МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Программирование мобильных систем»

Тема «Мобильное приложение магазина Acer»

Исполнитель

студентка 3 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. А. Юхневич

подпись, дата

Руководитель

асс. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. И. Уласевич

преподавателя

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. И. Уласевич

подпись дата инициалы и фамилия

**Содержание**

[Введение 3](#_Toc198054398)

[1 Аналитический обзор прототипов и литературных источников 4](#_Toc198054399)

[1.1 Обзор аналогичных решений 4](#_Toc198054400)

[1.1.1 Instacart 4](#_Toc198054401)

[1.1.2 Amazon Shopping 5](#_Toc198054402)

[1.2 Постановка задачи 6](#_Toc198054403)

[1.3 Вывод по разделу 6](#_Toc198054404)

[2 Анализ требований к программному средству и разработка функциональных требований 7](#_Toc198054405)

[2.1 Технические средства разработки 7](#_Toc198054406)

[2.2 Описание разрабатываемой функциональности приложения 8](#_Toc198054407)

[2.3 Спецификация функциональных требований 8](#_Toc198054408)

[2.4 Вывод по разделу 8](#_Toc198054409)

[3. Проектирования приложения 9](#_Toc198054410)

[3.1 Диаграмма вариантов использования 9](#_Toc198054411)

[3.2 Проектирование базы данных 10](#_Toc198054412)

[3.3 Описание информационных объектов и ограничений целостности 11](#_Toc198054413)

[3.3.1 Таблица Products 11](#_Toc198054414)

[3.3.2 Таблица Users 12](#_Toc198054415)

[3.3.3 Таблица UserCart 12](#_Toc198054416)

[3.3.4 Таблица MouseDetails 12](#_Toc198054417)

[3.3.5 Таблица LaptopDetails 13](#_Toc198054418)

[3.3.6 Таблица ChairDetails 13](#_Toc198054419)

[3.4 Анализ решений 14](#_Toc198054420)

[3.4.1 Выбор библиотек и технологий 14](#_Toc198054421)

[3.4.2 Выбор средств программирования 14](#_Toc198054422)

[3.5 Разработка API 15](#_Toc198054423)

[3.6 Выводы по разделу 16](#_Toc198054424)

[4. Реализация приложения 17](#_Toc198054425)

[4.1 Технические средства разработки 17](#_Toc198054426)

[4.2 Подключение базы данных 17](#_Toc198054427)

[4.3 Разработка мобильного приложения 19](#_Toc198054428)

[4.4 Выводы по разделу 21](#_Toc198054429)

[5. Анализ информационной безопасности приложения 23](#_Toc198054430)

[5.1 Защита пользовательских данных 23](#_Toc198054431)

[5.2 Вывод по разделу 23](#_Toc198054432)

[6. Тестирование приложения 24](#_Toc198054433)

[6.1 Тестирование клиентской области 24](#_Toc198054434)

[6.2 Вывод по разделу 26](#_Toc198054435)

[7. Руководство пользователя 27](#_Toc198054436)

[7.1 Пользователь 27](#_Toc198054437)

[7.2 Администратор 31](#_Toc198054438)

[7.3 Вывод по разделу 32](#_Toc198054439)

[Заключение 33](#_Toc198054440)

[Список литературы 34](#_Toc198054441)

[Приложение А. Диаграмма вариантов использования пользователя 35](#_Toc198054442)

[Приложение Б. Диаграмма вариантов использования администратора 36](#_Toc198054443)

[Приложение В. Диаграмма схемы базы данных 37](#_Toc198054444)

# **Введение**

Мобильные приложения стали неотъемлемой частью современного мира, особенно в сфере электронной коммерции, предоставляя пользователям быстрый и удобный доступ к товарам и услугам. В условиях стремительного развития цифровых технологий и повсеместного распространения смартфонов использование мобильных решений становится не просто удобным, а необходимым элементом успешного бизнеса. Электронная коммерция с каждым годом набирает всё большие обороты, и наличие функционального мобильного приложения становится ключевым фактором конкурентоспособности компании. Разработка мобильного приложения для продажи товаров позволяет потребителям легко находить, выбирать и приобретать продукцию непосредственно со своих мобильных устройств в любое время и в любом месте. Таким образом, мобильные платформы становятся эффективным средством взаимодействия между бизнесом и клиентами, обеспечивая высокий уровень сервиса и гибкость обслуживания. Такие приложения становятся важным инструментом цифровой трансформации торговли, позволяя компаниям оперативно реагировать на потребности клиентов и предлагать персонализированные предложения. Приложение упрощает процесс покупки благодаря интуитивно понятному каталогу, эффективному поиску, удобному управлению корзиной и быстрому оформлению заказа, а функции персонализации улучшают пользовательский опыт и повышают лояльность аудитории.

Целью данного курсового проекта является разработка мобильного приложения для электронной коммерции, предоставляющего пользователям удобный инструмент для поиска, выбора и покупки товаров. В рамках проекта будут исследованы и реализованы ключевые аспекты такого приложения, включая его функциональные возможности, методы взаимодействия с бэкенд-системой и базой данных, вопросы обработки ошибок и валидации данных, а также особенности проектирования пользовательского интерфейса. Также будет рассмотрен вопрос адаптивности интерфейса под различные устройства и операционные системы, что является важным аспектом при разработке современных мобильных решений.

Курсовой проект будет завершен выводами по каждому исследуемому разделу работы и общим заключением, обобщающим проделанную работу. Таким образом, проект направлен на всестороннее изучение процесса создания мобильного приложения для электронной коммерции и демонстрирует его значимость в условиях цифровой экономики. Создание мобильного приложения для электронной коммерции актуально и востребовано, так как оно значительно упрощает процесс онлайн-покупки для потребителей и предоставляет эффективный канал продаж для бизнеса в современном цифровом пространстве.

# **1 Аналитический обзор прототипов и литературных источников**

## **1.1 Обзор аналогичных решений**

В современном мире мобильные приложения для онлайн-покупок продуктов становятся все более востребованными. Их основная задача – обеспечить пользователям удобный доступ к ассортименту товаров, возможность выбора и оформления заказов прямо со смартфона. При разработке собственного приложения я обратил внимание на самые успешные примеры популярных сервисов для онлайн-покупок. Аналоги рассмотрены ниже.

### 1.1.1 Instacart

«Instacart»[1] – это приложение предоставляет пользователям возможность заказывать товары из ближайших супермаркетов с доставкой на дом или возможностью самовывоза. Особенности приложения включают удобный поиск товаров, персонализированные рекомендации, поддержку множества магазинов, а также функцию замены товаров в случае их отсутствия. Кроме того, Instacart предлагает различные способы оплаты и отслеживание заказа в режиме реального времени. Интерфейс приложения «Instacart» представлен на рисунке 1.1.

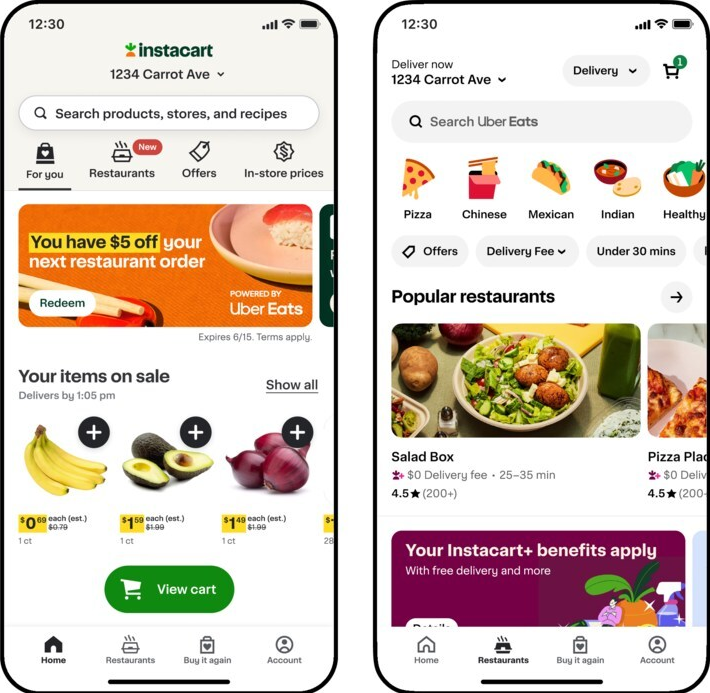


Рисунок 1.1 – Интерфейс приложения «Instacart»

Приложение «Instacart» предоставляет пользователю удобный интерфейс для просмотра каталогов продуктовых магазинов и выбора необходимых товаров. Оно содержит необходимые функции для эффективного управления процессом покупки: поиск товаров по категориям и названию, добавление позиций в корзину, выбор параметров доставки или самовывоза, а также возможность отслеживать статус выполнения заказа в реальном времени и повторять предыдущие покупки.

К недостаткам приложения «Instacart» можно отнести: потенциально перегруженный информацией интерфейс на некоторых экранах из-за обилия предложений и акционных баннеров, что может усложнять поиск; а также не всегда интуитивно понятный процесс взаимодействия с персональным сборщиком заказа, особенно при необходимости замены отсутствующих товаров.

### 1.1.2 Amazon Shopping

«Amazon Shopping»[2] – Мобильное приложение «Amazon Shopping» предоставляет пользователям удобный доступ к широкому ассортименту товаров, включая электронику, одежду, бытовую технику и многое другое. Функционал приложения охватывает поиск и фильтрацию товаров, просмотр детальных описаний и отзывов, а также возможность сравнения цен. Одной из ключевых особенностей является интеграция с голосовым помощником Alexa, что позволяет совершать покупки с помощью голосовых команд. Кроме того, приложение поддерживает уведомления о скидках и отслеживание доставки заказов в реальном времени, обеспечивая комфортный и эффективный процесс онлайн-шопинга. Интерфейс приложения показан на рисунке 1.2.

