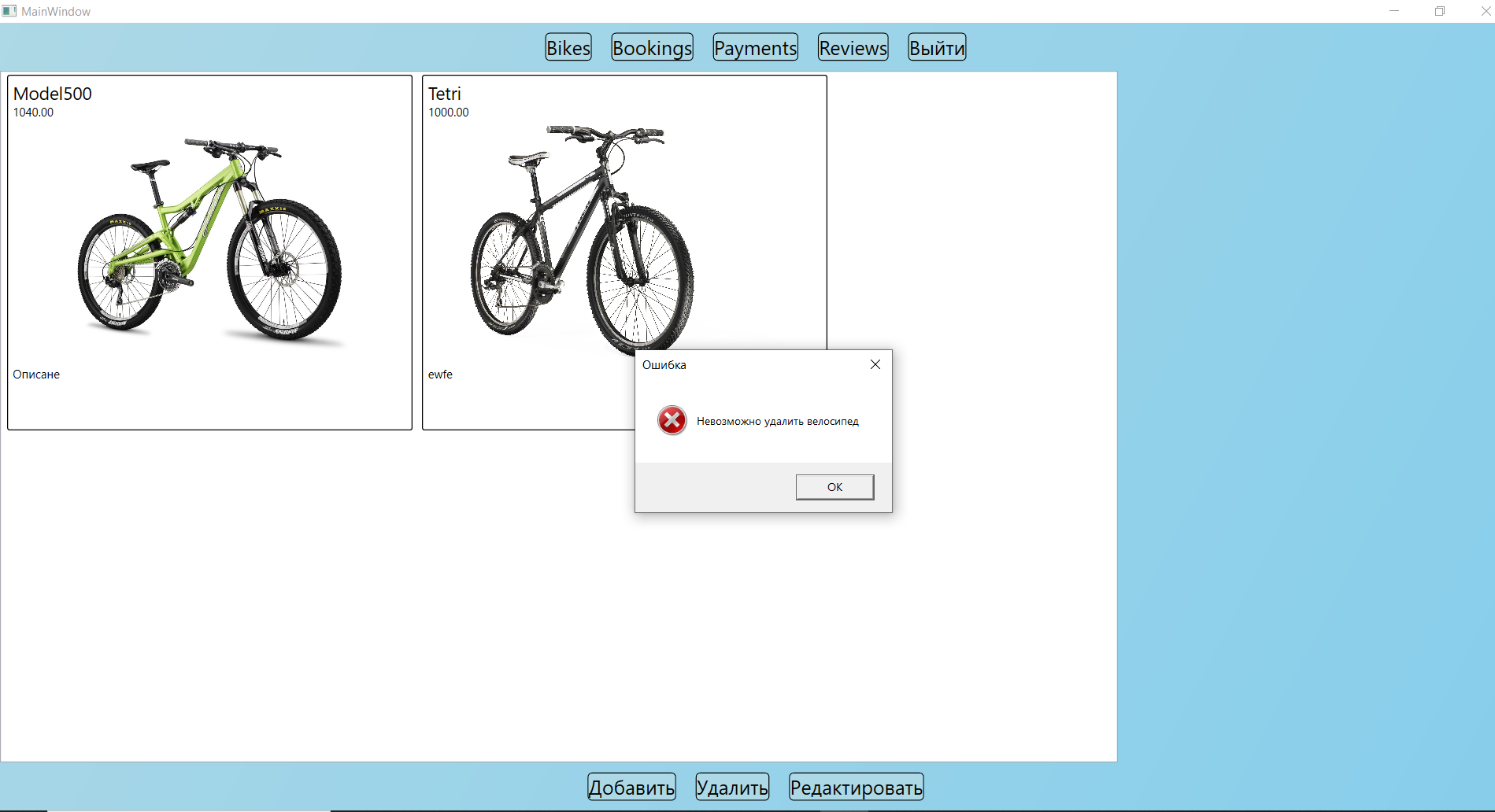


Рисунок 5.4 — Обработка нажатий на кнопки

Это делается, потому что на сущность велосипеда ссылаются такие сущности, как Booking, Review.

Поскольку в список функций, выполняемых администратором, не входит написание отзывов, данная возможность для администратора была отключена. Вместо этого он получит сообщение об ошибке «Вы не можете добавить этот объект».

