МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования»

Тема: «Программное средство «Интернет-магазин кроссовок»

Исполнитель

студент 3 курса группы 2 Ободов Павел Андреевич

(Ф.И.О.)

Руководитель работы преп.-стажер Якунович А.В.

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

Курсовой проект защищен с оценкой

(подпись)

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc154110905)

[1 Аналитический обзор литературы и формирование требований 7](#_Toc154110906)

[1.1 Анализ прототипов 7](#_Toc154110907)

[1.2 Требования к проекту 11](#_Toc154110908)

[1.3 Вывод 12](#_Toc154110909)

[2 Анализ и проектирование архитектуры приложения 13](#_Toc154110910)

[2.+1 Описание средств разработки 13](#_Toc154110911)

[2.1.1 Microsoft Visual Studio 2022 13](#_Toc154110912)

[2.1.2 Программная платформа .NET 7 13](#_Toc154110915)

[2.1.3 Язык программирования C# 13](#_Toc154110916)

[2.1.4 Расширяемый язык разметки XAML 14](#_Toc154110917)

[2.1.5 Технология WPF 14](#_Toc154110918)

[2.1.6 Технология Entity Framework 15](#_Toc154110919)

[2.1.7 SQL Server 15](#_Toc154110920)

[2.1.8 Паттерн MVVM 16](#_Toc154110921)

[2.2 Спецификация функциональных требований к программному средству 16](#_Toc154110922)

[2.3 Спецификация функциональных требований 17](#_Toc154110923)

[2.4 Вывод 18](#_Toc154110924)

[3 Проектирование программного средства 19](#_Toc154110925)

[3.1 Общая структура 19](#_Toc154110926)

[3.1.1 Структура клиентской части 19](#_Toc154110927)

[3.2 Взаимоотношение между классами 21](#_Toc154110928)

[3.3 Модель базы данных 21](#_Toc154110929)

[3.4 Вывод 24](#_Toc154110930)

[4 Разработка функциональной модели и модели данных 25](#_Toc154110931)

[4.1 Основные модели программного средства 25](#_Toc154110932)

[4.2 Регистрация 25](#_Toc154110933)

[4.3 Авторизация 28](#_Toc154110934)

[4.4 Просмотр каталога кроссовок 32](#_Toc154110935)

[4.5 Добавление модели в каталог 33](#_Toc154110936)

[4.6 Поиск модели 34](#_Toc154110937)

[4.7 Изменение цветового оформления интерфейса 35](#_Toc154110938)

[4.8 Просмотр истории заказанных моделей кроссовок 36](#_Toc154110939)

[4.9 Вывод 37](#_Toc154110940)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 38](#_Toc154110941)

[5.1 Тестирование авторизации и регистрации 38](#_Toc154110942)

[5.2 Тестирование добавления кроссовок 40](#_Toc154110943)

[5.3 Тестирование панели администратора 42](#_Toc154110944)

[5.4 Вывод 44](#_Toc154110945)

[6 Руководство по установке и использованию 44](#_Toc154110946)

[Заключение 48](#_Toc154110947)

[Список использованных источников 49](#_Toc154110948)

[Приложение А 50](#_Toc154110949)

# **ВВЕДЕНИЕ**

Одежда и обувь являются одними из самых востребованных товаров, уступая только продуктам питания. Особенность рынка одежды и обуви заключается в том, что он постоянно претерпевает изменения. Это связано с развитием модной индустрии, которая вводит новые тенденции, тем самым формируя высокий спрос на товары из данной категории. Ежегодно число магазинов одежды и обуви увеличивается, продажи растут, что делает данный сегмент бизнеса достаточно привлекательным. Даже в период кризиса люди не всегда готовы экономить на этом виде товаров. Только 30% опрошенных признали, что снизили расходы на одежду и обувь. В среднем, на одежду и обувь люди тратят большое количество денег. Среднестатистический потребитель покупает одну пару каждые 1-1,5 месяца и имеет минимум 4 вида обуви.

Особенности климатических условий вынуждают менять обувь в зависимости от сезона – т.е. четыре (минимум два) раза в год. Таким образом, идея по продаже обуви никогда не потеряет своей актуальности. Таким образом, основными тенденциями обувного ритейла в мире являются: смещение спроса в более дешевый сегмент, что обусловлено ростом стоимости на обувь и сокращением доходов россиян; переориентация обувных магазинов на более универсальные модели обуви; консолидация рынка, укрепление позиций крупных обувных сетей и уход с рынка мелких игроков; укрепление на рынке продукции российского производства, которое укрепляется в условиях ценового преимущества, полученного вследствие девальвации рубля и смещения спроса в сторону недорогой, универсальной обуви; активный процесс импортозамещения. Происходит сокращение импорта обуви: если в докризисный период доля импорта на отечественном обувном рынке составляла 80%, то сегодня она составляет менее 70%. Ожидается, что в ближайшие два года соотношение российской  и импортной обуви на рынке составит 35% и 65% соответственно; сегодня портрет клиента обувного магазина содержит следующие характеристики: рационализм, сокращение частоты покупок, приверженность сложившемуся представлению о качестве потребляемых покупок. В период сокращения доли игроков на рынке есть возможность закрепиться в отрасли, которая постепенно восстанавливается.

Таким образом, открытие обувного магазина может стать перспективным проектом, особенно если будут учтены основные тренды обувного ритейла.

Данная курсовая работа посвящена разработке интернет-магазин кроссовок "SneackersShop". Наш сервис является актуальным, т.к вы можете приобрести любую понравившуюся модель быстро, качественно и, что самое главное на свой личный адрес. SneackersShop представляет собой платформу, предназначенную для просмотра, поиска и заказа кроссовок различных популярных брендов. Основные характеристики SneackersShop включают в себя:

Просмотр каталога: Пользователи могут просматривать нужные модели кроссовок, а именно весь ассортимент товаров.

Поиск и Сортировка: Каждый пользователь может искать нужную модель по названию, цене, цвету, размеру и типу.

Оформление заказа: После выбора нужной модели каждый пользователь может оформить заказа указав при этом адрес доставки.

Просмотр истории заказов: После оформления заказа пользователь может просмотреть истории заказанных моделей.

Редактирование личной информации: У каждого клиента есть возможность изменить личную информацию учётной записи.

Технические детали проекта включают в себя:

Серверная и Клиентская часть на WPF: Для обеспечения серверной части SneakersShop будет использован Windows Presentation Foundation (WPF). Это обеспечит интуитивно понятный дизайн, масштабируемость, безопасность и производительность сервера, а также быстрый доступ к базе данных сервиса. WPF предоставляет большой спектр средств для разработки интерфейсов и визуализации данных.

SmtpClient: Для обеспечения уведомлений пользователей о товарах используется стандартный класс .NET Fraemwork SmtpClient. Он служит для отправки почтовых сообщений через SMTP-сервер.

База данных SQL Server: Для хранения информации о кроссовках, пользователях будет использоваться SQL, стабильная и производительная реляционная система управления базами данных.

Подключение к Базе данных: Сервис SneakersShop будет взаимодействовать с базой напрямую, обеспечивая обмен данными о актуальном наличии каталога и клиентов.

Проект будет спроектирован с учетом принципов ООП, SOLID с использованием Microsoft SQL Server 2019 и .Net Framework 4.8

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

## **1.1 Анализ прототипов**

Для создания быстрого и динамического интернет-приложения для продажи кроссовок необходимо использовать современные технологии, которые обеспечат удобство пользователей и функциональность приложения. Тема интернет-приложения должна быть востребованной, а также актуальной. Прежде чем приступить к разработке интернет-приложения, необходимо проанализировать аналоги, чтобы понять, какие технологии используются для создания приложений подобного рода, а также выявить их недостатки и преимущества. И самое главное, найти в каждом аналоге свои уникальные особенности, которые сделают ваше приложение выдающимся. Для этого рассмотрим 3 аналога интернет-приложений для продажи кроссовок.

Аналог №1 – сайт «Nike» [[1]](#_Список_использованных_источников),

Nike является одним из крупнейших интернет-магазинов в мире. Основным преимуществом данного сайта является его минимализм и удобство.

Покупка продукции: Nike предлагает широкий ассортимент спортивной одежды, обуви и аксессуаров, включая кроссовки, футболки, шорты, штаны, куртки и многое другое.

Кастомизация продукции: Покупатели могут создавать уникальные товары, настраивая цвет, дизайн и материалы для кроссовок и других изделий.

Поиск продукции: Удобный поиск и фильтрация товаров по категориям, размерам, цветам и другим параметрам помогают покупателям найти именно то, что им нужно.

Информация о продукции: Каждый товар снабжен подробным описанием, фотографиями и техническими характеристиками, чтобы покупатели могли принять информированное решение.

Отзывы и рейтинги: Покупатели могут оставлять отзывы и оценки продукции, делая покупки более прозрачными и информативными для других пользователей.

Акции и скидки: Nike регулярно проводит акции и предлагает скидки на продукцию, что позволяет экономить на покупках.

Блог и статьи: Сайт предоставляет информацию о спортивных событиях, тренировках, здоровом образе жизни и многом другом через свой блог и статьи.

Поддержка клиентов: Nike предоставляет разнообразные варианты связи с командой поддержки клиентов, чтобы решать вопросы и помогать пользователям.

Технологические инновации: Nike представляет новейшие технологии и продукты, такие как умные кроссовки с интегрированными датчиками и приложениями для отслеживания физической активности.

Возможность предзаказа: Пользователи могут предзаказывать ожидаемые продукты, что позволяет им быть среди первых, кто получит доступ к новым коллекциям.

Глобальная доставка: Nike предоставляет возможность доставки продукции по всему миру, что делает их товары доступными для покупателей в разных странах.

Возврат и обмен: Сайт предоставляет информацию о политике возврата и обмена, что помогает пользователям в случае необходимости вернуть или обменять приобретенные товары.

Интерфейс интернет-магазина представлен на рисунке 1.1.

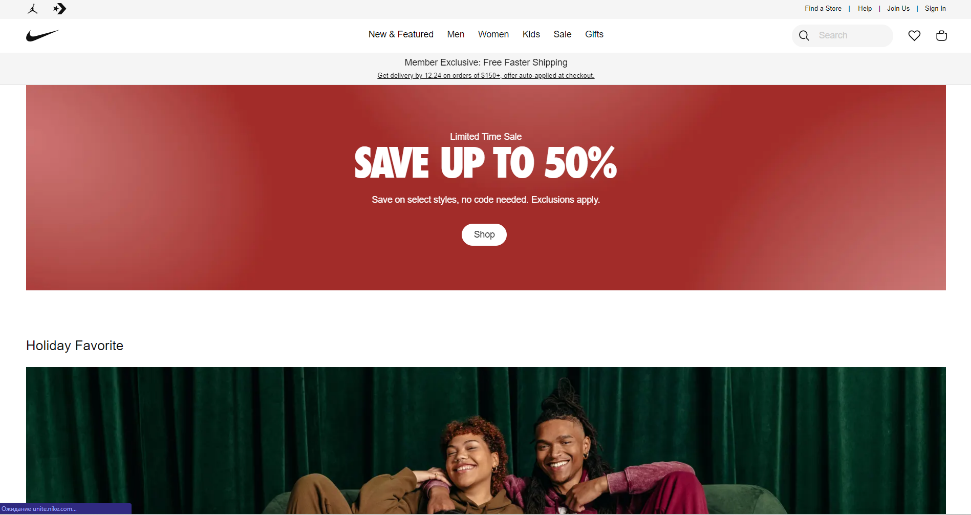


Рисунок 1.1 – Главный экран интернет-магазина «Nike»

Из минусов данного веб-сайта можно выделить то, что он долго загружается из-за большого количества товаров, которые находятся на сайте. Также из за большого количества ассортимента, тяжело выбрать модель которая именно нужна. Большое количество рекламных данных, которые могут не очень заинтересовать пользователя.

Аналог №2 – сайт Adidas [[2],](#_Список_использованных_источников)

Следующим аналогом будет сайт Adidas, который является одним из крупнейших интернет-магазинов и брендов мира.

Каталог товаров: Adidas предлагает разнообразные кроссовки и спортивную одежду, представленные в разных моделях, размерах и цветах.

Фильтры и категории: Пользователи могут легко найти желаемую продукцию, используя фильтры по категориям, размерам, цветам и другим параметрам.

Подробные описания: Каждый товар снабжен подробным описанием, включая характеристики, материалы, технологии и фотографии.

Рейтинги и отзывы: Пользователи могут оставлять отзывы и выставлять рейтинги продуктам, что помогает другим покупателям с выбором..

Акции и скидки: Adidas предоставляет информацию о текущих акциях и скидках, что позволяет экономить при покупке.

Опции доставки: Сайт предоставляет различные варианты доставки, включая стандартную и экспресс-доставку.

Отслеживание заказа: Зарегистрированные пользователи могут отслеживать статус своих заказов и узнавать предполагаемую дату доставки.

Возврат и обмен: Сайт предоставляет информацию о политике возврата и обмена, что обеспечивает уверенность покупателям.

Советы и рекомендации: Adidas может предоставлять советы и рекомендации по выбору правильного размера и уходу за продукцией.

Интеграция с социальными сетями: Пользователи могут делиться своими покупками и предпочтениями через социальные сети.

Информация о бренде: Сайт предоставляет информацию о бренде Adidas, его истории и ценностях.

Служба поддержки: Пользователи могут обратиться в службу поддержки для получения помощи и консультаций.