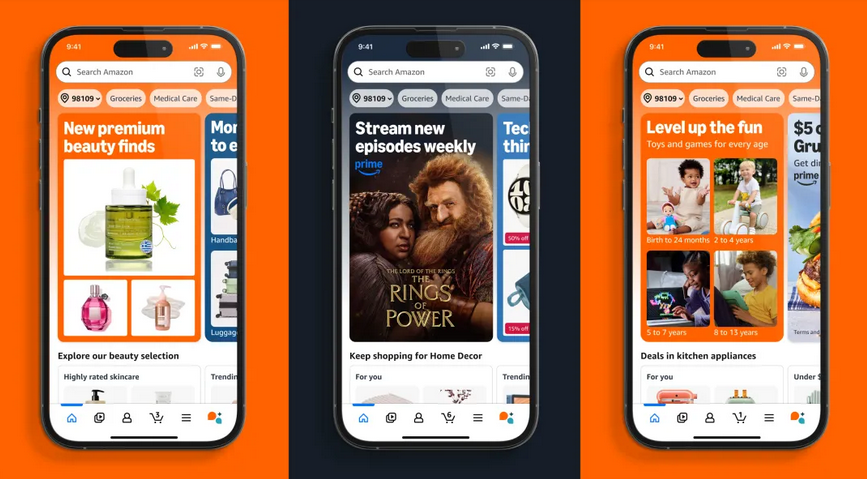


Рисунок 1.2 – Интерфейс приложения «Amazon Shopping»

Приложение «Amazon Shopping» предоставляет удобный доступ к огромному каталогу товаров и упрощает процесс поиска, выбора и оформления покупок с мобильного устройства.

К недостаткам можно отнести потенциально перегруженный интерфейс и сложности с быстрой оценкой подлинности товаров или достоверности отзывов, так же трудности возникают при покупке товаров.

### 1.2 Постановка задачи

Основываясь на анализе аналогичных решений из пункта 1.1, проектируемое программное средство для покупки товаров должно иметь:

1. Понятный интерфейс: пользователи должны иметь доступ к интуитивно понятному интерфейсу, который поможет с выбором товара.
2. Гибкость и настраиваемость: пользователи должны иметь возможность управлять корзиной.
3. Поиск товаров: пользователи должны иметь возможность масштабируемого поиска.
4. Авторизация пользователей: пользователи должны иметь возможность различного вида регистрации/авторизации.
5. Управление товарами: администраторы должны иметь возможность управления товарами.

# **1.3 Вывод по разделу**

В данной главе был проведен анализ существующих программных средств для онлайн-покупок, таких как «Instacart» и «Amazon Shopping», выявлены их достоинства и недостатки, были сформулированы основные требования к проектируемому программному средству. Разработка нового программного средства для покупки товаров должна основываться на положительном опыте существующих решений и учитывать их недостатки. Применение современных технологий и подходов в разработке поможет улучшить функциональность и удобство использования. Таким образом, разработка нового программного средства должна не только восполнять пробелы, выявленные в существующих решениях, но и предлагать функции, которые сделают процесс онлайн-покупки более удобным и надежным. Все, описанное в данной главе, важно для создания высококачественного продукта, способного удовлетворить актуальные запросы пользователей и соответствовать современным стандартам UX/UI-дизайна.

Особое внимание при проектировании должно уделяться аспектам безопасности, стабильности работы приложения и его масштабируемости. Также важно обеспечить гибкость системы для возможного расширения функционала в будущем с учетом изменяющихся требований рынка и поведения пользователей. Анализ существующих решений и выявление их слабых сторон позволяют заложить прочную основу для разработки конкурентоспособного и востребованного программного продукта.

# **2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований является фундаментальным этапом в процессе создания программного обеспечения. Он охватывает сбор, упорядочивание, документирование, глубокое изучение и выявление любых несоответствий или неполноты в потребностях системы, а также разрешение возникающих в ходе этого процесса разногласий.

Ключевая цель анализа требований в рамках любого проекта заключается в получении максимально полной информации о заказчике и специфике его деятельности, точном определении масштаба и границ проекта, а также в оценке потенциальных рисков. На этой фазе происходит определение ключевых методологических и технологических требований, формулируются конкретные цели и задачи проекта, а также устанавливаются основные критерии успеха, которые будут служить ориентирами при оценке результатов внедрения. Точное формулирование и документирование требований на начальном этапе во многом предопределяют успешность всего проекта, оказывая прямое влияние на все последующие стадии разработки.

## **Технические средства разработки**

Для создания мобильного приложения магазина «Acer» были выбраны передовые технологии и инструменты, чтобы обеспечить высокое качество и удобство использования.

В процессе разработки был использован язык программирования C#, а в качестве основного фреймворка — .NET MAUI. [4]

.NET MAUI (Multi-platform App UI) – это кроссплатформенный фреймворк с открытым исходным кодом, разработанный и поддерживаемый корпорацией Microsoft. Он позволяет создавать современные мобильные и настольные приложения из единой кодовой базы C# и XAML. .NET MAUI упрощает разработку пользовательского интерфейса, который автоматически адаптируется к различным форм-факторам и операционным системам, включая iOS, Android, Windows и macOS.

Для обеспечения надежного и стабильного взаимодействия мобильного приложения с данными, а также реализации основной бизнес-логики, был разработан серверный программный интерфейс (API). Этот API играет ключевую роль в архитектуре всего программного комплекса, выступая в качестве промежуточного звена между клиентской частью — мобильным приложением, разработанным с использованием .NET MAUI, — и серверной частью, включающей базу данных. Данный интерфейс был создан с применением современной и производительной технологии ASP.NET Core. API обрабатывает все входящие запросы от мобильного клиента, выполняет необходимую бизнес-логику и обеспечивает доступ к базе данных PostgreSQL, где хранятся структурированные сведения о товарах, пользователях, заказах и других элементах системы.

## **Описание разрабатываемой функциональности приложения**

Программное средство предоставляет пользователю следующие функциональные возможности:

* просмотр товаров;
* сортировка товаров;
* управление информацией пользователя;
* добавление товаров в корзину;
* покупка товаров;
* управление товаров администраторами;
* авторизация через Google;
* поиск товаров.

## **Спецификация функциональных требований**

Спецификация функциональных требований определяет набор конкретных функций, операций и процессов, которые разрабатываемое программное обеспечение должно выполнять для соответствия потребностям пользователей и достижению целей проекта. Для мобильного приложения магазина «Acer» эти требования детализируют взаимодействие пользователя с системой, управление данными и реализацию бизнес-логики.

Для функциональности приложения необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В приложении магазина «Acer» при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для регистрации пользователя необходимо указать почту пользователя и пароль. Для авторизации входными параметрами являются почта и пароль пользователя, которые содержатся в базе данных или же вход через аккаунт Google.

## **Вывод по разделу**

В данной главе была проведена детальная аналитика требований к проектируемому мобильному приложению магазина «Acer». Анализ требований является ключевым этапом разработки, так как он формирует четкие рамки проекта.

Таким образом, сформулированные требования и спецификации в данной главе, а также описание выбранных технических средств для разработки, создают прочную основу для дальнейшей разработки программного средства. Разработка данного приложения направлена на создание удобной платформы для онлайн-покупок, которая позволит пользователям легко находить, выбирать и приобретать товары.

# **3. Проектирования приложения**

## **3.1 Диаграмма вариантов использования**

Проектирование является фундаментальной стадией в цикле разработки программного обеспечения, важность которой, к сожалению, часто недооценивается начинающими специалистами. Часто новички в этой области предпочитают удерживать все детали в уме или фиксировать лишь ключевые аспекты в разрозненных записях. Такой подход приводит к отсутствию четкого, структурированного плана действий, что может стать причиной значительных задержек в реализации проекта.

Как правило, процесс проектирования включает визуальное представление будущей системы в виде диаграмм, поскольку графический формат значительно облегчает восприятие сложной информации. Вместо создания объемных текстовых описаний каждой отдельной функции, разработчики используют различные типы диаграмм для наглядного моделирования архитектуры и поведения системы. Это помогает систематизировать информацию, облегчает запоминание ключевых аспектов реализации и позволяет быстро вводить новых участников команды в контекст проекта.

Диаграмма вариантов использования (use-case diagram) – это тип диаграммы, предназначенный для описания функциональности программной системы с точки зрения взаимодействия с ней различных групп пользователей или внешних систем.

Суть построения такой диаграммы заключается в представлении проектируемой системы как совокупности взаимодействий между "актерами" и "вариантами использования". Актером называется любая внешняя сущность (человек, другое приложение, оборудование), которая взаимодействует с моделируемой системой. Вариант использования, в свою очередь, представляет собой спецификацию сервиса или функции, предоставляемой системой актеру, описывая определенный набор действий или сценариев, которые система выполняет в ответ на запросы актера, без детализации внутренней реализации этих действий.

Перед тем как приступить к непосредственному написанию кода, необходимо определить все роли пользователей и внешних систем, их задачи и все возможные сценарии их взаимодействия с программным средством. Для наглядного представления и фиксации этих аспектов строится диаграмма вариантов использования.

Диаграмма вариантов использования для пользователей представлена в приложениях А и Б, а также на рисунке 3.1.