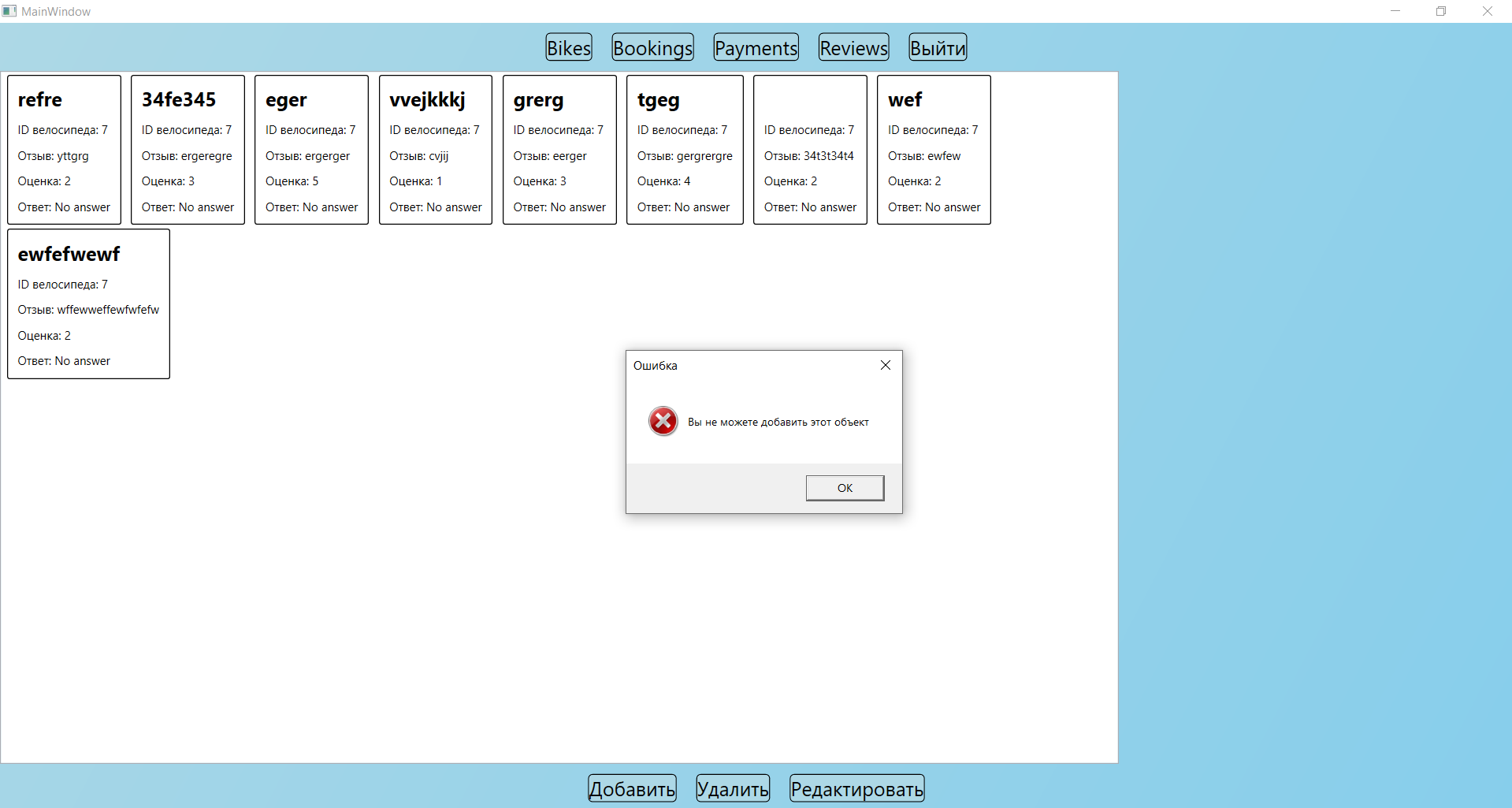


Рисунок 5.5 — невозможность добавления объекта

При оставлении какого-либо поля пустым в форме добавления велосипеда и нажатии «Save», возникнет сообщение об ошибке «Невозможно добавить», .