Интерфейс интернет-магазина представлен на рисунке 1.2.

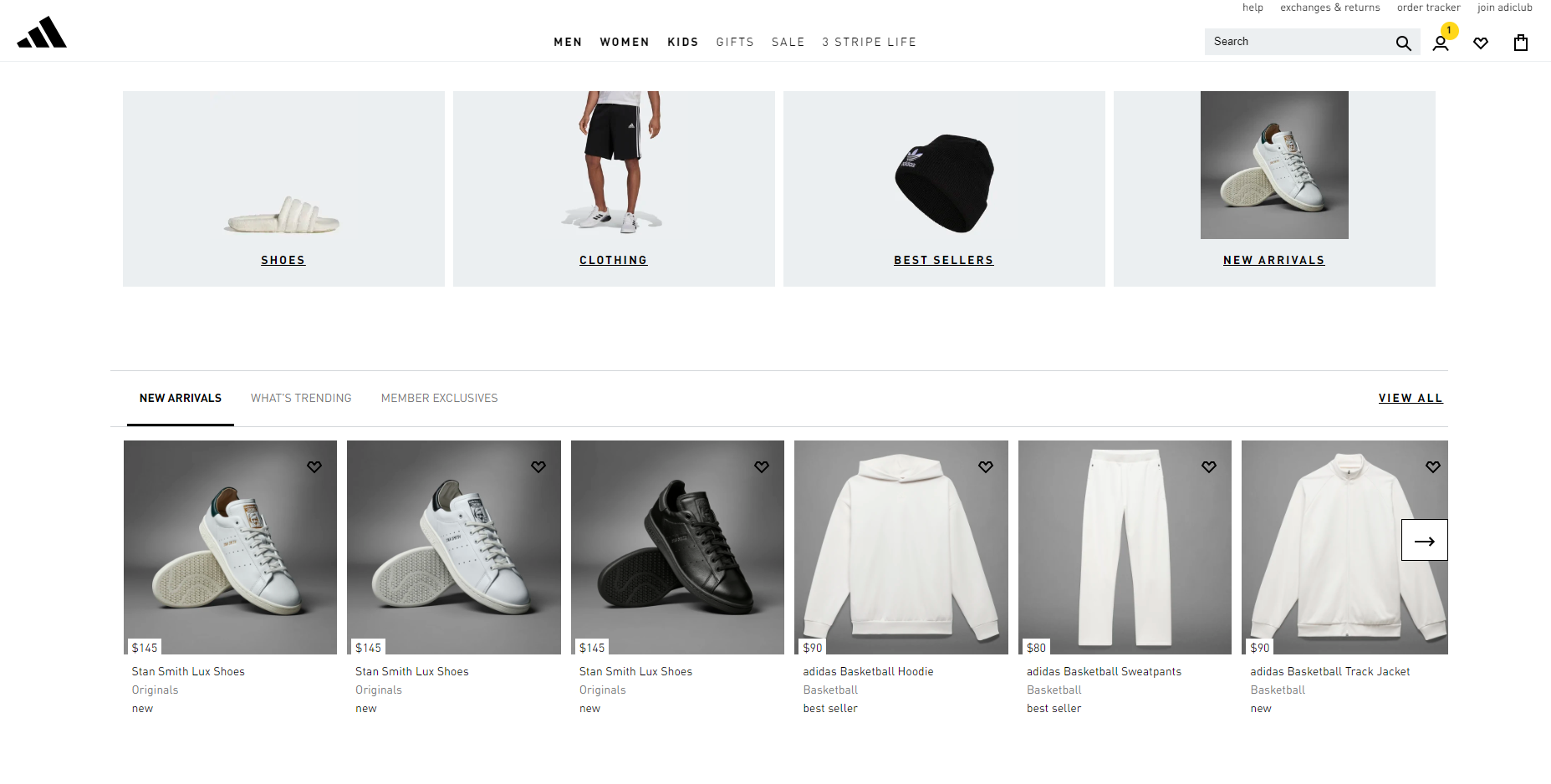


Рисунок 1.2 – Главный экран интернет-магазина «Adidas»

Из недостатков можно выделить то, сайт нагружен большим количеством информации: содержит новостные данные, которые могут отвлекать пользователя.

Аналог №3 – сайт Puma [[3],](#_Список_использованных_источников)

Puma – сайт, который также является крупнейшим интернет-магазином и брендом в мире.

Широкий ассортимент товаров: Puma предоставляет доступ к огромному выбору кроссовок, спортивной обуви, одежды и аксессуаров в различных стилях, размерах и цветах.

Фильтрация и сортировка товаров: Покупатели могут фильтровать и сортировать товары по различным критериям, таким как размер, цвет, стиль, цена и другие, чтобы найти идеальный продукт.

Подробные описания товаров: Каждый товар сопровождается подробным описанием, включая информацию о материалах, технологиях и характеристиках, чтобы помочь покупателям сделать информированный выбор.

Изображения и видео: Puma предоставляет высококачественные изображения и видеоматериалы для каждого товара, чтобы покупатели могли рассмотреть его со всех ракурсов.

Корзина и оформление заказа: Покупатели могут легко добавлять товары в корзину, просматривать ее и оформлять заказ с учетом выбранных товаров и способа доставки.

Информация о доставке и возврате: Puma.com предоставляет информацию о доступных методах доставки, сроках и стоимости, а также условиях возврата и обмена товаров.

Регистрация и профиль пользователя: Покупатели могут зарегистрировать аккаунт, где они могут сохранять свои настройки, адреса доставки и отслеживать статус своих заказов.

Служба поддержки и чат: Puma предоставляет контактную информацию для связи с клиентской поддержкой, включая онлайн-чат для оперативных ответов на вопросы.

Акции и скидки: Сайт предлагает информацию о текущих акциях, скидках и специальных предложениях для покупателей.

Мультиязычность и валюты: Puma обеспечивает мультиязычность и возможность выбора валюты, что делает его доступным для международных покупателей.

История заказов: Зарегистрированные пользователи могут просматривать историю своих заказов и отслеживать их статус.

Оценки и отзывы: Покупатели могут оставлять отзывы и оценки товаров, что помогает другим пользователям принимать более обоснованные решения при выборе товаров.

Информация о бренде: Сайт предоставляет информацию о бренде Puma, его истории, ценностях и миссии.

Интеграция с социальными сетями: Покупатели могут делиться своими покупками и отзывами через социальные сети, что способствует продвижению бренда.

Следование трендам и коллекциям: Puma активно обновляет свой ассортимент в соответствии с модными тенденциями и выпускает новые коллекции.

Интерфейс интернет-магазина представлен на рисунке 1.3.

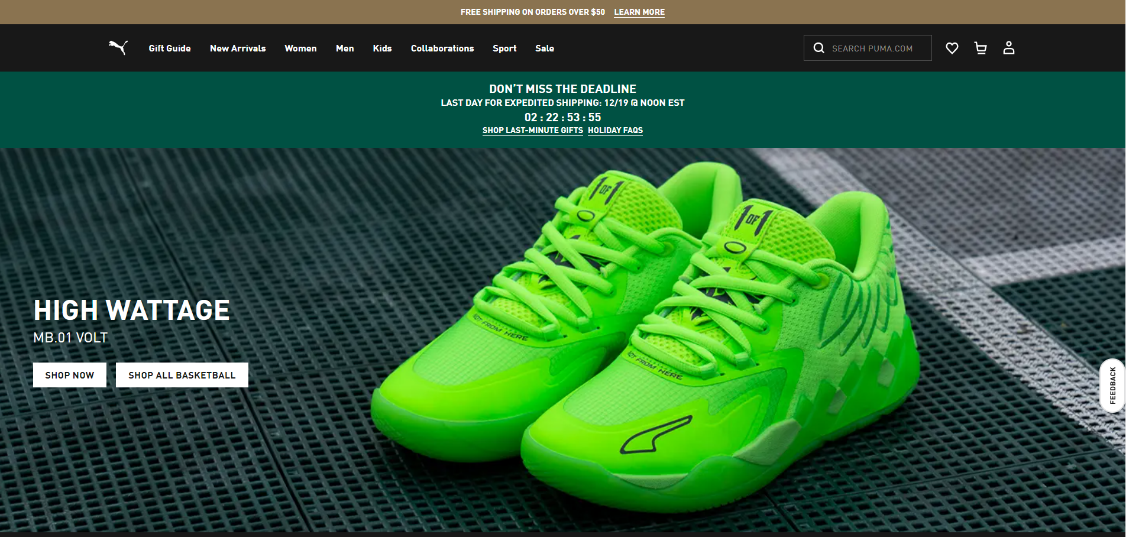


Рисунок 1.3 – Главный экран интернет-магазина «Puma»

Анализируя аналоги Nike, Adidas и Puma, можно понять, что все три интернет-магазина имеют широкий функционал для выбора кроссовок. Каждое из них имеет свой собственный уникальный каталог товаров, позволяет искать и сортировать модели, добавлять в избранное и корзину, оформлять заказа и отслеживать его местонахождение.

Общими моментами всех трех приложений являются простота и удобство использования. Все они предлагают производительный функционал для пользователей и вариативный каталог, что позволяет адаптировать интерне-магазин под свой выбор. Благодаря этим особенностям, каждый пользователь может выбрать нужный товар и создать пользовательский опыт, чтобы доверять этому сервису.

## **1.2 Требования к проекту**

Обзор аналогов интернет-магазинов кроссовок позволяет определить требования к разрабатываемой платформе. Платформа должна иметь следующие функции:

1. Управление базой данных:

- Сервис должен предоставлять администратору возможность управления базой данных. Это включает в себя добавление новых моделей кроссовок, редактирование информации о кроссовках и удаление их из базы данных.

2. Регистрация и авторизация:

-Пользователям должна быть предоставлена возможность регистрации на сервисе с созданием учетной записи, которая идентифицируется под уникальным номером.

- Зарегистрированные пользователи должны иметь возможность авторизации на платформе.

3. Редактирование личной информации:

- Зарегистрированные пользователи должны иметь возможность изменения своих учётных данных.

4. Публикация моделей в каталог:

- Администратору должна быть предоставлена возможность публикации кроссовок на платформу. Это позволит расширить каталог доступных моделей на платформе.

5. Поиск и сортировка:

- Платформа должна предоставлять пользователям возможность поиска кроссовок по различным критериям, таких как названию, цена, цвет, размер и тип.

6. Оформление заказа:

- Платформа должна предоставлять возможность заказа кроссовок пользователю. После подтверждения заказа пользователь может отслеживать статус заказа.

Эти требования определяют основной функционал программного средства. Реализация этих функций позволит пользователям успешно просматривать, выбирать и заказывать кроссовки, а также оценивать товар и работу персонала.

## **1.3 Вывод**

Итого, был проведен аналитический обзор аналогов интернет-магазинов кроссовок, которые уже существуют на рынке. Этот обзор позволил определить основные характеристики и функциональные возможности, которые необходимо предусмотреть в разрабатываемой системе. Также были определены функциональные требования приложения.

# **2 Анализ и проектирование архитектуры приложения**

# **2.1 Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET 7.0;
* язык программирования C#;
* расширяемый язык разметки XAML и текстовый формат передачи данных JSON;
* технология WPF;
* технология Entity Framework;
* SQL Server.

### **2.1.1 Microsoft Visual Studio 2022**

### Microsoft Visual Studio предоставляет множество инструментов и ресурсов для разработки, включая редактор кода, отладчик, систему управления версиями, дизайнер интерфейсов и другие инструменты, которые помогают разработчикам создавать высококачественное программное обеспечение. Эта IDE популярна среди разработчиков и используется для широкого спектра задач, от создания приложений настольных компьютеров до разработки мобильных приложений и веб-сервисов.

### **2.1.2 Программная платформа .NET 7**

Платформа .NET - .NET 7 - это платформа разработки, созданная Microsoft, которая предоставляет инструменты и технологии для создания разнообразных приложений, включая веб-приложения, настольные приложения, мобильные приложения и многое другое. Эта платформа включает в себя несколько компонентов, включая .NET Framework, .NET Core, .NET 7.

### **2.1.3 Язык программирования C#**

В качестве языка программирования используется C# – это современный, высокоуровневый, объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он был представлен в конце 2000 года и стал одним из ключевых элементов Microsoft .NET Framework - технологической платформы для разработки разнообразных приложений.

### **2.1.4 Расширяемый язык разметки XAML**

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML и текстовый формат передачи данных JSON.

XAML - это язык разметки, который используется в основном для создания пользовательских интерфейсов в приложениях, разработанных на платформах Microsoft, таких как Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), и Silverlight. XAML позволяет разработчикам определить структуру и внешний вид элементов интерфейса, таких как окна, кнопки, текстовые поля и др., с использованием текстового формата. Он отделяет дизайн от логики приложения, что упрощает разработку и редактирование пользовательского интерфейса.

### **2.1.5 Технология WPF**

WPF, или Windows Presentation Foundation, является одной из технологий, используемых для разработки графических пользовательских интерфейсов (GUI) в операционной системе Windows. WPF представляет собой часть технологического стека .NET Framework и была впервые представлена Microsoft в Windows Vista. Она предоставляет разработчикам мощные инструменты для создания современных и интерактивных пользовательских интерфейсов для приложений, работающих под управлением Windows.

WPF позволяет создавать интерфейсы с использованием декларативного языка разметки XAML (eXtensible Application Markup Language), который позволяет разработчикам описывать структуру и внешний вид интерфейса в виде разметки, подобной HTML.