Например, для зарегистрированного пользователя магазина доступен ряд функциональных возможностей. После успешной авторизации он может просматривать каталог товаров, искать и сортировать продукцию, добавлять выбранные позиции в корзину и оформлять заказ. Администратор системы может осуществлять управление ассортиментом товаров (добавлять новые, редактировать существующие, удалять неактуальные). Наличие диаграмм вариантов использования на раннем этапе разработки повышает эффективность коммуникации между участниками проекта и снижает риск недопонимания в процессе реализации.



Рисунок 3.1 – Диаграмма вариантов использования

Таким образом были определены границы функциональных возможностей разрабатываемого приложения.

# **3.2 Проектирование базы данных**

Модель данных можно определить как абстрактное и логическое описание структуры информации. Она включает совокупность объектов, правил работы с ними (операторов) и их связей, представляя собой концептуальную основу для доступа пользователей к данным. Модель данных является центральным элементом в процессе проектирования базы данных, так как от нее зависит способ организации хранения информации и методы ее обработки.

Для реализации модели данных и обеспечения хранения основной информации приложения (товары, заказы и т.д.) была выбрана реляционная база данных PostgreSQL. При этом для управления процессом авторизации и регистрации пользователей был использован сервис Firebase Authentication. Связь мобильного приложения на .NET MAUI с базой данных PostgreSQL, а также реализация основной бизнес-логики, осуществлялась через разработанный серверный API на базе ASP.NET Core, детальное описание которого представлено в пункте 3.5. Логическая модель базы данных PostgreSQL представлена на рисунке 3.2.

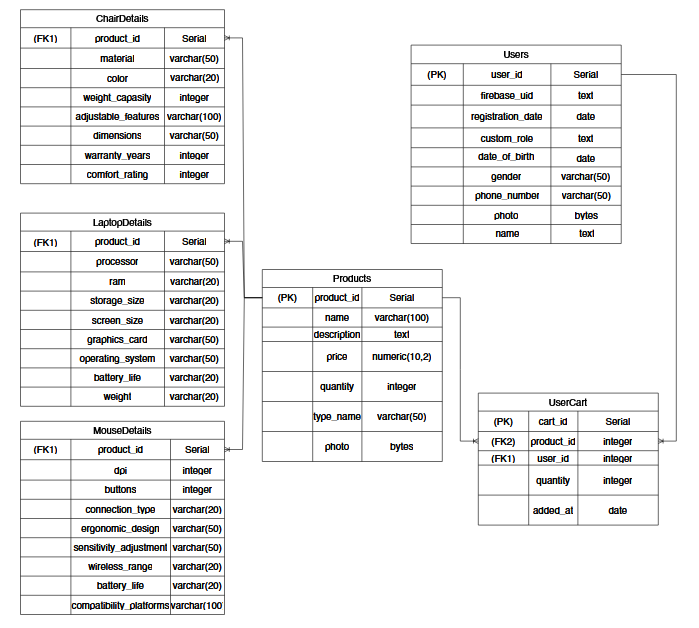


Рисунок 3.2 – Модель БД

Созданная база данных содержит в себе 11 таблиц: Products, ChairDetails, LaptopDetails, MouseDetails, Users, UserCart.

## **3.3** **Описание информационных объектов и ограничений целостности**

## **3.3.1 Таблица Products**

Таблица «Products» будет содержать информацию о товарах. Структура представлена в таблице 3.1.

Таблица 3.1 – Описание таблицы Products

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| product\_id | integer | Уникальный идентификатор товара (первичный ключ) |
| name | varchar(50) | Название товара |
| description | text | Описание товара |
| price | numeric(10,2) | Цена товара |
| quantity | integer | Количество товара на складе |
| type\_name | varchar(50) | Тип товара |
| photo | bytes | Фотография товара |

## **3.3.2 Таблица Users**

Таблица «Users» будет содержать информацию о пользователях. Структура представлена в таблице 3.2.

Таблица 3.2 – Описание таблицы Users

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| user\_id | integer | Уникальный идентификатор пользователя (первичный ключ) |
| uirebase\_uid | text | Уникальный идентификатор пользователя(Firebase) |
| registration\_date | date | Дата регистрации пользователя |
| custom\_role | text | Роль пользователя |
| date\_of\_birth | date | Дата рождения пользователя |
| gender | varchar(50) | Гендер пользователя |
| phone\_number | varchar(50) | Телефон пользователя |
| photo | bytes | Фотография профиля |
| name | text | Имя пользователя |

## **3.3.3 Таблица UserCart**

Таблица «UserCart» будет содержать информацию о корзине пользователей. Структура представлена в таблице 3.3.

Таблица 3.3 – Описание таблицы UserCart

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| cart\_id | integer | Уникальный идентификатор корзины (первичный ключ) |
| user\_id | integer | ID пользователя (внешний ключ к Users.user\_id) |
| product\_id | integer | ID товара (внешний ключ к Products.product\_id) |
| quantity | integer | Количество товара в корзине пользователя |
| added\_at | date | Время добавления товара в корзину |

## **3.3.4 Таблица MouseDetails**

Таблица «MouseDetails» будет содержать информацию о характеристиках мыши. Таблицы деталей в основе первичного ключа будут иметь такой же id, как и товар, к которому относится описание. Структура представлена в таблице 3.4.

Таблица 3.4 – Описание таблицы MouseDetails

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| product\_id | integer | Уникальный идентификатор продукта(первичный ключ) |
| dpi | integer | Количество точек на шкалный дюйм |
| buttons | integer | Количество кнопок |
| connection\_type | varchar(20) | Тип связи с устройством |
| ergonomic design | varchar(50) | Эргономичный дизайн |
| sensitivity\_adjustment | varchar(50) | Возможность настройки скорости движения по экрану |
| wireless\_range | varchar(20) | Максимальное расстояние связи беспроводной мыши |
| compatibility\_platforms | varchar(100) | Поддерживаемые платформы |

## **3.3.5 Таблица LaptopDetails**

Таблица «LaptopDetails» будет содержать информацию о характеристиках ноутбука. Структура представлена в таблице 3.5.

Таблица 3.5 – Описание таблицы LaptopDetails

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| product\_id | integer | Уникальный идентификатор продукта(первичный ключ) |
| processor | varchar(50) | Тип процессора |
| ram | varchar(20) | Количество и тип оперативной памяти |
| storage\_size | varchar(20) | Вместимость накопителя |
| screen\_size | varchar(30) | Количество дюймов экрана |
| graphics\_card | varchar(50) | Тип видеокарты |
| operating\_system | varchar(50) | Тип ОС |
| buttery\_life | varchar(20) | Количество часов работы ноутбука без подключения к сети |
| weight | varchar(20) | Вес ноутбука |

## **3.3.6 Таблица ChairDetails**

Таблица «ChairDetails» будет содержать информацию о характеристиках стула. Структура представлена в таблице 3.6.

Таблица 3.6 – Описание таблицы ChairDetails

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| product\_id | integer | Уникальный идентификатор продукта (первичный ключ) |
| material | varchar(50) | Тип материала кресла |
| color | varchar(50) | Цвет кресла |
| weight\_capacity | integer | Весовая нагрузка кресла |
| adjustable\_features | varchar(100) | Регулируемые функции |
| dimensions | varchar(50) | Габариты кресла |
| warranty\_years | integer | Гарантийный срок кресла |
| comfort\_rating | integer | Рейтинг комфорта. |

## **3.4 Анализ решений**

Выбор языка программирования является критически важным этапом разработки ПО, поскольку он определяет скорость реализации функционала, удобство тестирования и отладки, возможность кроссплатформенной поддержки, а также гибкость при дальнейшем масштабировании и модификации системы, при этом не существует универсального решения — каждый язык представляет собой специализированный инструмент с уникальным набором сильных и слабых сторон для разных задач.

## **3.4.1 Выбор библиотек и технологий**

Для разработки кроссплатформенного приложения был выбран .NET MAUI (Multi-platform App UI) — современная платформа от Microsoft, позволяющая создавать нативные приложения для Android, iOS, Windows и macOS из единой кодовой базы на языке C#. Будучи преемником Xamarin. Forms, MAUI сочетает проверенную экосистему .NET с современными подходами к кроссплатформенной разработке, обеспечивая полный доступ к нативным API и высокую производительность.

Размеры собранных приложений в MAUI сравнительно компактны: базовое Android-приложение занимает около 15-20 МБ, iOS-версии обычно немного меньше. В отличие от Flutter, использующего язык Dart, MAUI опирается на популярный в корпоративной среде C#, что упрощает интеграцию с существующими .NET-библиотеками и сервисами. Особенно MAUI выделяется при разработке для десктопных платформ, предлагая более глубокую интеграцию с Windows и macOS по сравнению с другими кроссплатформенными решениями.

## **3.4.2 Выбор средств программирования**

Для разработки приложения была выбрана интегрированная среда разработки Visual Studio 2022 — современная и мощная IDE от Microsoft, предоставляющая полный набор инструментов для создания кроссплатформенных решений. Эта среда отличается высокой производительностью, умным авто дополнением кода, встроенными средствами отладки и богатыми возможностями для работы с .NET MAUI-проектами. Visual Studio 2022 поддерживает все этапы разработки — от проектирования интерфейсов до тестирования и публикации приложений на различных платформах.

В качестве серверной части системы использовалась PostgreSQL — надежная и производительная реляционная база данных с открытым исходным кодом. Для реализации механизмов аутентификации и авторизации пользователей был интегрирован Firebase Authentication — безопасное и масштабируемое решение от Google, поддерживающее различные методы входа (по email, через соцсети и др.). Такое сочетание технологий обеспечило стабильную работу backend-части приложения, безопасное хранение данных и удобный процесс идентификации пользователей. Выбор средств напрямую зависит от желаний разработчика и умения использовать ту или иную платформу.