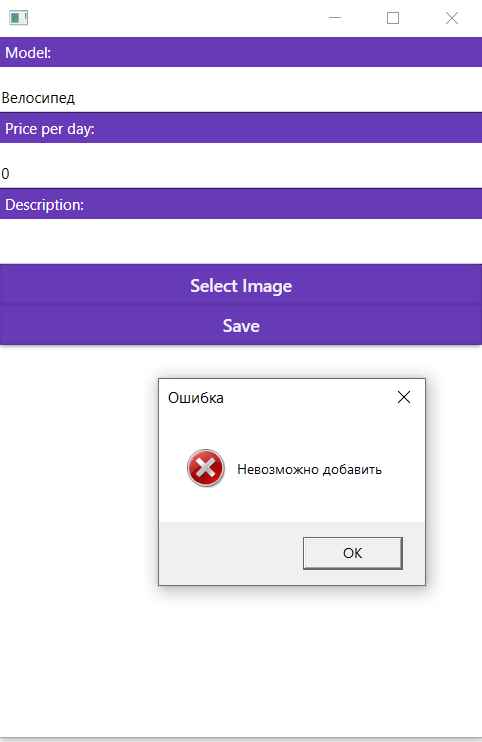


Рисунок 5.5 — невозможность добавления объекта

**6. Руководство по использованию**

То, как вам следует пользоваться данным приложением, зависит от того, какими правами доступа к системе вы обладаете. Если у вас есть аккаунт администратора, то вы сможете использовать операции для манипулирования данными приложения. Если вы клиент, то вы будете иметь возможность просматривать только велосипеды, а таже бронировать и оплачивать их и писать отзывы.

При запуске приложения нас приветствует экран регистрации с несколькими полями для ввода информации нового пользователя. Если пользователь уже имеет аккаунт в системе, он может нажать кнопку «Войти», которая закроет окно регистрации и откроет окно входа. Кнопка «Sign Up»,

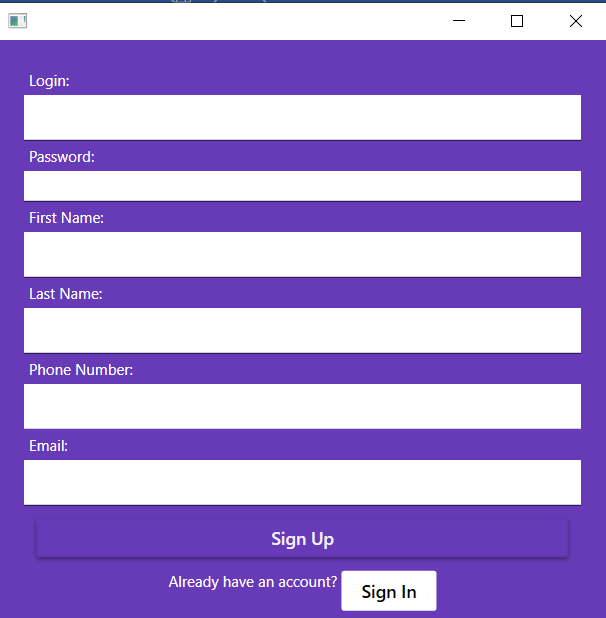


Рисунок 6.1 — Регистрация

После успешного входа, пользователя перенаправит в каталог велосипедов, который изображено на рисунке 6.2.

В этом окне можно выбрать определённый велосипед и выполнить с ним контекстные действия