WPF предоставляет продвинутые возможности рендеринга, включая поддержку векторной и растровой графики, анимацию, трансформации и многое другое.

WPF упрощает связывание данных между элементами пользовательского интерфейса и данными, что облегчает создание динамических приложений.

WPF позволяет определять стили и шаблоны для элементов интерфейса, что обеспечивает единообразный и красивый внешний вид приложения.

WPF предоставляет богатый набор предопределенных элементов управления, таких как кнопки, текстовые поля, таблицы и другие, а также позволяет создавать собственные элементы управления.

WPF интегрирована с другими технологиями .NET, такими как Windows Forms, и может использоваться для создания разнообразных приложений, включая десктопные приложения, приложения с поддержкой тачскринов, а также приложения для Windows Store.

### **2.1.6 Технология Entity Framework**

Entity Framework (EF) - это набор технологий и инструментов, разработанных Microsoft, для работы с данными в приложениях, созданных на платформе .NET. Он предоставляет абстракцию для доступа к базам данных и упрощает взаимодействие между приложением и базой данных, позволяя разработчикам работать с данными в объектно-ориентированной парадигме.

Entity Data Model, которая представляет собой абстрактное представление структуры данных в базе данных. Она включает в себя сущности (entities), связи между сущностями и маппинг между сущностями и таблицами в базе данных.

Entity Framework поддерживает два основных способа разработки приложений: Code-First, где модель данных определяется с использованием кода на C# (или других поддерживаемых языках), и Database-First, где модель данных извлекается из существующей базы данных.

Entity Framework интегрируется с LINQ, что позволяет разработчикам писать запросы к данным с использованием C#-подобного синтаксиса.операций, управление транзакциями, облегчение тестирования.

### **2.1.7 SQL Server**

SQL Server (Microsoft SQL Server) - это система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. SQL Server предназначен для хранения, анализа и управления данными, а также предоставляет средства для создания, управления и обслуживания баз данных. Он поддерживает структурированный язык запросов SQL (Structured Query Language) для работы с данными в базах данных.

SQL Server предоставляет механизм для создания, изменения и управления данными, хранящимися в реляционных таблицах. Обеспечивает транзакционную целостность данных, что означает, что изменения в базе данных могут быть атомарными, согласованными, изолированными и долговечными (свойства ACID). Предоставляет средства для управления доступом к данным и обеспечения их конфиденциальности и целостности. Включает оптимизатор запросов, который пытается выполнить SQL-запросы с максимальной эффективностью, выбирая оптимальные планы выполнения.Поддерживает репликацию данных, кластеризацию и другие механизмы масштабирования для обеспечения высокой доступности и производительности.Позволяет создавать хранимые процедуры и триггеры для автоматизации операций с данными и реагирования на события в базе данных.

SQL Server предлагает различные версии с разными возможностями и лицензионными моделями, включая бесплатную версию под названием SQL Server Express. SQL Server широко используется в предприятиях для хранения и обработки данных, а также для разработки приложений, связанных с базами данных.

### **2.1.8 Паттерн MVVM**

MVVM (Model-View-ViewModel) — это архитектурный паттерн, применяемый в разработке программного обеспечения для разделения пользовательского интерфейса (View) от бизнес-логики (Model) и их связывания через промежуточный слой, называемый ViewModel.

В MVVM, Model представляет данные и бизнес-логику приложения, включая операции чтения/записи данных из и в источники данных, такие как базы данных или веб-сервисы.

View отвечает за отображение данных и взаимодействие с пользователем. В ней не содержится бизнес-логики, и она должна быть максимально независимой от Model.

ViewModel служит промежуточным слоем между View и Model. В нем содержится логика, необходимая для обработки действий пользователя, обновления данных в Model и уведомления View о изменениях. ViewModel предоставляет свойства и команды, которые связываются с элементами пользовательского интерфейса во View, позволяя им отображать данные и реагировать на действия пользователя.

MVVM позволяет четко разделять ответственность между компонентами приложения, облегчает тестирование и повышает переиспользуемость кода. Этот паттерн также поддерживает параллельную разработку пользовательского интерфейса и бизнес-логики, так как разработчики могут независимо работать над своими частями приложения.

## **2.2 Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* редактирование учётной записи;
* широкий спектр оформления;
* просмотр каталога;
* поиск нужной модели;
* оформление заказа.

Для администратора:

* авторизация;
* рассылка сообщений;
* просмотр статистики приложения;
* просмотр всех аккаунтов;
* удаление аккаунтов;
* просмотр каталога;
* редактирование каталога;
* добавление товара
* удаление товара.

## **2.3 Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Описание базы данных будет в следующем разделе.

В программном средстве необходимо реализовать регистрацию и авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для регистрации входными данными являются логин, пароль, электронный адрес почты. Введенные данные, успешно прошедшие проверку, заносятся в базу данных.

Администратор обладает расширенными правами включая возможность добавления, удаления, редактирования кроссовок каталога; добавления, удаления, редактирования пользователей; управления базой данных сервиса, а также просмотра статистики и аналитики платформы.

Пользователи могут регистрироваться, авторизоваться, просматривать каталог, редактировать свои личные учётные данные, искать нужные модели кроссовок, а также удалять свою учётную запись.

На главной странице интернет-магазина авторизованный пользователь может иметь доступ к различным функциям и возможностям, которые обычно недоступны для неавторизованных пользователей. Вот некоторые из типичных действий, которые может выполнять авторизованный пользователь:

Оформление заказа: Авторизованный пользователь может оформлять заказа на нужный адрес;

Оставление отзывов и рейтингов: Пользователь может оставлять отзывы о моделях и ставить им рейтинги. Это может помочь другим пользователям при выборе кроссовок.

Личный кабинет: Авторизованный пользователь может управлять своим профилем, изменять личные данные, просматривать историю заказов и т. д.

Описание функциональности программного средства «SneackersShop» представлено на диаграмме User Case, изображенной на рисунке 2.1.



Рисунок 2.1 – Диаграмма вариантов использования

Разделение прав ролей учётных записей позволяет эффективно управлять доступом и ограничивать возможности использования интернет-магазина в соответствии с требованиями и политиками. При выходе или удаления учётной записи пользователь возвращается к окну регистрации, а кнопка для выхода из приложения его закрывает.

## **2.4 Вывод**

В данном разделе были выбраны основные средства разработки для приложения SneackersShop. В завершение, правильный выбор основных средств разработки для приложения SneackersShop обеспечит не только эффективное взаимодействие между его частями, но и обеспечит пользователей высококачественным и интуитивно понятным пользовательским опытом. Процесс разработки, спроектированный с учетом оптимальной структуры и инструментов, сделает приложение готовым к успешному внедрению на рынок, соответствуя современным требованиям в области электронной коммерции и пользовательских интерфейсов.

# **3 Проектирование программного средства**

## **3.1 Общая структура**

Общая структура программного средства SneackersShop состоит из реализации интерфейса, функциональности требований, а также технологий обеспечивающих взаимосвязь с базой данных. Структура представляет собой клиентское приложение. Ниже приведено подробное описание структуры этих частей.

### **3.1.1 Структура клиентской части**

Программное средство SneackersShop клиентской части имеет следующею структуру, представленную на рисунке 3.1.

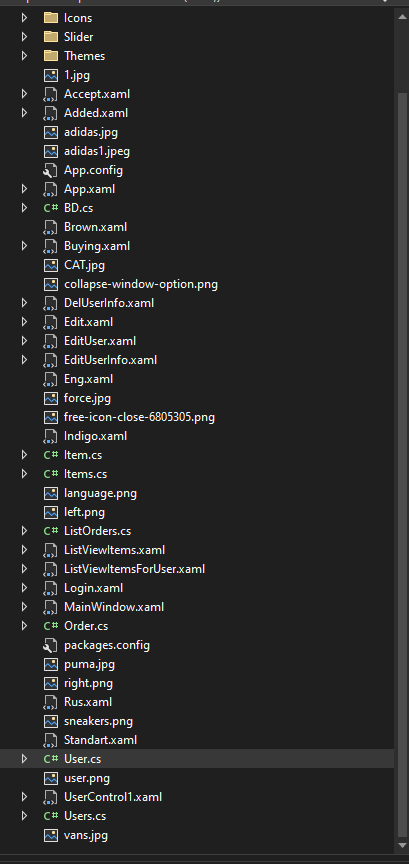


Рисунок 3.1 – Структура клиентской части проекта

Описание структуры основных папок клиентской части проекта представлено в таблице 3.1.

Таблица 3.1 – Описание клиенсткой структуры папок проекта

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Icons | Здесь используются иконки для представлений интерфейса. |
| Папка Slider | В этой папке хранятся изображения для слайдера. |
| Папка Themes | В данной папке хранятся цветовые оформления для представлений интерфейса. |

Описание файлов структуры Model представлено в таблице 3.2.

Таблица 3.2 – Описание файлов структуры Model

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл Item | Описание модели товаров. |
| Файл Items | Описание списка товаров. |
| Файл User | Описание модели пользователя. |
| Файл Users | Описание списка пользователей. |
| Файл Order | Описание модели заказа |
| Файл ListOrders | Описание списка заказов |

Описание файлов структуры View представлено в таблице 3.3.

Таблица 3.3 – Описание основных файлов структуры View

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Файл MainWindow | Хранит в себе страницу регистрации. |
| Файл Login | Хранит в себе страницу авторизации. |
| Файл ListViewItems | Хранит в себе страницу административной панели. |
| Файл ListViewItemsForUser | Хранит в себе страницу отображение каталога товаров и истории заказов. |
| Файл EditUserInfo | Хранит в себе окно для редактирования администратором информации пользователей. |
| Файл EditUser | Хранит в себе окно для редактирования личной информации пользователями. |
| Файл DelUserInfo | Хранит в себе окно для удаления администратором пользователей. |
| Файл Buying | Хранит в себе окно для оформления заказа. |

Описание файлов структуры ViewModel представлено в таблице 3.4.

Таблица 3.4 – Описание файлов и внутренних папок папки ViewModel

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Файл BD | Содержит класс BD. Данный класс описывает взаимосвязь представления, модели с базой данных. |

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

## **3.2 Взаимоотношение 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.

## **3.3 Модель базы данных**

Для успешной реализации целей по созданию и функционированию приложения SneackerShop, была разработана база данных, использующая систему управления реляционными базами данных SQL Server. Структура базы данных включает в себя семь основных таблиц: Users, Roles, Sneackers, Payments, Statuses,Orders и OrderHistory. База данных обеспечивает хранение и управление информацией о пользователях и ролях, кроссовках, платежах и заказах, статусе заказов, истории заказов. Подробности о структуре базы данных и скрипт ее создания представлены в приложении В.

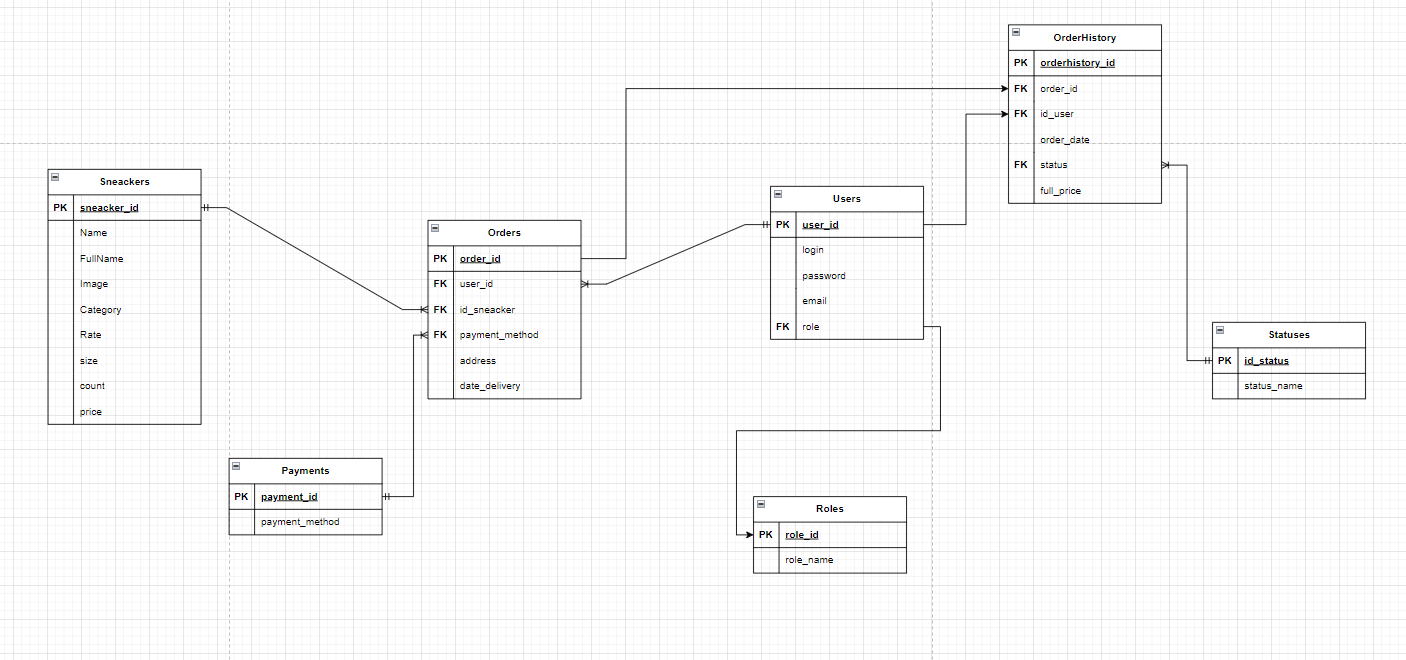


Рисунок 3.8 – База данных SneackerShop

В таблице 3.8 представлена структура таблицы Users, где каждая запись хранит важную информацию о пользователе.