## **3.5 Разработка API**

Для курсового проекта я разработал сервер на ASP.NET Core, который обеспечивает связь между мобильным приложением и PostgreSQL. Сервер обрабатывает запросы от клиента, выполняет бизнес-логику и взаимодействует с базой данных посылая обработанные ответы обратно пользователю.

ASP.NET Core был выбран из-за высокой производительности, встроенной поддержки асинхронности и удобной работы с базами данных через Entity Framework Core. Этот стек технологий обеспечил стабильную работу API, безопасность данных и простую масштабируемость проекта. Структура API представлена на рисунке 3.3.

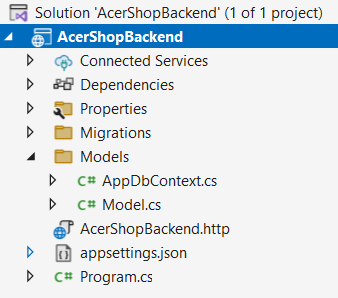


Рисунок 3.3 – Структура API

Серверная часть приложения разработана на ASP.NET Core и обеспечивает взаимодействие между клиентским приложением и базой данных PostgreSQL. Основой структуры данных служат модели Entity Framework Core, которые не только определяют схему базы данных, но и автоматически создают соответствующие таблицы через механизм миграций.

Все API-эндпоинты реализованы в едином контроллере, что обеспечивает централизованное управление бизнес-логикой приложения. Класс AppDbContext выступает в качестве моста между объектной моделью приложения и реляционной базой данных, предоставляя удобные методы для выполнения CRUD-операций.

Для аутентификации пользователей используется встроенная система Identity Framework, интегрированная с Firebase. Такая архитектура обеспечивает надежное хранение данных в PostgreSQL и безопасную авторизацию через проверенное временем решение от Google.

Кроме реализации базовой логики обработки запросов, большое внимание было уделено организации слоев ответственности внутри серверного приложения. В частности, для повышения читаемости и упрощения сопровождения кода была реализована модель слоистой архитектуры, включающая уровни контроллеров, сервисов и репозиториев. Такой подход позволяет разделить бизнес-логику и доступ к данным, что делает систему более гибкой и масштабируемой при дальнейшем развитии проекта. В будущем это упростит добавление новых функций без риска нарушить существующий функционал.

Особое внимание было уделено обеспечению безопасности при работе с пользовательскими данными. Помимо использования Identity Framework и интеграции с Firebase для аутентификации, в API реализована валидация входных данных, обработка исключений и защита от типичных угроз, таких как SQL-инъекции и межсайтовый скриптинг (XSS). Эти меры позволяют значительно повысить уровень надежности системы и доверие со стороны конечных пользователей, особенно в контексте работы с персональными и платёжными данными.

## **3.6 Выводы по разделу**

В ходе проектирования приложения была разработана комплексная архитектура, включающая клиентскую часть на .NET MAUI, серверное API на ASP.NET Core и базу данных PostgreSQL. Основное внимание уделено созданию диаграммы вариантов использования, отображающей взаимодействие пользователей с системой, и проектированию логической модели базы данных из 6 таблиц. Для хранения файлов и аутентификации пользователей интегрирован сервис Firebase.

Серверная часть реализована как RESTful API на ASP.NET Core с единым контроллером, содержащим все необходимые эндпоинты для работы с данными. Использование Entity Framework Core позволяет автоматизировать создание и обновление структуры БД через механизм миграций. Для аутентификации применена связка Firebase Authentication с Identity Framework, обеспечивающая безопасный доступ к функционалу.

Выбранный технологический стек (.NET MAUI, ASP.NET Core, PostgreSQL, Firebase) обеспечивает оптимальное сочетание производительности, безопасности и удобства разработки. Архитектура системы предусматривает четкое разделение ответственности между компонентами и возможность дальнейшего масштабирования функционала при необходимости. Все решения приняты с учетом требований пользователей и особенностей разрабатываемого приложения.

# **4. Реализация приложения**

## **4.1 Технические средства разработки**

Для разработки приложения был выбран технологический стек на основе .NET. Основным языком стал C#, обеспечивающий высокую производительность и надежность кода. Кроссплатформенный интерфейс реализован с помощью .NET MAUI, что позволило создать единую кодовую базу для всех поддерживаемых платформ.

Серверная часть построена на ASP.NET Core, обеспечивающем быструю и безопасную работу API. Для хранения данных используется PostgreSQL - надежная реляционная СУБД с открытым исходным кодом. Аутентификация пользователей реализована через Firebase, что гарантирует безопасность и простоту интеграции.

В качестве основной среды разработки применялась Visual Studio 2022. Ее богатый функционал и глубокая интеграция с .NET экосистемой значительно ускорили процесс создания приложения.

## **4.2 Подключение базы данных**

Для работы с базой данных Firestore нашего проекта необходимо определить автоматически сгенерированный файл google-services.json, основные параметры которого представлены на листинге 4.1.[3]

{

"project\_info": {

"project\_number": "223019245263",

"project\_id": "mauiacershop",

"storage\_bucket": "mauiacershop.firebasestorage.app"

},

"client": [

{

"client\_info": {

"mobilesdk\_app\_id": "1:223019245263:android:7d5bbcca683a5eafaa21ba",

"android\_client\_info": {

"package\_name": "com.companyname.acershop"

}

},

"oauth\_client": [

{

"client\_id": "223019245263-co0nq6db805amiigrsi7fmf6fcef7u0p.apps.googleusercontent.com",

"client\_type": 1,

"android\_info": {

"package\_name": "com.companyname.acershop",

"certificate\_hash": "6173dbd063495e28db7b4c8988870eede01029a7"

}

}}}};

}

},

{

"client\_id": "223019245263-blsgcj84m2g88hs492ac8o9cffkkf1g4.apps.googleusercontent.com",

"client\_type": 3

}

],

"api\_key": [

{

"current\_key": "AIzaSyARBEiPM32MB1d8x3U\_1JrjTbyNmM6pKao"

}

],

"services": {

"appinvite\_service": {

"other\_platform\_oauth\_client": [

{

"client\_id": "223019245263-blsgcj84m2g88hs492ac8o9cffkkf1g4.apps.googleusercontent.com",

"client\_type": 3

}

]

}

}

}

],

"configuration\_version": "1"

}

Листинг 4.1 – Параметры google-services.json

Перед добавлением вышенаписанного кода в проект необходимо добавить пакет зависимостей «FirebaseAuthentication.net». А после определить сервис, представленный в листинге 4.2.

using Android.App;

using Android.Content;

using Android.Gms.Auth.Api;

using Android.Gms.Auth.Api.SignIn;

using Android.Gms.Common;

using Android.Gms.Tasks;

using Java.Lang;

using System;

using System.Threading.Tasks;

namespace AcerShop.Platforms.Android

{public class GoogleAuthService : Java.Lang.Object, IOnCompleteListener

{ public static GoogleAuthService Instance { get; } = new GoogleAuthService();

private TaskCompletionSource<string> \_tcs;

public Task<string> SignInAsync()

{

\_tcs = new TaskCompletionSource<string>();

var gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DefaultSignIn)

.RequestIdToken("223019245263-blsgcj84m2g88hs492ac8o9cffkkf1g4.apps.googleusercontent.com")

.RequestEmail()

.Build();

var client = GoogleSignIn.GetClient(Platform.CurrentActivity, gso);

var signInIntent = client.SignInIntent;

Platform.CurrentActivity.StartActivityForResult(signInIntent, 9001);

return \_tcs.Task;

}

public void OnComplete(global::Android.Gms.Tasks.Task task)

{if (task.IsSuccessful){

var account = (GoogleSignInAccount)task.GetResult(Java.Lang.Class.FromType(typeof(GoogleSignInAccount)));

\_tcs?.TrySetResult(account?.IdToken);}

else{\_tcs?.TrySetException(new Java.Lang.Exception(task.Exception?.Message ?? "Google Sign-in failed"));

}

}

}

}

Листинг 4.2 – Библиотеки в файле pubspec.yaml

При помощи данной зависимости и кода, написанного выше, была подключения база данных Firebase для разрабатываемого приложения.

## **4.3 Разработка мобильного приложения**

Мобильное приложение AcerShop организовано с учетом принципов модульности и паттерна MVVM (Model-View-ViewModel), что обеспечивает четкое разделение ответственности между компонентами. В проекте выделены ключевые папки, такие как Model, View, ViewModel, Services и другие, каждая из которых играет важную роль в архитектуре приложения.

Папка Properties содержит настройки и конфигурации приложения, включая параметры сборки и метаданные необходимые для корректной компиляции и запуска программы.

Controls предназначена для пользовательских элементов управления, таких как PasswordMeter.xaml, который обеспечивает визуальную индикацию сложности пароля, а также связи самого элемента управления с логикой.

Converters включает конвертеры данных, используемые для преобразования значений в XAML-привязках, что упрощает отображение данных в интерфейсе. Например, InCartToEnabledConverter.cs позволяет нам отслеживать товары, которые находятся в корзине и в случае выполнения данного условия блокирует возможность повторного нажатия на элемент управления.

В папке Model расположены классы, описывающие структуру данных приложения. Например, PasswordValidationModels.cs содержит модели для валидации паролей, Products.cs определяет структуру товаров, а User.cs представляет данные пользователя.

Platforms включает платформа-специфичный код для различных операционных систем, что обеспечивает корректную работу приложения на разных устройствах.

Сервисный слой, находящийся в папке **Services**, реализует ключевую логику взаимодействия с внешними системами. ApiService.cs отвечает за отправку и получение данных через API, а PasswordValidationService.cs предоставляет методы для проверки надежности паролей.