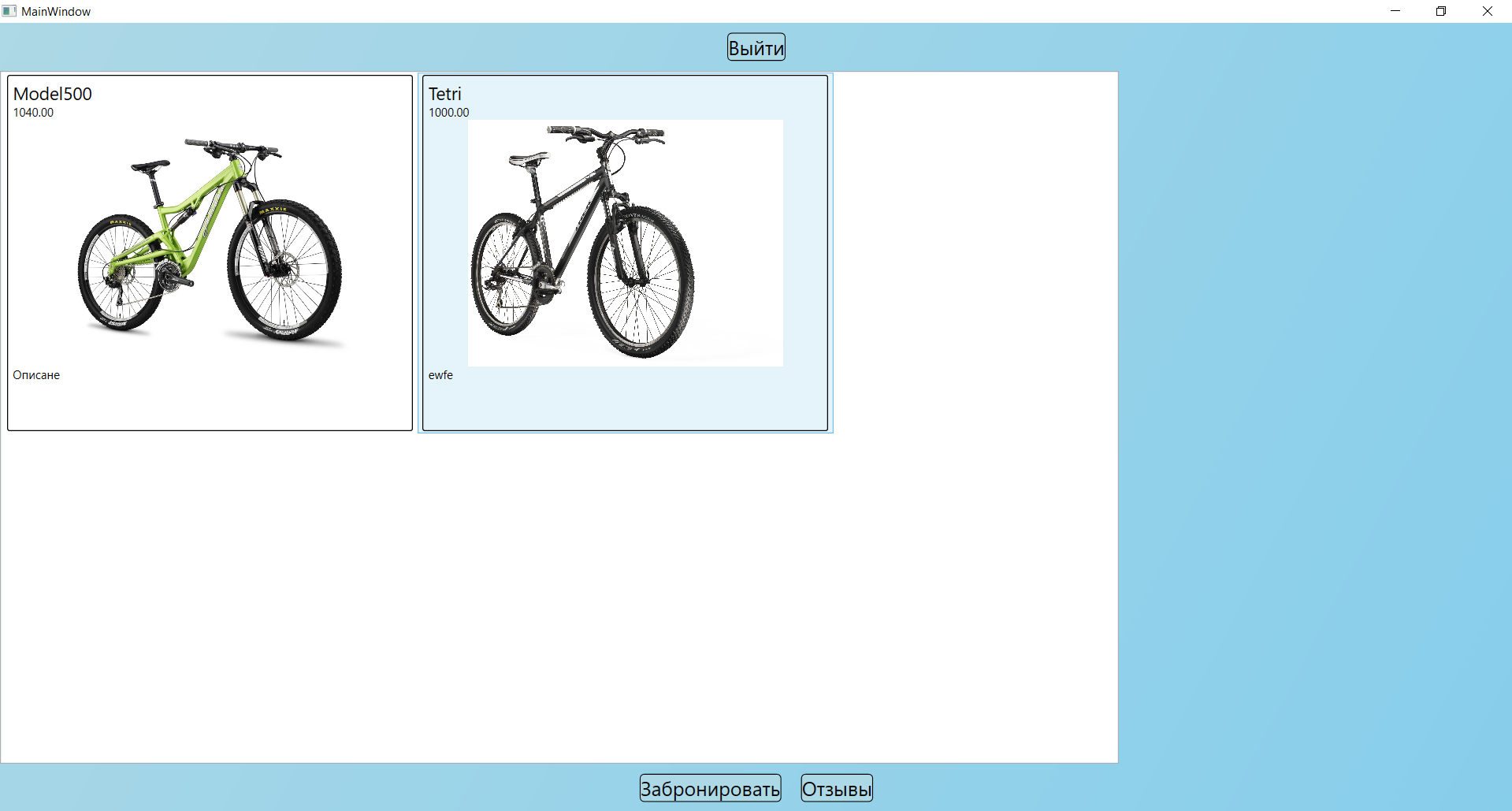


Рисунок 6.2 — Главный каталог

Клиент рассматривает велосипеды, читая информацию о них. Если его заинтересует какой-либо из вариантов, он нажимает на кнопку Отзывы.