Таблица 3.8 – Структура таблицы Users

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Столбец | Тип | | Ограничение | | | Описание |
| User\_id | integer | | Primary key | | | Уникальный идентификатор пользователя |
| login | text | | Not null | | | Имя пользователя |
| Email | | text | | Not null | Хранит email для входа в систему | |
| Password | | text | | Not null | Хранит пароль пользователя | |
| Roles | | integer | | Default 2, Foreign key | Хранит роль (user-2,admin-1) | |

В таблице 3.9 проиллюстрирована структура таблицы Sneackers, где каждая запись содержит важную информацию о модели кроссовок.

Таблица 3.9 – Структура таблицы Sneackers

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| sneaker\_id | integer | Primary key | Уникальный идентификатор модели кроссовок |
| Name | Varchar(50) | NOT NULL | Название бренда |
| FullName | Varchar(50) | NOT NULL | Полное название модели |
| Image | Varchar(20) | NOT NULL | Хранит путь к изображению |
| Category | Varchar(20) | NOT NULL | Категория модели |
| Rate | float | NOT NULL | Рейтинг модели |
| size | integer | NOT NULL – | Размер кроссовок |
| count | integer | NOT NULL | Количество товара |
| price | Decimal(10,2) | NOT NULL | Цена товара |

В таблице 3.10 представлена структура таблицы Orders, где каждая запись содержит важную информацию о заказах товаров.

Таблица 3.10 – Структура таблицы Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| order\_id | integer | Primary key | Уникальный идентификатор заказа |
| User\_Id | integer | Foreign key | Хранит id пользователя |
| id\_sneacker | integer | Foreign key | Хранит id модели кроссовок |
| payment\_method\_id | integer | Foreign key | Хранит id вида оплаты |
| user\_address | Varchar(50) |  | Хранит адресс клиента |
| date\_delivery | DATETIME |  | Дата доставки заказа |

В таблице 3.11 представлена структура таблицы Payments, где каждая запись содержит информацию о виде оплаты.

Таблица 3.11 – Структура таблицы Payments

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| payment\_id | integer | Primary key | Уникальный идентификатор вида оплаты |
| payment\_method | Varchar(50) | NOT NULL | Хранит вид оплаты |

В таблице 3.12 представлена структура таблицы Statuses, где каждая запись содержит информацию о статусе заказа.

Таблица 3.12 – Структура таблицы Statuses

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| id\_status | integer | Primary key | Уникальный идентификатор статуса заказа |
| status\_name | integer | NOT NULL | Статус заказа |

В таблице 3.13 представлена структура таблицы Roles, где каждая запись содержит информацию о статусе заказа.

Таблица 3.13 – Структура таблицы Roles

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| Role\_id | integer | Primary key | Уникальный идентификатор роли пользователя |
| Role\_name | Varchar(50) | - | Роль пользователя |

В таблице 3.14 представлена структура таблицы OrderHistory, где каждая запись содержит информацию о истории заказов каждого пользователя.

Таблица 3.14 – Структура таблицы OrderHistory

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Ограничение | Описание |
| orderhistory\_id | integer | Primary key | Уникальный идентификатор личного заказа пользователя |
| order\_id | integer | Foreign key | Уникальный идентификатор заказа |
| id\_user | integer | Foreign key | Уникальный идентификатор клиента |
| order\_date | DATETIME | NOT NULL | Дата заказа |
| status | integer | Foreign key | Уникальный идентификатор статуса заказа |
| full\_price | Decimal(10,2) | NOT NULL | Полная стоимость заказа |

В целом, структура базы данных Sneackers обеспечивает необходимую основу для хранения и управления информацией о товарах в приложении SneackersShop.

# **3.4. Проектирование последовательностей проекта**

Диаграмма последовательностей (или диаграмма взаимодействия) - это графическое представление взаимодействия различных элементов (объектов) в системе в рамках определенной последовательности событий. Обычно она используется для моделирования и визуализации взаимодействия между объектами в рамках определенного сценария или процесса.

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.

Диаграмма состоит из вертикальных линий, которые представляют различные объекты или компоненты, и горизонтальных стрелок, которые представляют сообщения, передаваемые между этими объектами. Вертикальные линии располагаются в порядке участия объектов во взаимодействии, а сообщения отображаются в виде стрелок, указывающих направление передачи информации.

Каждое сообщение в диаграмме последовательностей имеет указанное время или порядок выполнения, отображаемое по горизонтальной оси. Это позволяет понять, когда и в какой последовательности объекты взаимодействуют друг с другом.

Диаграмма последовательностей представлена в приложении Г.

## **3.5 Вывод**

В результате проектирования программного средства была разработана общая структура, включая взаимоотношение между классами и таблицами в базе данных. Определены основные принципы взаимодействия между компонентами приложения, что обеспечивает эффективное функционирование системы. Модель базы данных играет ключевую роль в обеспечении стабильной и эффективной работы.

# **4 Разработка функциональной модели и модели данных**

## **4.1 Основные модели программного средства**

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

* регистрация;
* авторизация;
* просмотр каталога кроссовок;
* добавление модели в каталог;
* поиск модели;
* изменение цветового оформления
* просмотр истории заказанных моделей кроссовок

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

## **4.2 Регистрация**

Логика регистрации содержится в файле MainWindow. Создаётся метод Check\_Reg который совершает проверку введенных данных.

После нажатия на кнопку зарегистрироваться, срабатывает этот метод. Затем вызывается метод CheckLog, который проверяет на существование уже логина введенного пользователем. После этого вызывается метод CheckEmail, который проверяет на существование уже аккаунта с введенным адресом электронной почты После проверки данных и успешной регистрации, пользователь перенаправляется на главную страницу, а его данные записываются в базу данных с помощью метода InsertUser. В случае возникновения ошибки, выводится соответствующее сообщение. Описание метода Check\_Reg представлено в листинге 4.1. Описание метода CheckLog представлено в листинге 4.2. Описание метода CheckEmail представлено в листинге 4.3. Описание метода InsertUser представлено в листинге 4.4.

|  |
| --- |
| private void Check\_Reg(object sender, RoutedEventArgs e)  {  username = tb1.Text;  password = tb2.Password;  email = UserControl1.email;  BD bd1 = new BD();  if (bd1.CheckLog(username) == false)  {  if (bd1.CheckEmail(email) == false)  {  if (tb1.Text == "Имя" || tb1.Text == "Name" || tb1.Text == "" || tb1.Text.Length > 10)  {  MessageBox.Show("Введите настоящее имя");  tb1.BorderBrush = Brushes.Red;  }  else  {  if (!char.IsUpper(tb1.Text[0]))  {  MessageBox.Show("Первая буква в логине должна быть заглавной!");  tb1.BorderBrush = Brushes.Red;  }  else if (tb2.Password.Length < 8)  {  tb1.BorderBrush = Brushes.Black;  MessageBox.Show("Пароль должен содержать минимум 8 символов!");  tb2.BorderBrush = Brushes.Red;  }  else if (!IsValidEmail(UserControl1.email))  {  MessageBox.Show("Введите корректный адрес электронной почты!");  }  else  {  tb2.BorderBrush = Brushes.Black;  tb1.BorderBrush = Brushes.Black;  bd1.InsertUser();  ListViewItemsForUser listView = new ListViewItemsForUser();  listView.NameAdmin.Text = tb1.Text;  login = tb1.Text;  listView.Show();  this.Close();  }  }  }  }  } |

Листинг 4.1 – Описание метода Check\_Reg

|  |
| --- |
| public bool CheckLog(string login)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Проверка существования пользователя по логину  string queryLogin = "SELECT COUNT(\*) FROM Users WHERE login = @login";  using (SqlCommand commandLogin = new SqlCommand(queryLogin, connection))  {  commandLogin.Parameters.AddWithValue("@login", login);  int userCountLogin = (int)commandLogin.ExecuteScalar();  if (userCountLogin > 0)  {  MessageBox.Show("RU: Пользователь с данным логином существует.\n ENG: A user with this login exists.");  return true;  }  else  {  return false;  }  }} } |

Листинг 4.2 – Описание метода CheckLog

|  |
| --- |
| public bool CheckEmail(string email)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Проверка существования пользователя по логину  string queryLogin = "SELECT COUNT(\*) FROM Users WHERE email = @email";  using (SqlCommand commandLogin = new SqlCommand(queryLogin, connection))  {  commandLogin.Parameters.AddWithValue("@email", email);  int userCountEmail = (int)commandLogin.ExecuteScalar();  if (userCountEmail > 0)  {  MessageBox.Show("RU: Пользователь с данной электронной почтой существует. \n ENG: A user with this email exists.");  return true;  }  else  {  return false;}}} } |

Листинг 4.3 – Описание метода CheckEmail

|  |
| --- |
| public void InsertUser()  {  string username1 = MainWindow.username;  string password1 = MainWindow.password;  string email = UserControl1.email;  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения к базе данных  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  using (SqlCommand command = new SqlCommand("InsertIntoUsers", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  command.Parameters.AddWithValue("@login", username1);  command.Parameters.AddWithValue("@password", password1);  command.Parameters.AddWithValue("@email", email);  command.ExecuteNonQuery();  }  } } |

Листинг 4.4– Описание метода InsertUser

Таким образом, данные пользователя будут проверяться относительно создания новых пользователей, и при отсутствии совпадений записывать в базу данных.

## **4.3 Авторизация**

Для графической части процесса входа в систему создан класс Login. В нем описана разметка для формы входа.

После нажатия на кнопку Login, срабатывает метод Check\_Log, который будет проверять данные. Затем вызывается метод CheckLogininLog, который проверяет на существование введенного логина пользователя. После проверки логина система вызывает метод CheckPassAndLoginLog, который сверяет пароль с введенным логином. После успешной проверки всех данных система вызывает метод CheckRole и сверяет какую роль имеет пользователь. Если роль администратора то пользователь переходит на административную панель, если роль пользователя, то пользователь переходит на клиентскую часть приложения. При не успешном прохождении сверки данных выводиться ошибка которая поможет решить проблему. Описание метода Check\_Log представлено в листинге 4.5. Описание метода CheckLoginLog представлено в листинге 4.6. Описание метода CheckPassAndLoginLog представлено в листинге 4.7. Описание метода CheckRole представлено в листинге 4.8

|  |
| --- |
| private void Check\_Log(object sender, RoutedEventArgs e)  {  username = tb1.Text;  password = tb2.Password;  BD bd1 = new BD();  if (bd1.CheckLoginLog(username) == false)  {  if (bd1.CheckPassAndLoginLog(username, password) == false)  {  if (bd1.CheckRole(username, password) == false)  {  ListViewItems listViewItem = new ListViewItems();  listViewItem.NameAdmin.Text = username;  listViewItem.Show();  this.Close();  }  else  {  ListViewItemsForUser listViewItemuser = new ListViewItemsForUser();  listViewItemuser.NameAdmin.Text = username;  listViewItemuser.Show();  this.Close();  }  }  }} |

Листинг 4.5 – Описание метода Check\_Log

|  |
| --- |
| public bool CheckRole(string login, string password)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string queryRole = "SELECT role FROM Users WHERE login = @login and password = @password";  using (SqlCommand commandLogin = new SqlCommand(queryRole, connection))  {  commandLogin.Parameters.AddWithValue("@login", login);  commandLogin.Parameters.AddWithValue("@password", password);  int userRole = (int)commandLogin.ExecuteScalar();  if (userRole == 2)  {  return true;  }  else  {  return false;  }  }  } |

Листинг 4.6 – Описание метода CheckRole

|  |
| --- |
| public bool CheckPassAndLoginLog(string login, string password)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string queryLogin = "SELECT COUNT(\*) FROM Users WHERE login = @login and password = @password";  using (SqlCommand commandLogin = new SqlCommand(queryLogin, connection))  {  commandLogin.Parameters.AddWithValue("@login", login);  commandLogin.Parameters.AddWithValue("@password", password);  int userCountLogin = (int)commandLogin.ExecuteScalar();  if (userCountLogin == 0)  {  MessageBox.Show("RU:Пароль неверный. Попробуйте ещё раз. \n ENG: The password is incorrect. Try again.");  return true;  }  else{  return false;}}} } |

Листинг 4.7 – Описание метода CheckPassAndLoginLog

|  |
| --- |
| public bool CheckLoginLog(string login)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Проверка существования пользователя по логину  string queryLogin = "SELECT COUNT(\*) FROM Users WHERE login = @login";  using (SqlCommand commandLogin = new SqlCommand(queryLogin, connection)){  commandLogin.Parameters.AddWithValue("@login", login);  int userCountLogin = (int)commandLogin.ExecuteScalar();  if (userCountLogin == 0){  MessageBox.Show("RU: Пользователь с данным логином не существует. Перейдите на окно Регистрация. \n ENG: A user with this login does not exist. Go to the Registration window. ");  return true;  }else{  return false;}}} } |

Листинг 4.8 – Описание метода CheckLoginLog

Таким образом метод Check\_Log сверяет данные с базой данные и если данные найдены разрешает пользователю доступ к сервису.