Папка **View** содержит XAML-страницы, такие как LoginPage.xaml, RegisterPage.xaml, MainPage.xaml и другие, которые формируют пользовательский интерфейс приложения. Каждая страница связана с соответствующей ViewModel из папки **ViewModel**, где реализована логика взаимодействия с данными. Например, LoginPageViewModel.cs управляет аутентификацией пользователей, а CartPageViewModel.cs обрабатывает данные корзины покупок.

Корневой файл проекта, такой как App.xaml отвечает за настройку ресурсов всего приложения в целом, в нашем случае он служит для определения стилей, а также основных цветов, которые можно получить непосредственно во всем приложения без повторного определения. Сами же данные хранятся в папке ресурсов Resources, мы можем без всяких трудностей переопределить базовые цвета и стили для конкретных элементов управления, что очень удобно для разработчиков.

Структура проекта представлена на рисунке 4.1.

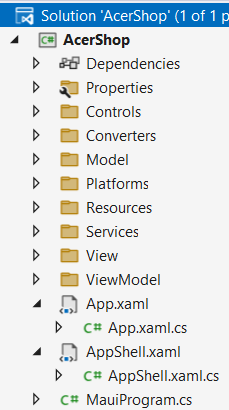


Рисунок 4.1 – Структура проекта

Инициализация приложения AcerShop начинается в классе MauiProgram, который служит точкой входа для настройки зависимостей и сервисов. Здесь используется механизм внедрения зависимостей (Dependency Injection), что позволяет гибко управлять компонентами приложения и обеспечивает модульность архитектуры.

Основная конфигурация выполняется в методе CreateMauiApp(), где задаются глобальные параметры, такие как шрифты и сторонние библиотеки.

Для создания современного и функционального интерфейса приложение подключает несколько внешних библиотек. CommunityToolkit.Maui расширяет стандартные возможности .NET MAUI, добавляя полезные компоненты и утилиты. UraniumUI и UraniumUIMaterial используются для стилизации интерфейса в соответствии с принципами Material Design, что улучшает визуальное восприятие и удобство взаимодействия с приложением.

Важной частью настройки является регистрация сервисов и страниц. Сервисы, такие как ApiService и PasswordValidationService, добавляются как Singleton, что гарантирует их единственный экземпляр на протяжении всей работы приложения. Страницы и их ViewModel регистрируются как Transient, обеспечивая создание новых экземпляров при каждом обращении. Это позволяет эффективно управлять состоянием приложения и избегать нежелательных побочных эффектов.

Класс MauiProgram представлен в листинге 4.3.

public static class MauiProgram{

public static MauiApp CreateMauiApp()

{

var builder = MauiApp.CreateBuilder();

builder

.UseMauiApp<App>()

.ConfigureFonts(fonts =>

{

fonts.AddFont("MaterialIcons-Regular.ttf", "Material");

fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");

fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");

fonts.AddFont("FontAwesome.ttf", "FontAwesome");

fonts.AddFontAwesomeIconFonts();

})

.UseUraniumUI()

.UseUraniumUIMaterial()

.UseMauiCommunityToolkit();

builder.Services.AddSingleton<ApiService>();

builder.Services.AddSingleton<PasswordValidationService>();

builder.Services.AddTransient<LoginPage>();

builder.Services.AddTransient<LoginPageViewModel>();

builder.Services.AddTransient<ProfilePage>();

builder.Services.AddTransient<ProfilePageViewModel>();

builder.Services.AddTransient<RegisterViewModel>();

builder.Services.AddTransient<RegisterPage>();

builder.Services.AddTransient<MainPage>();

builder.Services.AddTransient<MainPageViewModel>();

builder.Services.AddTransient<EditProductPage>();

builder.Services.AddTransient<EditProductViewModel>();

builder.Services.AddTransient<AdminProductsPage>();

builder.Services.AddTransient<AdminProductsViewModel>();

builder.Services.AddTransient<ProductDetailsPage>();

builder.Services.AddTransient<ProductDetailsViewModel>();

builder.Services.AddTransient<CartPage>();

builder.Services.AddTransient<CartPageViewModel>();

}

}

Листинг 4.3 – Параметры google-services.json

Таким образом мы регистрируем все классы нашего приложения в главном файле для корректной работы.

## **4.4 Выводы по разделу**

В данном разделе подробно описан поэтапный процесс создания приложения, начиная от выбора технологического стека и заканчивая финальной реализацией. На начальном этапе был проведен тщательный анализ требований к проекту, что позволило подобрать оптимальные инструменты разработки. Выбор .NET MAUI в качестве основной платформы обеспечил кроссплатформенную совместимость, а использование C# гарантировало высокую производительность и надежность кода. Интеграция с Firebase значительно упростила реализацию аутентификации и хранения данных, обеспечив при этом высокий уровень безопасности.

Особое внимание было уделено проектированию архитектуры приложения. Применение паттерна MVVM позволило четко разделить логику приложения на три основных слоя: Model, View и ViewModel. Это не только улучшило читаемость кода, но и упростило его тестирование и дальнейшую поддержку. Сервисный слой, включающий такие компоненты, как ApiService и PasswordValidationService, обеспечил гибкость и масштабируемость приложения. Кроме того, использование Dependency Injection (DI) для управления зависимостями сделало код более модульным и удобным для расширения.

Важным этапом разработки стала настройка пользовательского интерфейса. Подключение библиотек CommunityToolkit.Maui, UraniumUI и UraniumUIMaterial позволило создать современный и интуитивно понятный интерфейс, соответствующий принципам Material Design. Регистрация страниц и сервисов в классе MauiProgram обеспечила корректную работу всех компонентов приложения, а также их эффективное взаимодействие.

В результате проделанной работы было создано полноценное мобильное приложение, охватывающее все аспекты разработки – от серверной части до клиентских интерфейсов. Описанные в разделе решения демонстрируют комплексный подход к проектированию и реализации программного обеспечения. Приложение обладает четкой структурой, модульностью и высокой производительностью, что делает его удобным для дальнейшего развития и масштабирования.

Таким образом, все поставленные задачи были успешно выполнены. Приложение соответствует современным стандартам разработки, обладает надежной архитектурой и предоставляет пользователям удобный и функциональный интерфейс. Описанные в разделе методы и технологии могут быть применены и в других проектах, что подтверждает их универсальность и эффективность.

# **5. Анализ информационной безопасности приложения**

При разработке приложения особое внимание уделено информационной безопасности: реализовано строгое разграничение прав доступа между обычными пользователями и администраторами, применена двухфакторная аутентификация через Firebase с шифрованием паролей, а также внедрена система ролевого контроля доступа к данным с обязательной проверкой прав на уровне серверных API.

## **5.1 Защита пользовательских данных**

Firebase обеспечивает комплексную систему безопасности, включая надежную аутентификацию через SCRYPT-хеширование паролей с уникальной солью и поддержку OAuth-провайдеров, что исключает хранение учетных данных в приложении. Все данные защищены сквозным TLS/SSL-шифрованием при передаче и автоматическим шифрованием в Firestore/Realtime Database, с детальным контролем доступа через правила безопасности на уровне документов. Платформа соответствует международным стандартам ISO 27001 и GDPR, предлагает встроенный аудит операций и ежедневные резервные копии, а для особых случаев поддерживает дополнительное клиентское шифрование данных перед их отправкой в облако.

## **5.2 Вывод по разделу**

Таким образом, система безопасности приложения построена на двух проверенных решениях: Firebase Authentication обеспечивает надежную защиту пользовательских данных с помощью современного хеширования и OAuth-интеграции, в то время как Supabase дополняет эту систему безопасным хранением информации с применением JWT-токенов и строгим контролем доступа через RLS-политики. Оба сервиса соответствуют международным стандартам безопасности и предоставляют комплексные механизмы защиты данных - от шифрования передаваемой информации до детального аудита операций, что в совокупности создает многоуровневую систему защиты для всего приложения.

Перспективным направлением развития может стать внедрение адаптивной аутентификации, которая будет анализировать поведение пользователей и контекст доступа, автоматически повышая требования к проверке подлинности при обнаружении подозрительной активности. Дополнительным уровнем защиты могло бы стать регулярное автоматическое тестирование уязвимостей с интеграцией в CI/CD-конвейер, что позволит оперативно выявлять и устранять потенциальные угрозы на ранних этапах разработки.

# **6. Тестирование приложения**

## **6.1 Тестирование клиентской области**

Тестирование мобильных приложений играет ключевую роль в обеспечении стабильной работы и положительного пользовательского опыта. Оно позволяет выявить и устранить потенциальные ошибки до релиза, что минимизирует риск сбоев в работе приложения и способствует его коммерческому успеху.

В рамках тестирования была проверена работа страницы авторизации. Как показано на рисунке 6.1, система корректно обрабатывает различные сценарии: отказ в доступе при вводе неверных данных, предупреждение о незаполненных полях и блокировку входа для незарегистрированных пользователей. Это обеспечивает защищенный и удобный процесс аутентификации для конечных пользователей.