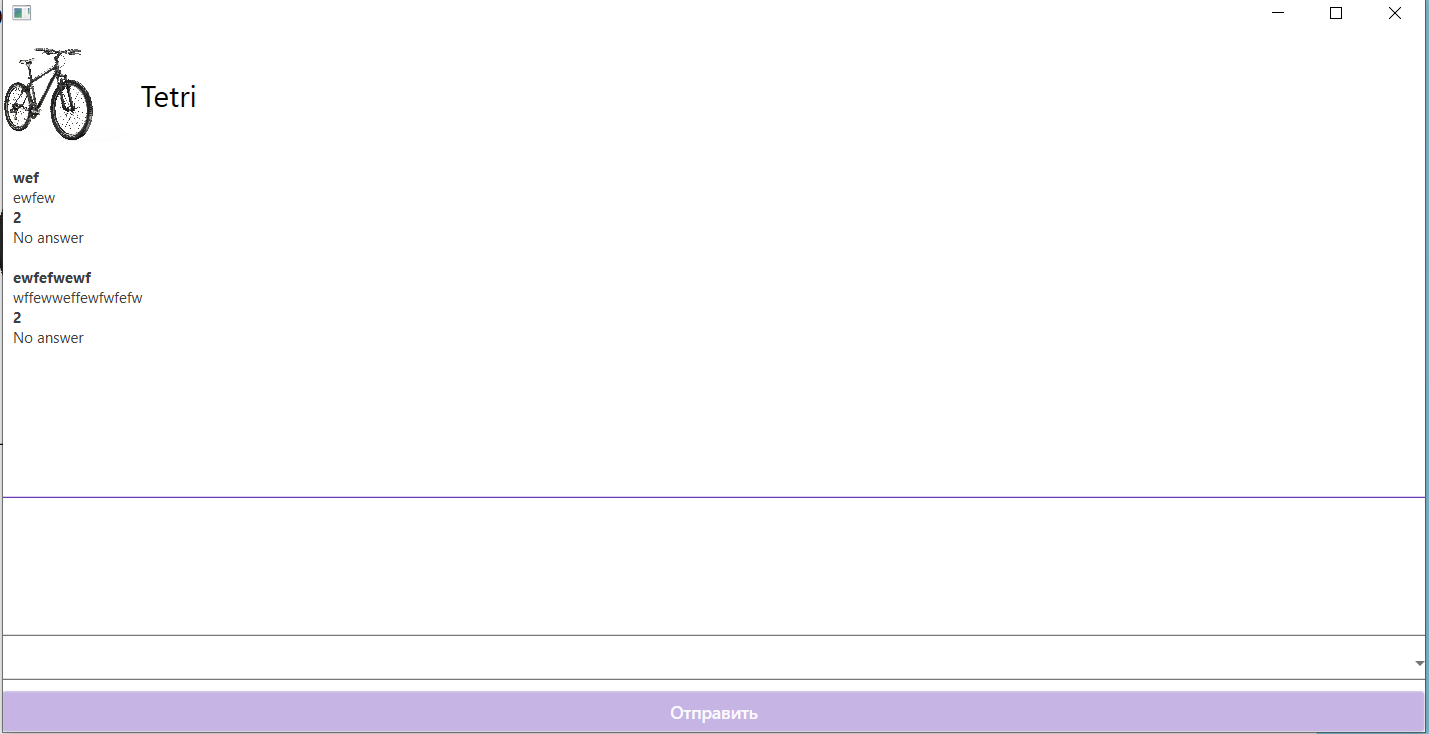


Рисунок 6.3 — Отзывы о велосипеде

Когда пользователь прочитает отзывы о велосипеде, и они его устроят, то он нажмет кнопку бронировать. Окно бронирования изображено на рисунке 6.4. Здесь клиент выбирает дату начала и окончания бронирования.

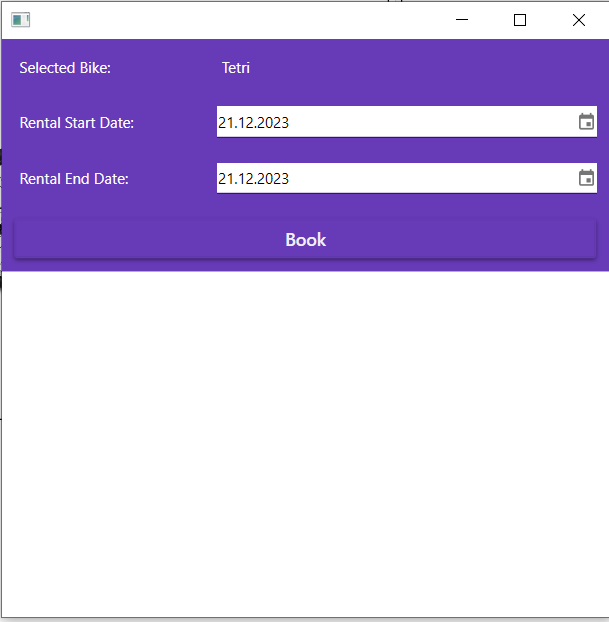


Рисунок 6.4 — Окно бронирования

После выбора дат начала и конца бронирования, клиент нажимает кнопку Book и открывается окно платежа, представленное на рисунке 6.5. Здесь клиент вводит свои платёжные данные, и, нажав кнопку Pay, подтверждает бронирование. Если бронирование и оплата проходят успешно, клиент получит уведомление об оплате.