## **4.4 Просмотр каталога кроссовок**

Для графической части процесса просмотра каталога кроссовок в файле ListViewItemsForUser создан метод GetProducts, который отображает список товаров при нажатии на кнопку Products. И этот метод также производит поиск моделей кроссовок по называнию модели с помощью select-запроса.

Описание метода GetProducts представлено в листинге 4.9.

|  |
| --- |
| private List<Item> GetProducts()  {  List<Item> list = new List<Item>();  if (SearchBox.Text.Length != 0)  {  var query = from p in sneakers  orderby p.Name  select new  {  Name = p.Name,  FullName = p.FullName,  Image = p.Image,  Category = p.Category,  Rate = p.Rate,  Size = p.Size,  Count = p.Count,  Price = p.Price  };  foreach (var item in query)  {  if (item.FullName.ToUpper().IndexOf(SearchBox.Text.ToUpper()) >= 0) { list.Add(new Item(item.Name, item.FullName, item.Image, item.Category, item.Rate, item.Size, item.Count, item.Price)); }  fil = list;  }  }  else { list = sneakers;  fil = sneakers;  }  return list;  } |

Листинг 4.9 – Описание метода GetProducts

Таким образом, вывод каталога товаров включается в себя этапы создания списка и извлечения данных. Этот процесс обеспечивает безопасный и надежный способ хранения товаров каталога.

## **4.5** **Добавление модели в каталог**

Для графической части процесса добавления моделей кроссовок в каталог в файле ListViewItemsForUser создан метод Button\_Click\_2, который отрабатывает на нажатие мыши на кнопку. Метод создает список в который заполняет новые данные моделей кроссовок, после этого сам список обновляется и выводит все значения каталога. Также при добавлении в список создан метод AddSneacker, который подключается к базе данных и добавляет весь каталог моделей. Описание метода Button\_Click\_2 представлено в листинге 5. Описание метода AddSneacker представлено в листинге 5.1.

|  |
| --- |
| private void Button\_Click\_2(object sender, RoutedEventArgs e)  {  BD bd1 = new BD();  bd1.AddSneacker();  Items list = new Items();  XmlSerializer serializer = new XmlSerializer(typeof(Items));  using (FileStream stream = new FileStream("book.xml", FileMode.Open))  {  list = (Items)serializer.Deserialize(stream);  }  list.list.Add(new Item(Added.newfield1, Added.newfield2, Added.newfield3, Added.selectedValue, Double.Parse(Added.newfield5), Int32.Parse(Added.newfield6), Int32.Parse(Added.newfield7), Double.Parse(Added.newfield8)));  using (FileStream stream = new FileStream("book.xml", FileMode.Create))  {  serializer.Serialize(stream, list);  }  ListViewItems listViewItem = new ListViewItems();  listViewItem.NameAdmin.Text = ListViewItems.userlog;  listViewItem.Show();  this.Close();  } |

Листинг 5 – Описание метода Button\_Click\_2

|  |
| --- |
| public void AddSneacker()  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на вашу строку подключения к базе данных  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  using (SqlCommand command = new SqlCommand("AddSneaker", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  command.Parameters.AddWithValue("@Name", Added.newfield1);  command.Parameters.AddWithValue("@FullName", Added.newfield2);  command.Parameters.AddWithValue("@ImageFileName", Added.newfield3);  command.Parameters.AddWithValue("@Category", Added.selectedValue);  command.Parameters.AddWithValue("@Rate", Double.Parse(Added.newfield5));  command.Parameters.AddWithValue("@Size", Int32.Parse(Added.newfield6));  command.Parameters.AddWithValue("@Count", Int32.Parse(Added.newfield7));  command.Parameters.AddWithValue("@Price", Double.Parse(Added.newfield8));  command.ExecuteNonQuery();  }  } } |

Листинг 5.1 – Описание метода AddSneacker

Таким образом, добавление новых товаров включает в себя этапы создания списка и добавление данных. Этот процесс обеспечивает безопасный и надежный способ хранения товаров каталога.

## **4.6 Поиск модели**

Для графической части процесса поиска моделей кроссовок в файле ListViewItemsForUser создан метод PriceBar\_TextChanged, который отображает список товаров в указанном диапазоне цен. Помимо этого имеется метод ComboCat\_SelectionChanged, который обеспечивает сортировку товаров по заданным критериям. Описание метода PriceBar\_TextChanged представлено в листинге 5.2. Описание метода ComboCat\_SelectionChanged представлено в листинге 5.3.

|  |
| --- |
| private void PriceBar\_TextChanged(object sender, TextChangedEventArgs e)  {  List<Item> list = new List<Item>();  var query = from p in sneakers  orderby p.Price  where p.Price < Double.Parse(PriceBar.Text)  select new  {  Name = p.Name,  FullName = p.FullName,  Image = p.Image,  Category = p.Category,  Rate = p.Rate,  Size = p.Size,  Count = p.Count,  Price = p.Price  };  foreach (var item in query)  {  list.Add(new Item(item.Name, item.FullName, item.Image, item.Category, item.Rate, item.Size, item.Count, item.Price));  }  if (list.Count != 0) ListViewSneakers.ItemsSource = list;  ComboCat.SelectedItem = ComboCat.Items.Cast<ComboBoxItem>().FirstOrDefault(item => item.Name == "All");} |

Листинг 5.2 – Описание метода PriceBar\_TextChanged

|  |
| --- |
| private void ComboCat\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  List<Item> list = new List<Item>();  var query = from p in sneakers  orderby p.Name  select new  {  Name = p.Name,  FullName = p.FullName,  Image = p.Image,  Category = p.Category,  Rate = p.Rate,  Size = p.Size,  Count = p.Count,  Price = p.Price};  foreach (var item in query){  if (item.Category == ((ComboBoxItem)ComboCat.SelectedItem).Content.ToString()){  list.Add(new Item(item.Name, item.FullName, item.Image, item.Category, item.Rate, item.Size, item.Count, item.Price));  }else if (((ComboBoxItem)ComboCat.SelectedItem).Name == "All"){  ListViewSneakers.ItemsSource = sneakers;  return;}}ListViewSneakers.ItemsSource = list;} |

Листинг 5.3 – Описание метода ComboCat\_SelectionChanged

## **4.7 Изменение цветового оформления интерфейса**

Для графической части процесса изменения цветового оформления в файле используется метод Button\_Click\_22, который меняет темы оформления в зависимости от изменения его на страницы регистрации. Помимо этого используется структура WPF ResourceDictionary, которая сохраняет состояние цветового оформления на всех окнах до момента её изменения. Описание метода Button\_Click\_22 представлено в листинге 5.4.

|  |
| --- |
| private void Button\_Click\_22(object sender, RoutedEventArgs e)  {  {  var uri = new Uri("Standart.xaml", UriKind.Relative);  ResourceDictionary resourceDict = Application.LoadComponent(uri) as ResourceDictionary;  Application.Current.Resources.Clear();  Application.Current.Resources.MergedDictionaries.Add(resourceDict);  }  }  private void Button\_Click\_23(object sender, RoutedEventArgs e)  {  {  var uri = new Uri("Brown.xaml", UriKind.Relative);  ResourceDictionary resourceDict = Application.LoadComponent(uri) as ResourceDictionary;  Application.Current.Resources.Clear();  Application.Current.Resources.MergedDictionaries.Add(resourceDict);  }  }  private void Button\_Click\_24(object sender, RoutedEventArgs e)  {  {  var uri = new Uri("Indigo.xaml", UriKind.Relative);  ResourceDictionary resourceDict = Application.LoadComponent(uri) as ResourceDictionary;  Application.Current.Resources.Clear();  Application.Current.Resources.MergedDictionaries.Add(resourceDict);  } } |

Листинг 5.4 – Описание метода Button с пользованием ResourceDictionary

## **4.8 Просмотр истории заказанных моделей кроссовок**

Для графической части процесса просмотра процесса создан класс Orders который хранит свойства заказа пользователей. После этого создаётся список, который будет хранить все заказы пользователей. Для этого списка создается метод LoadDataFromDatabaseUser, который извлекает из базы данных значения заказов и записывает их в созданный список. Описание метода LoadDataFromDatabaseUser представлено в листинге 5.5.

|  |
| --- |
| public void LoadDataFromDatabaseUser(string username)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на свою строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string sqlQuery = "SELECT \* FROM OrderDetails WHERE Login = @Username";  using (SqlCommand command = new SqlCommand(sqlQuery, connection))  {  command.Parameters.AddWithValue("@Username", username);  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  // Извлечение данных из SqlDataReader и создание объекта Order  Order order = new Order(  Login: reader["Login"].ToString(),  Name: reader["Name"].ToString(),  Full\_Name: reader["Full\_Name"].ToString(),  Category: reader["Category"].ToString(),  Size: reader["size"].ToString(),  User\_Address: reader["user\_address"].ToString(),  Order\_Date: Convert.ToDateTime(reader["order\_date"]),  Date\_Delivery: Convert.ToDateTime(reader["date\_delivery"]),  Status\_Name: reader["Status\_name"].ToString(),  Payment\_Method: reader["payment\_method"].ToString(),  Price: Convert.ToDecimal(reader["price"])  );  // Добавление объекта Order в список  this.Add(order);  }  }  }  connection.Close();} } |

Листинг 5.4 – Описание метода LoadDataFromDatabaseUser

## **4.9 Вывод**

В ходе реализации программного средства Sneackers были активно разработаны и интегрированы основные классы и методы, обеспечивающие корректное выполнение ключевых функций. Реализованы функционалы регистрации, авторизации, просмотра каталога кроссовок, добавление модели в каталог, поиск модели, изменение цветового оформления, а также просмотр истории заказанных моделей кроссовок. Все функции прошли тщательное тестирование, подтверждающее их надежную работу, что обеспечивает пользовательский опыт без сбоев и согласно ожиданиям пользователей

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

## **Тестирование авторизации и регистрации**

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим 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.

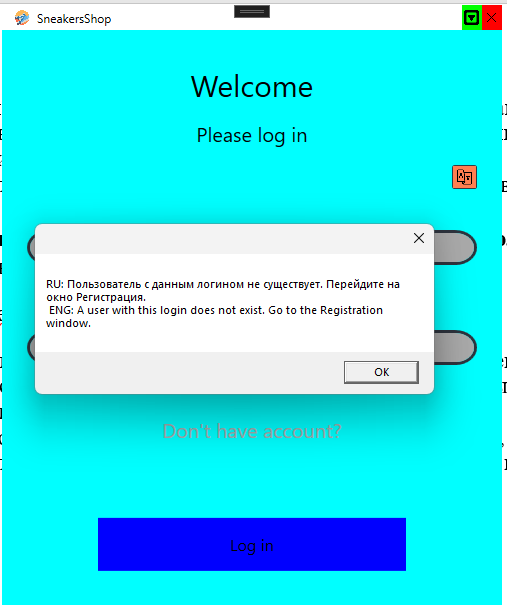


Рисунок 5.1 – Обработка пустых полей при авторизации

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

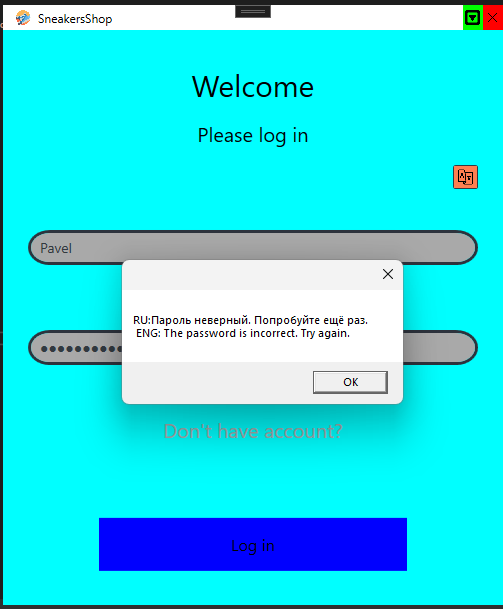


Рисунок 5.2 – Обработка некорректных данных

При регистрации может возникнуть ситуация, когда происходит попытка создать аккаунт с логином, который уже занят. Для предотвращения этой ситуации было принято решение внедрить проверку уникальности логина в процесс регистрации. В предусловии учитывается существование аккаунта с логином andrysa.shev.133@gmail.com. В случае попытки создания еще одного аккаунта с таким же логином, процесс регистрации будет корректно обработан, и результат этого теста отобразится на рисунке 5.3. Это обеспечивает более надежное и безошибочное управление процессом регистрации, предотвращая создание дубликатов логинов в системе.



Рисунок 5.3 – Обработка ввода уже зарегистрированного login

Дальше бы я хотел попробовать ввести некорректные данные. Например, ввести пароль, состоящий из 3 символов. Результат данного тестирования представлен на рисунке 5.4.