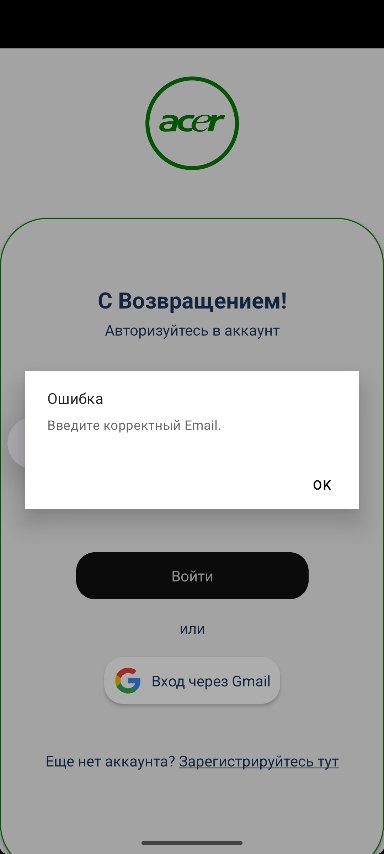
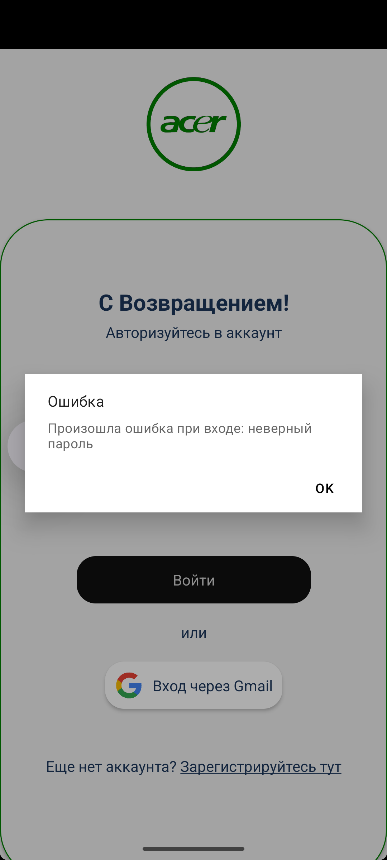
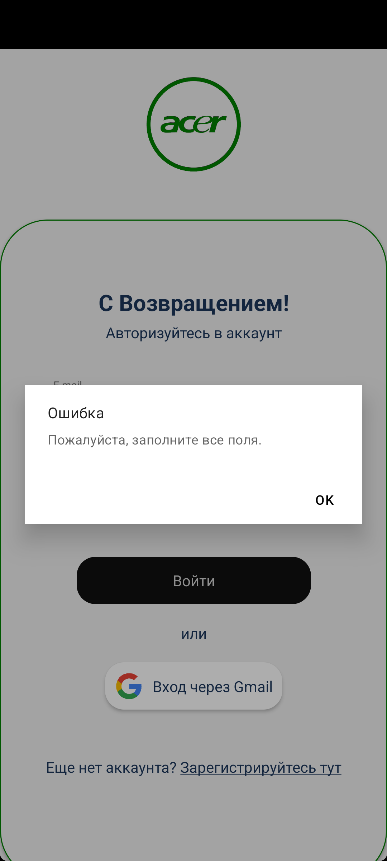


Рисунок 6.1 – Тестирование авторизации

Таким образом в случае введение пользователем некорректных данных приложение без каких-либо проблем обработает данные ситуации, в противном случае такого рода проблемы вызывали бы необработанные ошибки, которые в конечном итоге завершали был работу приложения в экстренном режиме, другими словами «крашило» приложение.

Далее протестируем страницу регистрации. Пользователь может зарегистрироваться с уникальной почтой. Роль пользователя автоматически считается User. Кроме того, есть проверка на корректный email и пароль. Тестирование продемонстрировано на рисунке 4.2.

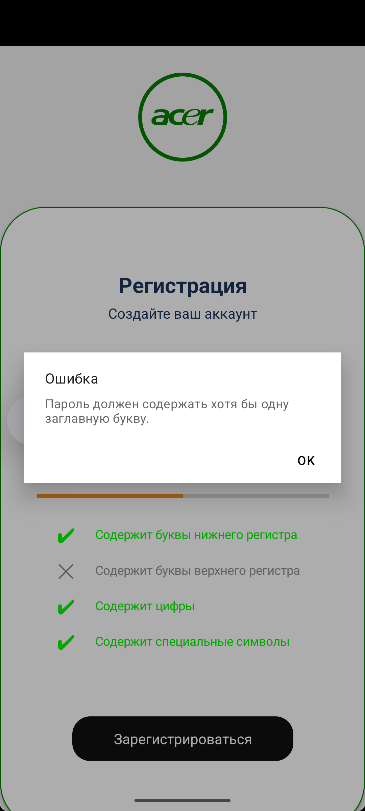
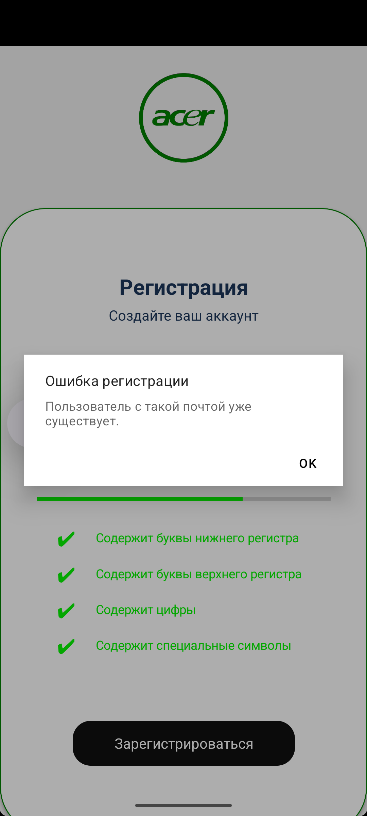
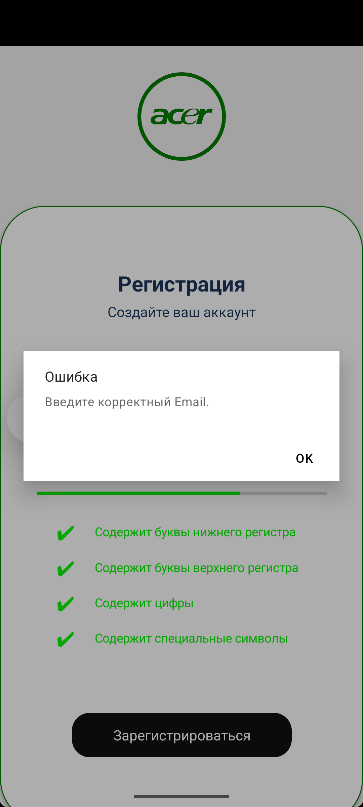


Рисунок 6.2 – Тестирование регистрации

После авторизации или регистрации пользователь попадает на главную страницу с отображением всех товаров.

Также необходимо протестировать выбор тем приложения. Они напрямую зависят от выбранной темы на мобильном устройстве(тёмная и светлая). Продемонстрированно на рисунке 6.3.

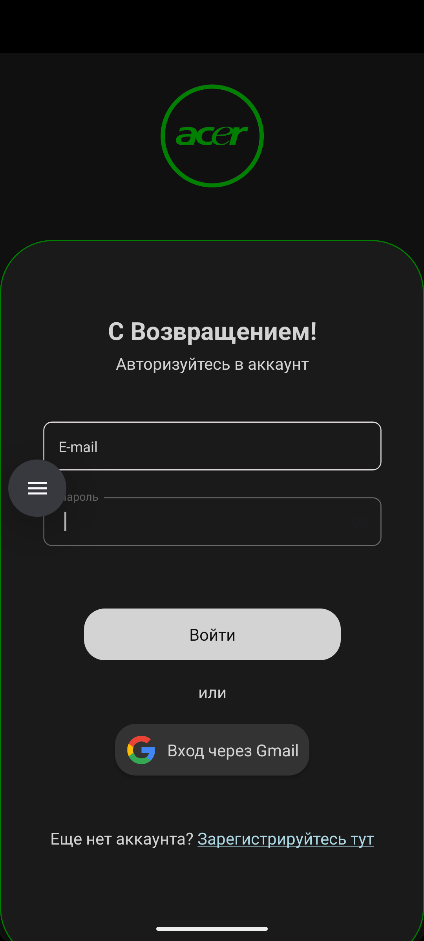
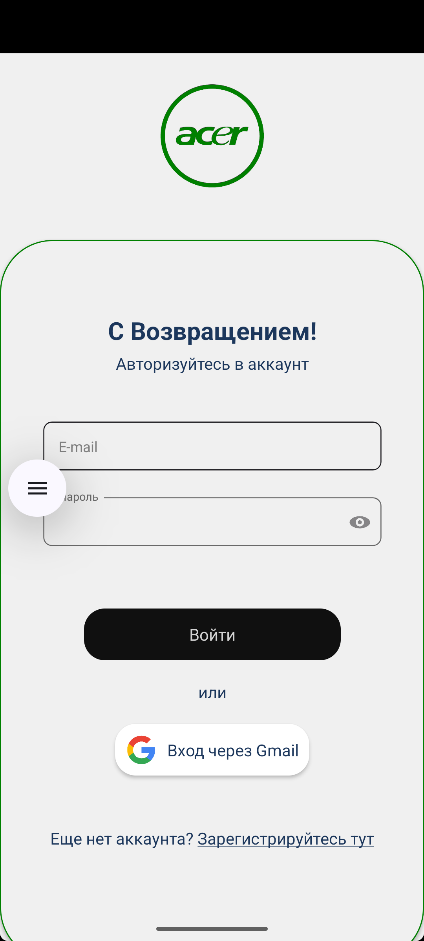


Рисунок 6.3 – Тестирование тем приложения

## **6.2 Вывод по разделу**

В рамках данного раздела была проведена всесторонняя проверка корректности обработки данных в мультимедийном приложении, охватывающая как базовые сценарии работы, так и различные граничные случаи. Особое внимание уделялось тестированию критически важных функциональных модулей, включая процессы регистрации и аутентификации пользователей. Были детально проанализированы различные сценарии ввода неполных, некорректных и заведомо ошибочных данных, что позволило выявить и устранить потенциальные уязвимости на ранних этапах разработки.