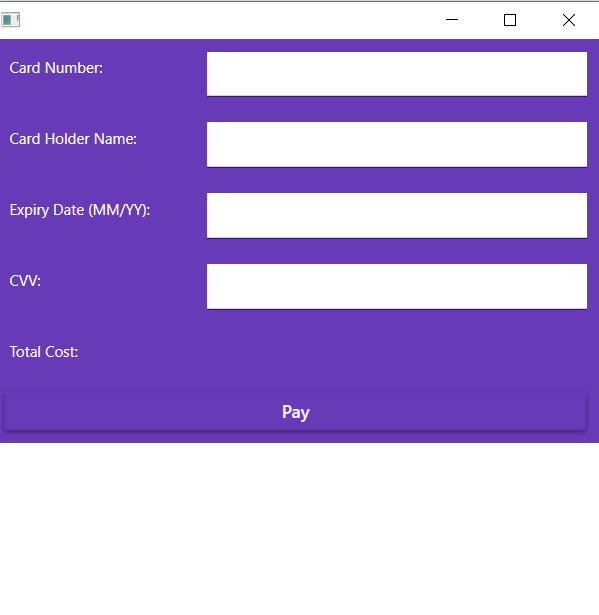


Рисунок 6.5 — окно оплаты бронирования

После добавления гитары в корзину пользователь может оформить заказ а если

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано программное средство «Велопрокат», которое предоставляет функционал современного мессенджера.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* регистрация пользователей;
* авторизация пользователей;
* просмотр каталога велосипедов;
* просмотр отзывов;
* добавление отзывов;
* бронирование велосипедов;
* оплата бронирования;
* хранение информации в базе данных.
* хранение изображений на сервере

Разработанное программное средство обрабатывает ошибки и выводит соответствующее собщению пользователю.

При проектировании и реализации были использованы паттерны MVVM, классы-сервисы, классы-репозитории.

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

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Руководство по WPF [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/wpf/

2 Паттерны проектирования [Электронный ресурс] / Режим доступа: <https://refactoring.guru/ru/design-patterns>

3 Документация по C# [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/

Приложение А

using Microsoft.EntityFrameworkCore;

using System;

using System.IO;

namespace BikeRental

{

public class BikeRentalContext : DbContext

{

public DbSet<Bike> Bike { get; set; }

public DbSet<Review> Review { get; set; }

public DbSet<Payment> Payment { get; set; }

public DbSet<Booking> Booking { get; set; }

public DbSet<UserAccount> UserAccount { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<UserAccount>().HasKey(u => u.UserId);

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

try

{

optionsBuilder.UseSqlServer(@"Server=localhost,1433;Database=BikeRental;User=sa;Password=1234;");

}

catch (Exception ex)

{

string logPath = @"D:\Work\Kusrach4\Application\BikeRental\log.txt";

File.WriteAllText(logPath, $"Произошла ошибка при подключении к базе данных: {ex.Message}");

}

}

}

}

Листинг – Класс BikeRentalContext

Приложение Б

public partial class ItemCatalog : Window

{

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

{

// Создать новое окно с ReviewService в качестве DataContext

var context = new BikeRentalContext();

var repository = new ReviewRepository(context);

var service = new ReviewService(repository);

var newWindow = new ItemCatalog

{

DataContext = new CatalogViewModel<Review, ReviewService>(service)

};

// Открыть новое окно

newWindow.Show();

// Закрыть текущее окно

this.Close();

}

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

{

// Создать новое окно с ReviewService в качестве DataContext

var context = new BikeRentalContext();

var repository = new BikeRepository(context);

var service = new BikeService(repository);

var newWindow = new ItemCatalog

{

DataContext = new CatalogViewModel<Bike, BikeService>(service)

};

// Открыть новое окно

newWindow.Show();

// Закрыть текущее окно

this.Close();

}

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

{

// Создать новое окно с ReviewService в качестве DataContext

var context = new BikeRentalContext();

var repository = new PaymentRepository(context);

var service = new PaymentService(repository);

var newWindow = new ItemCatalog

{

DataContext = new CatalogViewModel<Payment, PaymentService>(service)

};

// Открыть новое окно

newWindow.Show();

// Закрыть текущее окно

this.Close();

}

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

{

// Создать новое окно с ReviewService в качестве DataContext

var context = new BikeRentalContext();

var repository = new BookingRepository(context);

var service = new BookingService(repository);

var newWindow = new ItemCatalog

{

DataContext = new CatalogViewModel<Booking, BookingService>(service)

};

// Открыть новое окно

newWindow.Show();

// Закрыть текущее окно

this.Close();

}

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

{

SessionState.IsLoggedIn = false;

SignUpView signUpView = new SignUpView();

signUpView.Show();

this.Close();