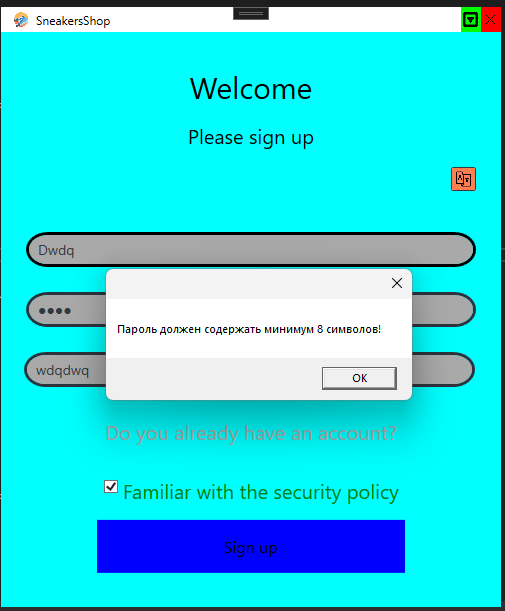


Рисунок 5.4 – Результат c паролем, состоящим из 4 символов

Тестирование с некорректным email будет отражено на рисунке 5.5.

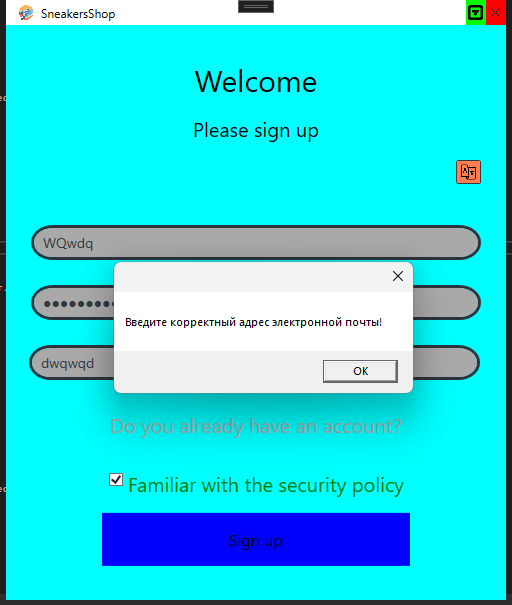


Рисунок 5.5 – Результат c некорректным email

Результаты тестирования показал корректную работу авторизации и регистрации.

## **Тестирование добавления кроссовок**

Для начала тестирования была проведена попытка добавления модели кроссовок с пустыми полями, в том числе и с незаполненными обязательными полями, такими как бренд, размер, цена и другие. Ожидаемый результат включает в себя корректную обработку запроса системой и вывод соответствующего сообщения об ошибке, указывающего на необходимость заполнения всех обязательных полей. Результат тестирования данной сценарий зафиксирован на рисунке 5.6. Полученные данные из теста позволяют утверждать, что система успешно обнаруживает отсутствие обязательных данных и предоставляет пользователю информативное уведомление о необходимости их заполнения.

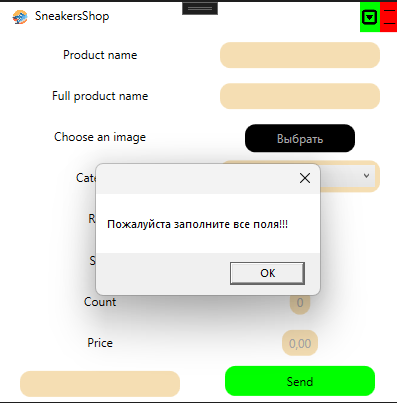


Рисунок 5.6 – Результат c пустыми полями

Дополнительно была проведена проверка на отрицательные значения полей Count и Price. Здесь осуществлялась попытка добавления модели кроссовок со значением количества равным -1 и значением цены равным -20,20. Результаты тестирования представлен на рисунках 5.7,5.8.



Рисунок 5.7 – Результат c отрицательным значением количества

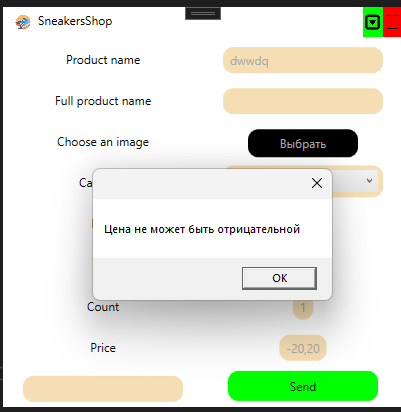


Рисунок 5.8 – Результат c отрицательным значением цены

Далее после полного ввода данных модели кроссовок система дополнительно выводит окно на запрос подтверждения добавления модели. Результат отражен на рисунке 5.7.



Рисунок 5.7 – Запрос на дополнительное подтверждение

В итоге, процесс добавления кроссовок в систему был успешно протестирован, и все вышеописанные сценарии были корректно обработаны. Различные тестовые сценарии, такие как попытки добавления кроссовок с пустыми полями, дополнительная проверка на подтверждение, и аудиофайлами других форматов, позволили убедиться в правильной функциональности системы.

## **5.3 Тестирование панели администратора**

В контексте тестирования панели администратора, основной функционал включает управление кроссовками и пользователями. Администратор обладает полномочиями по добавлению, редактированию и удалению кроссовок в системе. Также ему доступен контроль над пользователями, включая возможность редактирования и удаления пользовательских аккаунтов.

Одним из важных аспектов управления пользователями является возможность администратора удалить аккаунты с непристойными именами. Результат представлен на рисунке 5.9.

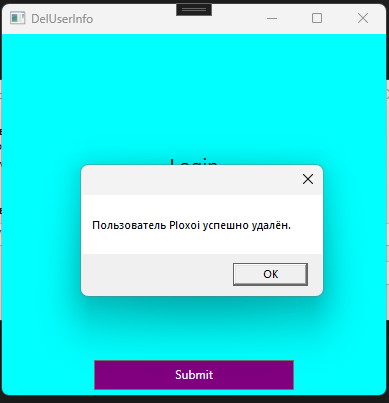


Рисунок 5.9 – Результат удаление пользователя

Также если администратор по ошибке попытается удалить свой аккаунт. Должна быть предупреждение, что свой аккаунт он удалить не сможет. Результат данного тестирования показан на рисунке 5.10.

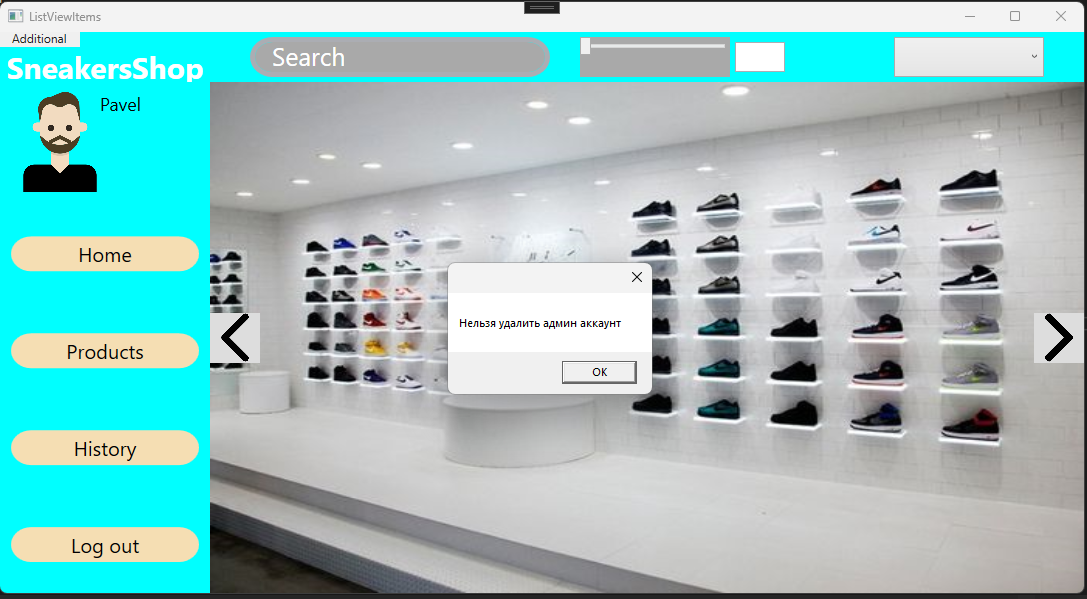


Рисунок 5.10 – Результат удаления своего аккаунта

## **5.4 Вывод**

В результате проведенного тестирования, было подтверждено успешное функционирование всех компонентов программного средства SneackersShop. Процессы добавления кроссовок на площадку успешно протестированы и соответствуют заявленным функциональным требованиям. Панель администратора подтверждает свою работоспособность. Полученные результаты тестирования подтверждают надежность и эффективность программного средства SneackersShop.

# **6 Руководство по установке и использованию**

При первом запуске приложения будет открыто окно регистрации, которое показано на рисунке 6.1. Поле адреса должно содеражать @gmail.com. Поле имени должно быть не более 10 символов. Пароль должен состоять от 8 символов.

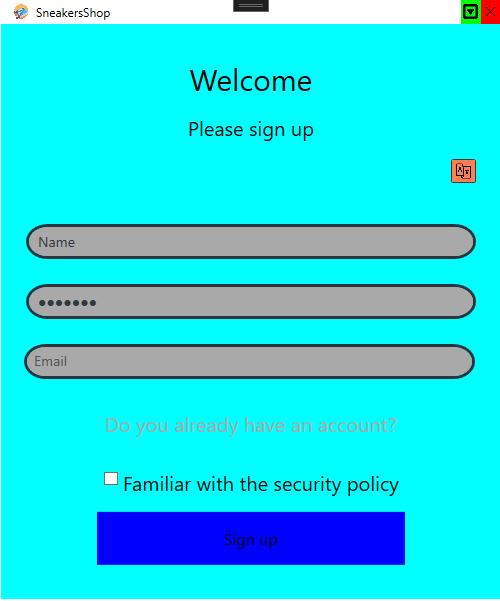


Рисунок 6.1 – Регистрации

Если у клиента уже имеется аккаунт, он может нажать на надпись **Do you already have an account?** для того, чтобы войти в свой аккаунт.

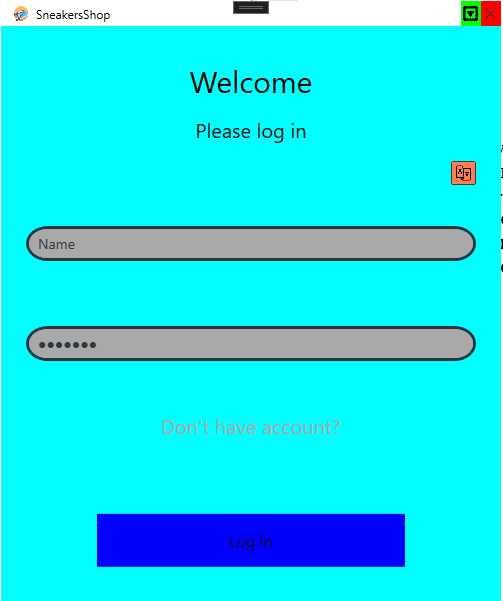


Рисунок 6.2 – Окно авторизации

После успешной регистрации пользователя перекинет на главную страницу. Главная страница приложения обеспечивает удобную навигацию с панелью слева, предоставляя доступ к основным разделам, таким как Home, Products, History, а также возможность управления профилем и выхода из приложения. Главная страница приложения представлена на рисунке 6.3.

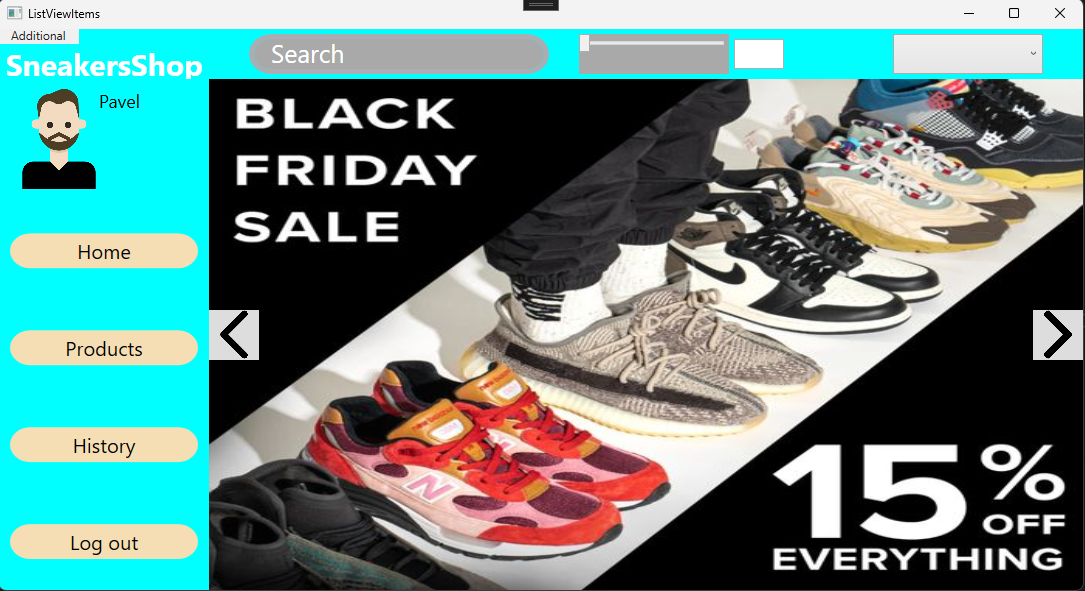


Рисунок 6.3 – Главная страница

Для поиска любой модели кроссовок необходимо перейти в раздел Products, где будут отображены карточки товаров кроссовок Пользователь может просматривать и заказывать кроссовки, выбрав интересующую категорию и стоимость, или воспользоваться полем поиска, введя название модели. Страница каталога, с отображением моделей кроссовок, представлена на рисунке 6.4.