Комплексная система валидации входных данных, реализованная в приложении, продемонстрировала высокую эффективность в обработке исключительных ситуаций. Разработанные механизмы обработки ошибок обеспечивают не только стабильность работы системы, но и понятную обратную связь для пользователей при возникновении проблем. Тестирование подтвердило, что приложение корректно реагирует на все возможные варианты ввода данных, включая пограничные случаи, что существенно повышает его надежность и отказоустойчивость.

Проведенные испытания стали важнейшим этапом жизненного цикла разработки, позволившим гарантировать стабильную работу приложения в различных условиях эксплуатации. Реализованные проверки и обработчики исключений создают надежную основу для бесперебойного функционирования системы, полностью соответствующего ожиданиям конечных пользователей. Особую ценность представляет тестирование в условиях, максимально приближенных к реальной эксплуатации, что обеспечивает высокий уровень доверия к качеству разработанного решения.

Важность всестороннего тестирования невозможно переоценить - это не просто обязательный этап разработки, а фундаментальный процесс, непосредственно влияющий на качество конечного продукта. Грамотно организованный процесс тестирования позволяет выявлять потенциальные проблемы на самых ранних стадиях, когда их устранение требует минимальных затрат времени и ресурсов. Более того, он обеспечивает не только техническую надежность решения, но и формирует тот уровень пользовательского опыта, который становится ключевым конкурентным преимуществом приложения.

## **7. Руководство пользователя**

В данном разделе описывается руководство для зарегистрированных пользователей, а именно для пользователя и администратора. У каждой роли есть свой функционал, что позволяет эффективно управлять системой и обеспечивает удобство в использовании.

## **7.1 Пользователь**

Для того, чтобы пользователь мог получить доступ к основному функционалу приложения доступна авторизация через почту и через Google. Страница с авторизацией представлена на рисунке 7.1.

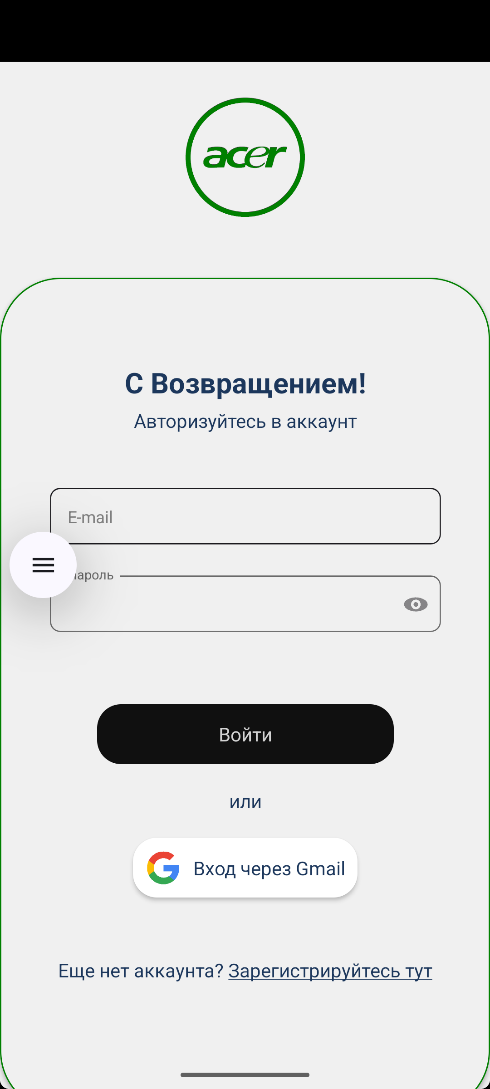


Рисунок 7.1 – Страница авторизации

В случае, если у пользователя еще нет аккаунта, он может нажать на кнопку «Sign Up» и быть перенаправленным на страницу регистрации, которая продемонстрирована на рисунке 7.2.

При регистрации пользователю требуется указать электронную почту, которая будет использоваться для входа в систему. Также пользователю необходимо придумать надежный пароль.

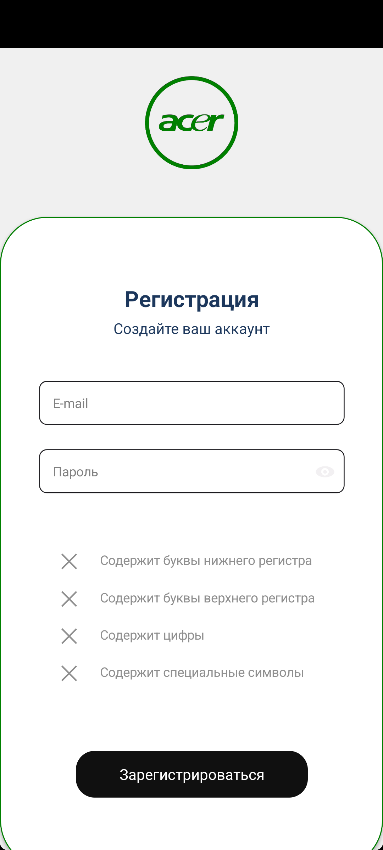


Рисунок 7.2 – Страница регистрации

После успешной регистрации и авторизации пользователь попадает на главный экран приложения под своим уже созданным аккаунтом. На главном экране приложения находится поисковая строка, фильтры, и карточки товаров.

На нижней части экрана расположено навигационное меню, с возможностью перехода на экран корзины и профиля пользователя. Данная страница продемонстрирована на рисунке 7.3.

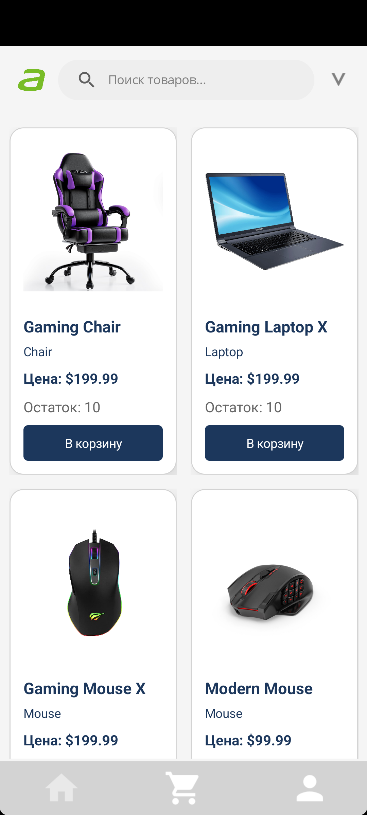


Рисунок 7.3 – Главная страница приложения

В профиле пользователь может загрузить фото и изменить информацию о себе, так же пользователь может сменить аккаунт. Фото представлены на рисунке 7.4.

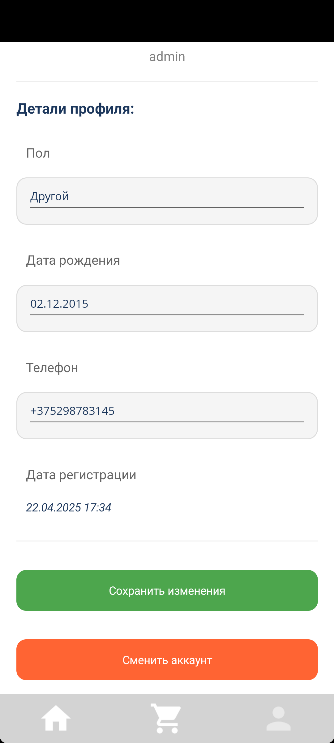
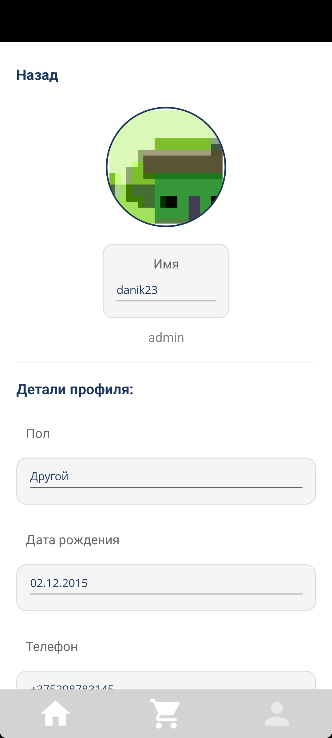


Рисунок 7.4 – Страница профиля

Пользователь может использовать масштабируемую сортировку для поиска нужных товаров. Использование масштабируемой сортировки представлено на рисунке 7.5.

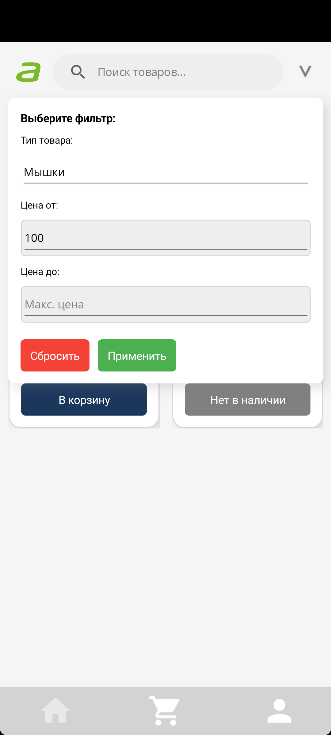
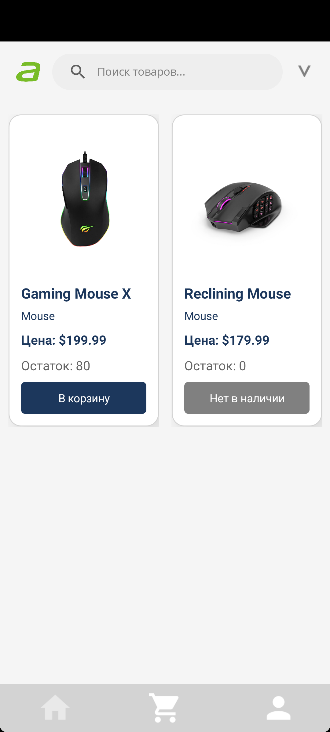


Рисунок 7.4 – Использование сортировки

Далее, нажав на кнопку добавления в корзину, пользователь добавит товар в корзину . Вид страницы корзины представлен на рисунке 7.5.