}

public ItemCatalog()

{

InitializeComponent();

try

{

var context = new BikeRentalContext();

var repository = new BikeRepository(context);

var service = new BikeService(repository);

DataContext = new CatalogViewModel<Bike, BikeService>(service);

}

catch (Exception ex)

{

string logPath = @"D:\Work\Kusrach4\Application\BikeRental\log.txt";

File.WriteAllText(logPath, $"Произошла ошибка: {ex.Message}");

DataContext = new CatalogViewModel<Bike, BikeService>();

}

}

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

{

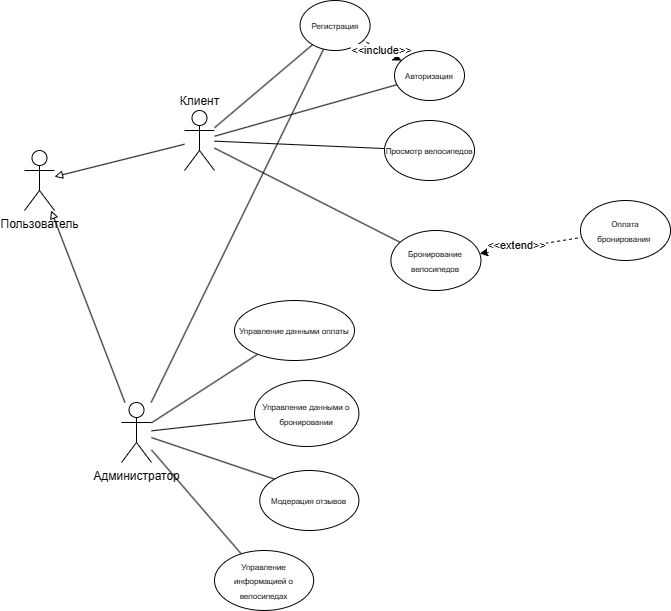
}

}

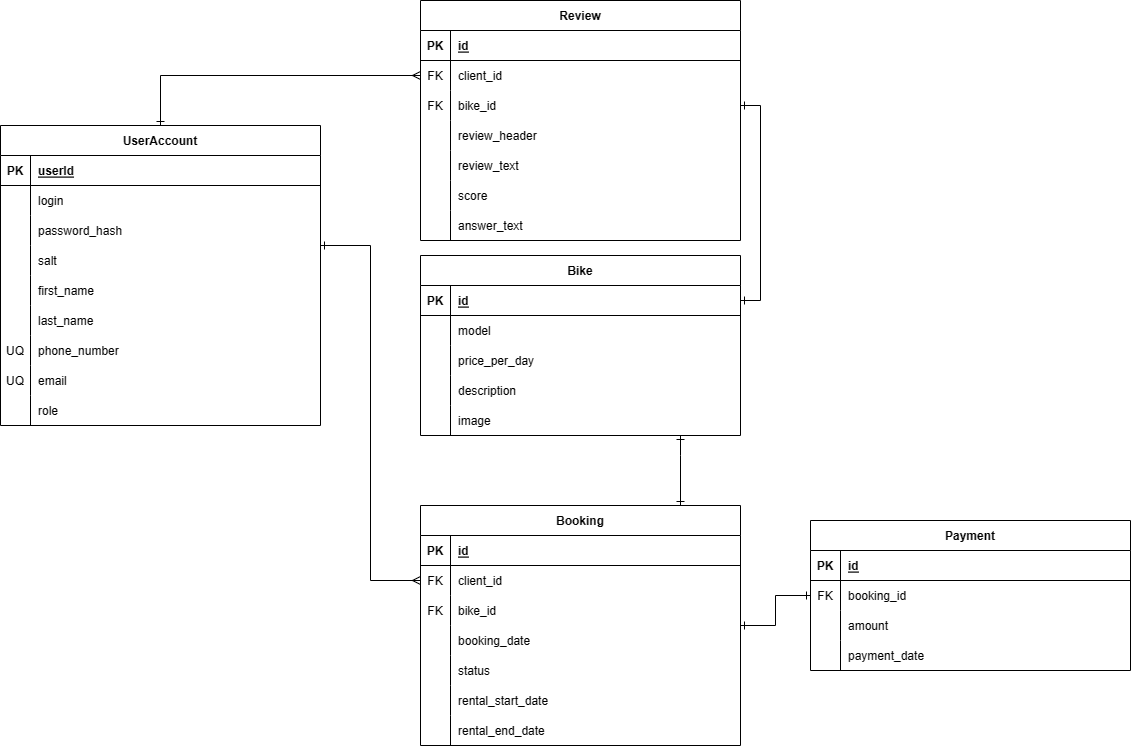
Листинг – Класс CatalogViewModel

Графический материал

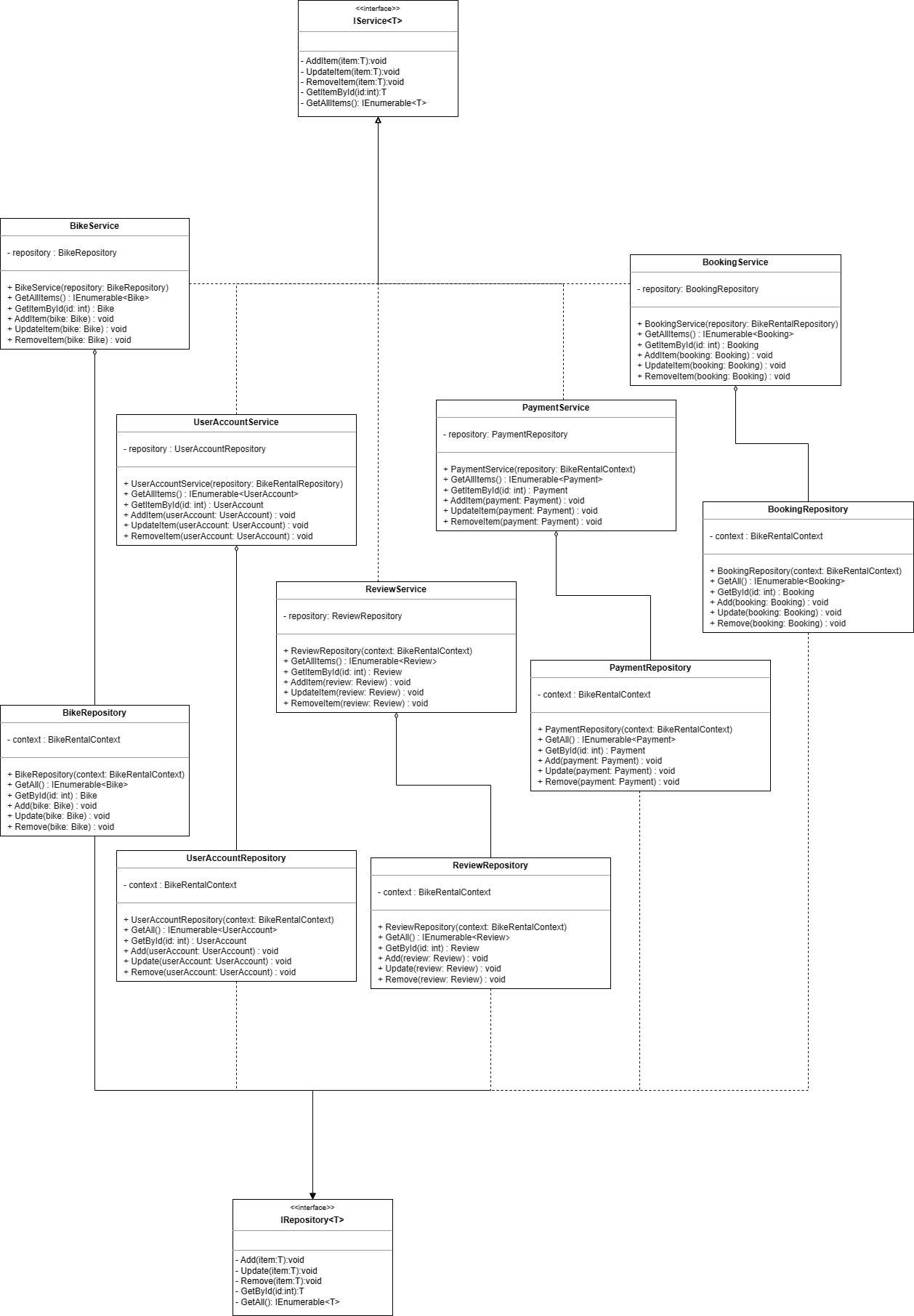
Диаграмма вариантов использования



Логическая схема базы данных



Диаграммы классов



Диаграммы последовательностей