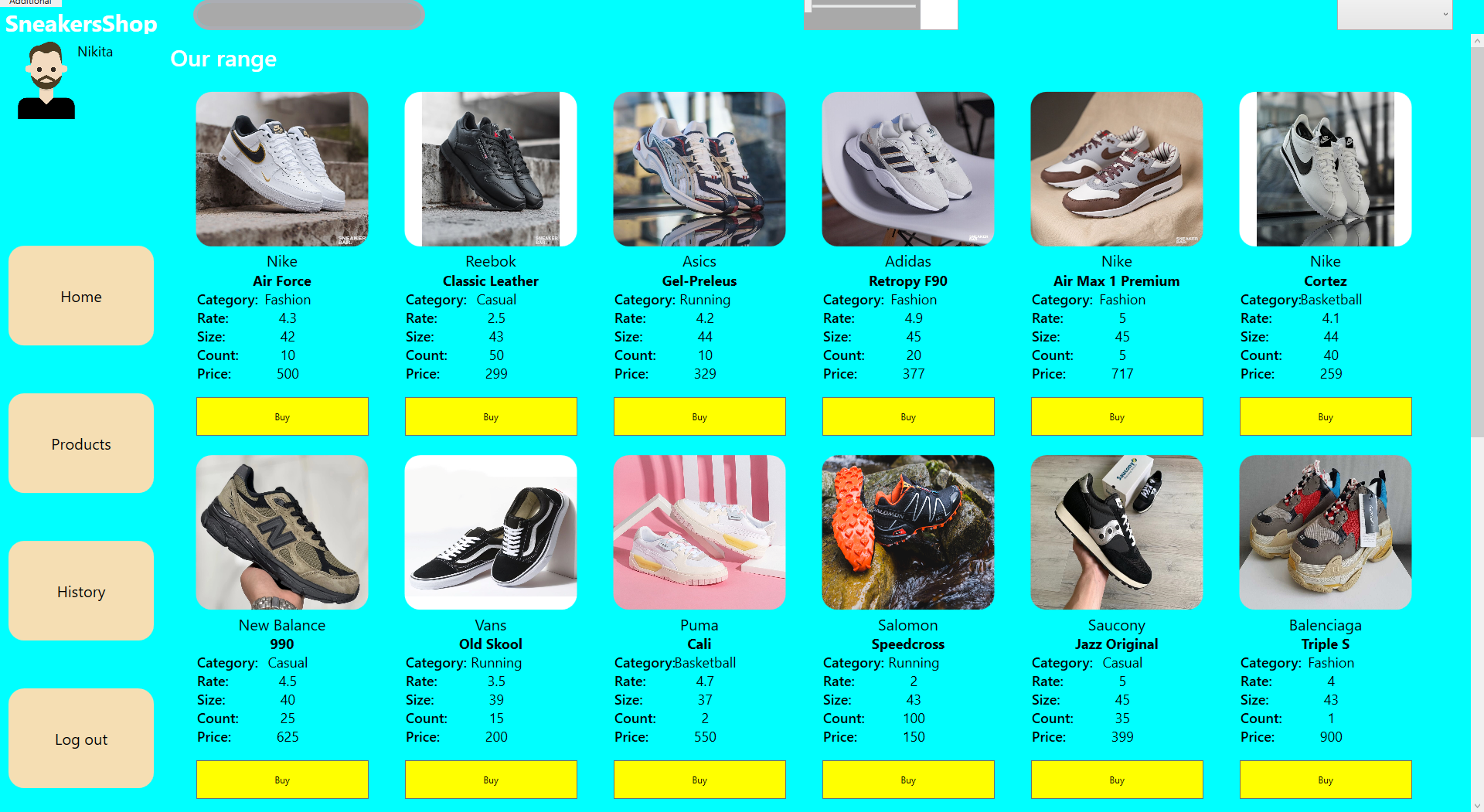


Рисунок 6.4 – Страница каталога кроссовок

Пользователь также имеет возможность изменить свои личные данные такие как логин и пароль. Для этого необходимо перейти на меню Additional(Дополнительно), где доступна функция редактирования профиля. Визуальное представление страницы редактирования профиля представлено на рисунке 6.6.

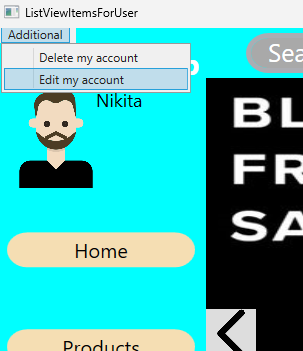


Рисунок 6.5 – Страница каталога кроссовок

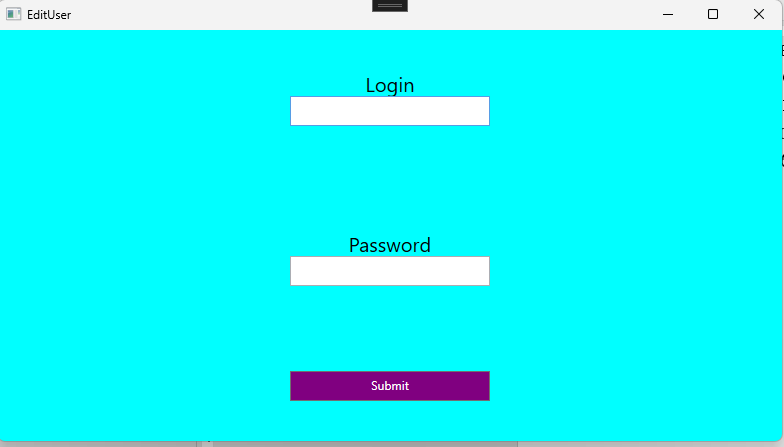


Рисунок 6.6 – Страница редактирования профиля

SneackersShop предоставляет пользователю возможность исследовать, добавлять и заказывать кроссовки. В дополнение к этому, приложение обеспечивает функционал для настройки профиля пользователя, что позволяет создать персонализированный опыт и удобное взаимодействие с интерфейсом.

# **Заключение**

В ходе разработки приложения SneackersShop были успешно реализованы его основные функциональности, предоставляя пользователям возможности добавления, заказа различных моделей кроссовок, просмотра заказов. Особое внимание уделено персонализации пользовательского опыта через настройку профиля, что способствует повышению удобства использования. Использование технологии Windows Presentation Foundation (WPF) и языка программирования C# обеспечивает современный и интуитивно понятный интерфейс.

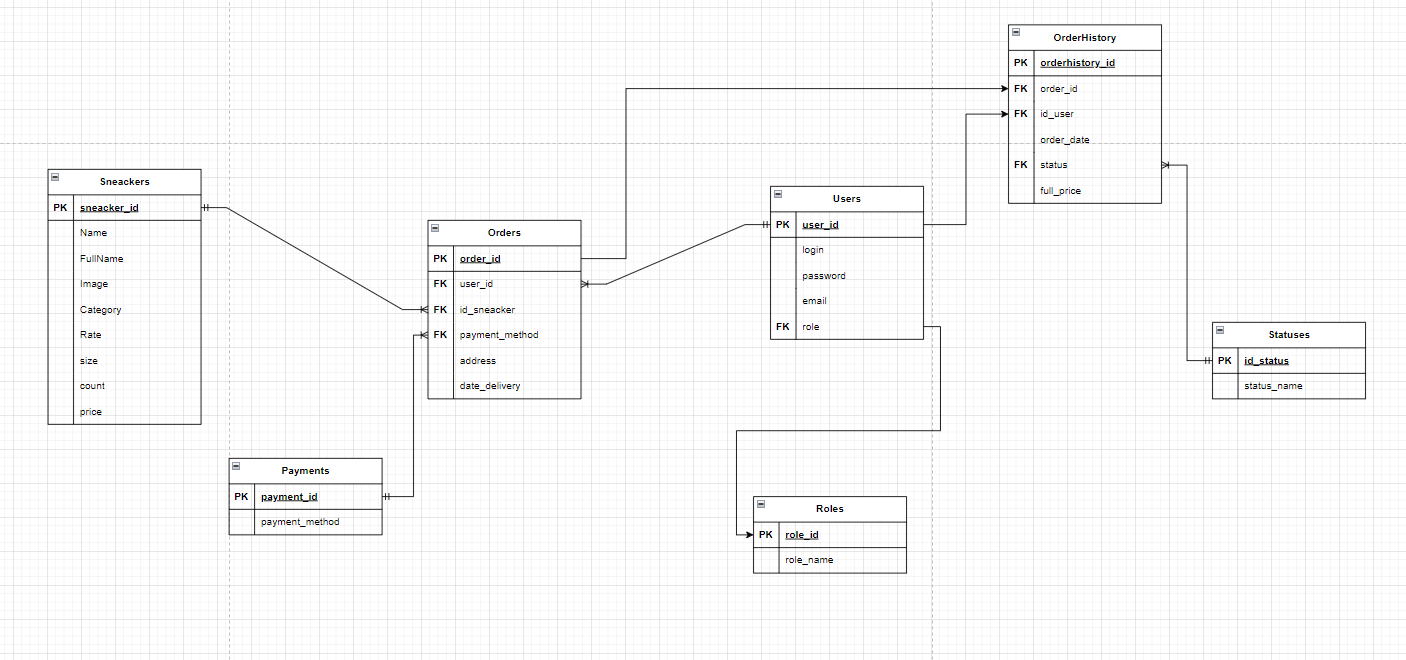
При разработке SneackersShop активно использовались принципы SOLID. В результате, приложение обеспечивает эффективное управление данными, высокую отзывчивость.

SneackersShop предоставляет не только возможности для заказа кроссовок, но и создает полное погружение в мир моды и стиля. В целом, SneackersShop успешно реализует свою задачу, предоставляя удобный и персонализированный пользовательский опыт в мире спортивной обуви.

# **Список использованных источников**

1. Nike [Электронный ресурс] / Режим доступа: https://nike.com – Дата доступа: 03.09.2023.
2. Adidas [Электронный ресурс] / Режим доступа: https://adidas.com/ – Дата доступа: 03.09.2023.
3. Puma [Электронный ресурс] / Режим доступа: https://puma.com / – Дата доступа: 03.09.2023.
4. Руководство по MS SQL Server 2022 [Электронный ресурс] / Режим доступа: https://metanit.com/sql/sqlserver/ – Дата доступа: 18.01.2023.
5. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 12.12.2023.
6. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 12.12.2023.
7. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 25. 12.12.2023.
8. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с
9. Взаимодействие с базой данных . [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/wpf/19.1.php/ – Дата доступа: 21.12.2023.

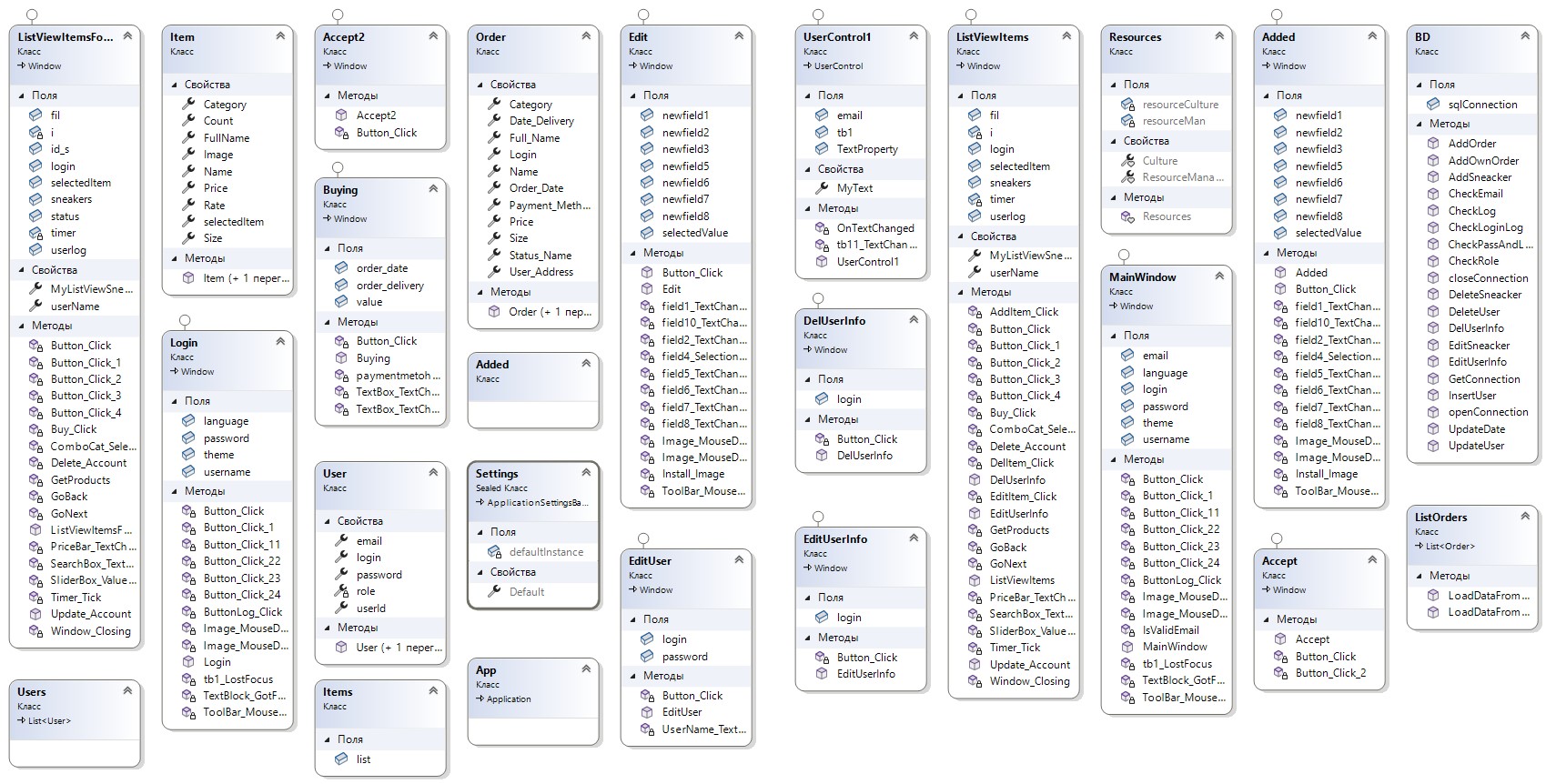
# Приложение А Диаграмма базы данных



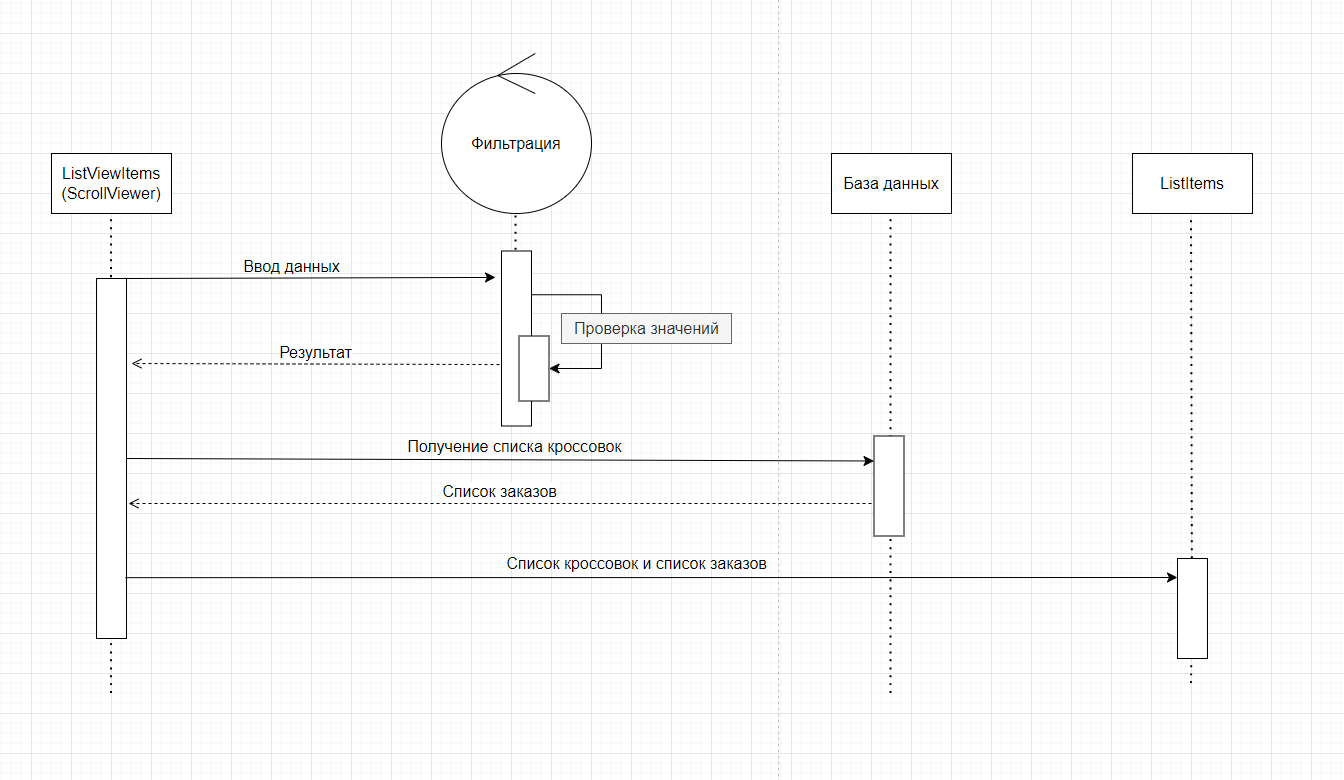
# Приложение Б Диаграмма вариантов использования(Use-Case)



# Приложение В Диаграмма классов



# Приложение Г Диаграмма последовательности



# Приложение Д Листинги кода

|  |
| --- |
| public void LoadDataFromDatabase()  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True"; // Замените на свою строку подключения  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string sqlQuery = "SELECT \* FROM OrderDetails"; // Запрос на выборку данных из представления  using (SqlCommand command = new SqlCommand(sqlQuery, connection))  {  using (SqlDataReader reader = command.ExecuteReader())  {  while (reader.Read())  {  // Извлечение данных из SqlDataReader и создание объекта Order  Order order = new Order(  Login: reader["Login"].ToString(),  Name: reader["Name"].ToString(),  Full\_Name: reader["Full\_Name"].ToString(),  Category: reader["Category"].ToString(),  Size: reader["size"].ToString(),  User\_Address: reader["user\_address"].ToString(),  Order\_Date: Convert.ToDateTime(reader["order\_date"]),  Date\_Delivery: Convert.ToDateTime(reader["date\_delivery"]),  Status\_Name: reader["Status\_name"].ToString(),  Payment\_Method: reader["payment\_method"].ToString(),  Price: Convert.ToDecimal(reader["price"])  );  // Добавление объекта Order в список  this.Add(order);}}}  connection.Close();} } |

Листинг 1 – Метод LoadDataFromDatabase класса ListOrders<Order>

|  |
| --- |
| private void Button\_Click(object sender, RoutedEventArgs e)  {  username = tb1.Text;  password = tb2.Password;  email = UserControl1.email;  BD bd1 = new BD();  if (bd1.CheckLog(username) == false)  {  if (bd1.CheckEmail(email) == false)  {  if (tb1.Text == "Имя" || tb1.Text == "Name" || tb1.Text == "" || tb1.Text.Length > 10)  {  MessageBox.Show("Введите настоящее имя");  tb1.BorderBrush = Brushes.Red;  }  else  {  if (!char.IsUpper(tb1.Text[0]))  {  MessageBox.Show("Первая буква в логине должна быть заглавной!");  tb1.BorderBrush = Brushes.Red;  }  else if (tb2.Password.Length < 8)  {  tb1.BorderBrush = Brushes.Black;  MessageBox.Show("Пароль должен содержать минимум 8 символов!");  tb2.BorderBrush = Brushes.Red;  }  else if (!IsValidEmail(UserControl1.email))  {  MessageBox.Show("Введите корректный адрес электронной почты!");  }  else  {  tb2.BorderBrush = Brushes.Black;  tb1.BorderBrush = Brushes.Black;  bd1.InsertUser();  ListViewItemsForUser listView = new ListViewItemsForUser();  listView.NameAdmin.Text = tb1.Text;  login = tb1.Text;  listView.Show();  this.Close();  }  }  }  }} |

Листинг 2 – Валидация окна Регистрации

|  |
| --- |
| private DispatcherTimer timer;  private void GoBack(object sender, RoutedEventArgs e)  {  i--;  if (i < 1)  {  i = 5;  }  PicHolder.Source = new BitmapImage(new Uri(@"Slider/" + i + ".jpg", UriKind.Relative));  }  private void GoNext(object sender, RoutedEventArgs e)  {  i++;  if (i > 5)  {  i = 1;  }  PicHolder.Source = new BitmapImage(new Uri(@"Slider/" + i + ".jpg", UriKind.Relative));  }  private void Timer\_Tick(object sender, EventArgs e)  {  i++;  if (i > 5)  {  i = 1;  }  PicHolder.Source = new BitmapImage(new Uri(@"Slider/" + i + ".jpg", UriKind.Relative));} |

Листинг 3 – Реализация cлайдера на странице Home

|  |
| --- |
| private void Install\_Image(object sender, RoutedEventArgs e)  {  // Создаем диалоговое окно для выбора файла  Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();  // Устанавливаем фильтр для выбора только изображений  dlg.DefaultExt = ".jpg|.jpeg|.png";  dlg.Filter = "Image files (\*.jpg, \*.jpeg, \*.png)|\*.jpg;\*.jpeg;\*.png";  // Открываем диалоговое окно и проверяем, что пользователь нажал кнопку "OK"  Nullable<bool> result = dlg.ShowDialog();  if (result == true)  {  // Получаем путь к выбранному файлу и сохраняем его в строку  string filename = dlg.FileName;  field10.Text = filename;  }} |

Листинг 4 – Установка изображения товара через проводник

|  |
| --- |
| public void DeleteUser()  {  string username23 = ListViewItems.login;  string username24 =ListViewItemsForUser.userlog;  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True";  int userId;  // Здесь мы устанавливаем соединение с базой данных и получаем идентификатор пользователя  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Запрос для получения идентификатора пользователя  using (SqlCommand selectCommand = new SqlCommand("SELECT user\_id FROM users WHERE login = @Username", connection))  {  if (!string.IsNullOrEmpty(username23))  {  selectCommand.Parameters.AddWithValue("@Username", username23);  }  else  {  selectCommand.Parameters.AddWithValue("@Username", username24);  }  MessageBox.Show(username23);  MessageBox.Show(username24);  userId = (int)selectCommand.ExecuteScalar();  }  // Удаление пользователя с использованием хранимой процедуры  using (SqlCommand command = new SqlCommand("DeleteUserByLogin", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  command.Parameters.AddWithValue("@UserId", userId);  if (!string.IsNullOrEmpty(username23))  {  command.Parameters.AddWithValue("@Login", username23);  }  else  {  command.Parameters.AddWithValue("@Login", username24);  }  command.ExecuteNonQuery();  }  } } |

Листинг 5 – Удаление пользователя с учётом конфликта внешних ключей таблиц

|  |
| --- |
| public void UpdateDate()  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True";  int newstatus;  DateTime now = DateTime.Now;  if (now > Buying.order\_delivery)  {  newstatus = 1;  }  else  {  newstatus = 2;  }  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Создание транзакции  SqlTransaction transaction = connection.BeginTransaction();  try  {    using (SqlCommand updateCommand = new SqlCommand("UPDATE OrdersHistory SET status = @NewStatus", connection))  {  updateCommand.Parameters.AddWithValue("@NewStatus", newstatus);  updateCommand.Transaction = transaction;  updateCommand.ExecuteNonQuery();  }  // Подтверждение транзакции  transaction.Commit();  }  catch (Exception ex)  {  // Обработка ошибок и откат транзакции при необходимости  transaction.Rollback();  Console.WriteLine("Ошибка при выполнении транзакции: " + ex.Message);  }  } |

Листинг 6 – Обновление статуса заказа, если дата доставки уже произошла

|  |
| --- |
| public void AddOrder(int ids, int payid, string address, DateTime date\_delivery)  {  string username24 = ListViewItemsForUser.userlog;  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  try  {  int userId;  using (SqlCommand selectCommand = new SqlCommand("SELECT USER\_ID FROM users WHERE login = @Username", connection))  {    selectCommand.Parameters.AddWithValue("@Username", username24);    userId = (int)selectCommand.ExecuteScalar();  }  using (SqlCommand command = new SqlCommand("AddOrder", connection))  {  command.CommandType = System.Data.CommandType.StoredProcedure;  ids = ids + 1;    command.Parameters.AddWithValue("@user\_id", userId);  command.Parameters.AddWithValue("@id\_sneacker", ids);  command.Parameters.AddWithValue("@payment\_method\_id", payid);  command.Parameters.AddWithValue("@user\_address", address);  command.Parameters.AddWithValue("@date\_delivery", date\_delivery);  // Execute the stored procedure  command.ExecuteNonQuery();  }  }  catch (Exception ex)  {  // Обработка ошибок и откат транзакции при необходимости  Console.WriteLine("Ошибка при выполнении " + ex.Message);  }  } } |

Листинг 7 – Оформление заказа пользователем

|  |
| --- |
| public void EditUserInfo(string login, string newlogin, string newpassword)  {  string connectionString = "Data Source=Pavel;Initial Catalog=Sneackers;Integrated Security=True";  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  SqlTransaction transaction = connection.BeginTransaction();  try  {  int userId;  using (SqlCommand selectCommand = new SqlCommand("SELECT USER\_ID FROM users WHERE login = @Username", connection))  {  selectCommand.Parameters.AddWithValue("@Username", login);  selectCommand.Transaction = transaction;  userId = (int)selectCommand.ExecuteScalar();  }  using (SqlCommand updateCommand = new SqlCommand("UPDATE users SET login = @NewLogin, password = @NewPassword WHERE user\_id = @UserId", connection))  {  updateCommand.Parameters.AddWithValue("@NewLogin", newlogin);  updateCommand.Parameters.AddWithValue("@NewPassword", newpassword);  updateCommand.Parameters.AddWithValue("@UserId", userId);  updateCommand.Transaction = transaction;  updateCommand.ExecuteNonQuery();  }  transaction.Commit();  }  catch (Exception ex)  {  // Обработка ошибок и откат транзакции при необходимости  transaction.Rollback();  Console.WriteLine("Ошибка при выполнении транзакции: " + ex.Message);  }  } } |

Листинг 8 – Обновление данных пользователя