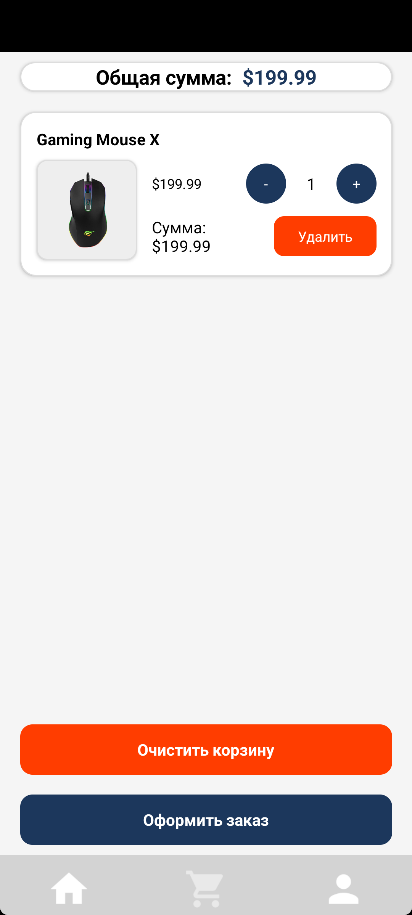


Рисунок 7.5 – Страница корзины пользователя

Нажав на кнопку оформления заказа, пользователь купит все товары, которые находятся в корзине. Демонстрация покупки на рисунке 7.6.

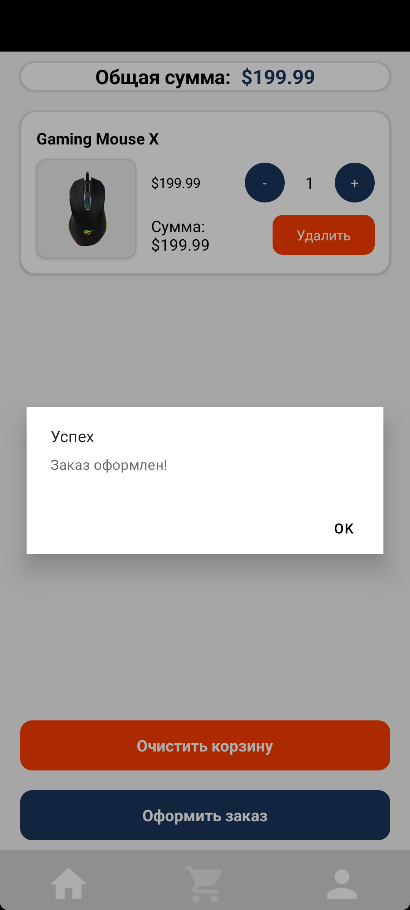


Рисунок 7.6 – Страница корзины пользователя

## **7.2 Администратор**

Курсовой проект разрабатывался с использованием фреймворка .NET MAUI, что позволяет приложению быть кроссплатформенным. Для администратора была разработана отдельная функциональность. Чтобы продолжить работу в приложении как администратор, пользователь должен иметь специальную роль, которую выдаёт главный администратор непосредственно через базу данных, а после администратор должен войти в приложение под соответствующей учетной записью. Такой пользователь может управлять товарами, а именно добавлять товары посредством указания всех деталей товара и выбора соответствующей фотографии, а также администратор может редактировать уже существующие товары, например если описание не соответствует действительности. Еще одной возможностью администратора является удаление товаров, в случае если товар больше не выпускается. При создании панели администратора соблюдался минимальный эргономичный дизайн для упрощения взаимодействия со страницей, а так же добавлена поисковая строка для удобства поиска товара над которыми нужно совершить какое либо действие. Страница для администратора представлена на рисунке 7.10.

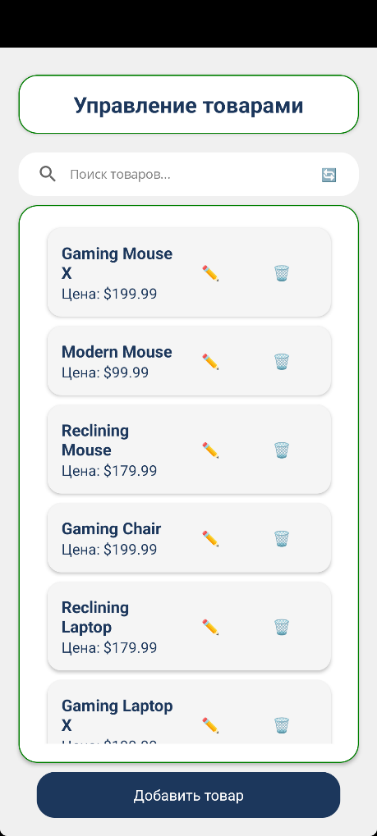


Рисунок 7.10 – Страница администратора

Как говорилось администратор может редактировать, удалять и добавлять товары, а также администратор может осуществлять поиск нужного для работы с ним товара, что видно на рисунке Страница редактирования товара представлен на рисунке 7.11.

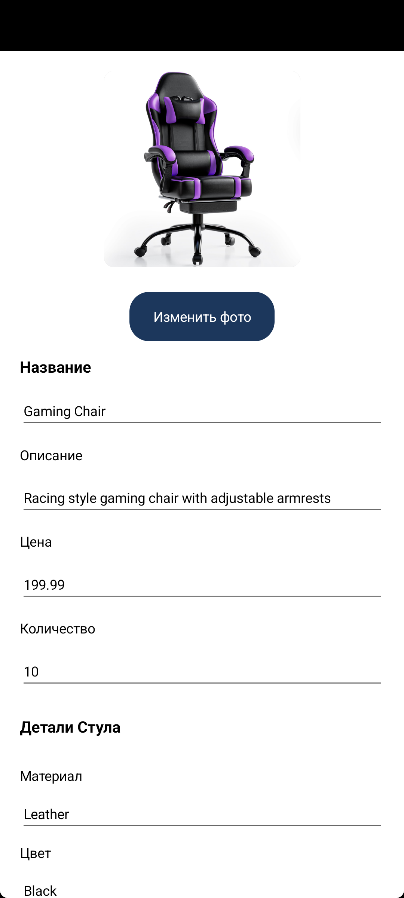


Рисунок 7.11 – Страница профиля для администратора

Для удаления товара администратору необходимо найти устаревший товар и нажать на иконку удаления, а после повторным действием подтвердить данное действие. Это создано для того, чтобы администратор по случайности не удалил нужный товар из каталога.

Для выхода из учетной записи администратору нужно нажать на иконку выхода в профиле.

## **7.3 Вывод по разделу**

Цель данного раздела заключалась в создании руководства пользователя, в соответствии с разработанным программным продуктом. В разделе представлены основные моменты работы с приложением для пользователя и администратора. В нем содержится подробное описание того, что каждая роль может делать в пределах данного мобильного приложения. Также описан порядок действий для достижения постеленных целей при выполнении работы.

Заключение

В ходе выполнения курсового проекта было разработано мобильное приложение интернет-магазина электроники, предоставляющее пользователям удобный доступ к каталогу товаров, функционалу корзины и оформлению заказов. На начальном этапе проведен детальный анализ существующих решений и сформулированы ключевые требования к функционалу, подтвердившие актуальность разработки.

Проектирование системы включало создание диаграммы вариантов использования, наглядно демонстрирующей взаимодействие разных категорий пользователей с приложением, а также разработку оптимальной структуры базы данных на PostgreSQL с интеграцией Firebase для безопасной аутентификации. Реализованный функционал охватывает все основные потребности пользователей: удобный просмотр и фильтрацию товаров по категориям и характеристикам, управление корзиной, процесс оформления заказа, а также административные инструменты для управления контентом.

Технологической основой проекта стал современный стек разработки: кроссплатформенное клиентское приложение на .NET MAUI, серверная часть на ASP.NET Core с REST API, надежная система хранения данных на PostgreSQL и безопасная аутентификация через Firebase. Особое внимание уделено вопросам информационной безопасности - реализована многоуровневая система защиты данных, включающая ролевое управление доступом, JWT-аутентификацию и строгие правила безопасности Firebase.

Работа над проектом позволила не только закрепить теоретические знания, но и получить ценный практический опыт полного цикла разработки - от проектирования архитектуры до тестирования готового решения. Результатом стало стабильное и безопасное приложение, отвечающее всем современным требованиям к электронной коммерции. Перспективы развития включают расширение функциональности за счет интеграции платежных систем, внедрения рекомендательных алгоритмов и увеличения ассортиментной базы.

Список литературы

1) Приложение Instacart [Электронный ресурс]. – Режим доступа https://www.instacart.com/ Дата доступа: 10.03.2025.

2) Приложение Amazon [Электронный ресурс]. – Режим доступа https://www.amazon.com/ Дата доступа: 10.03.2025.

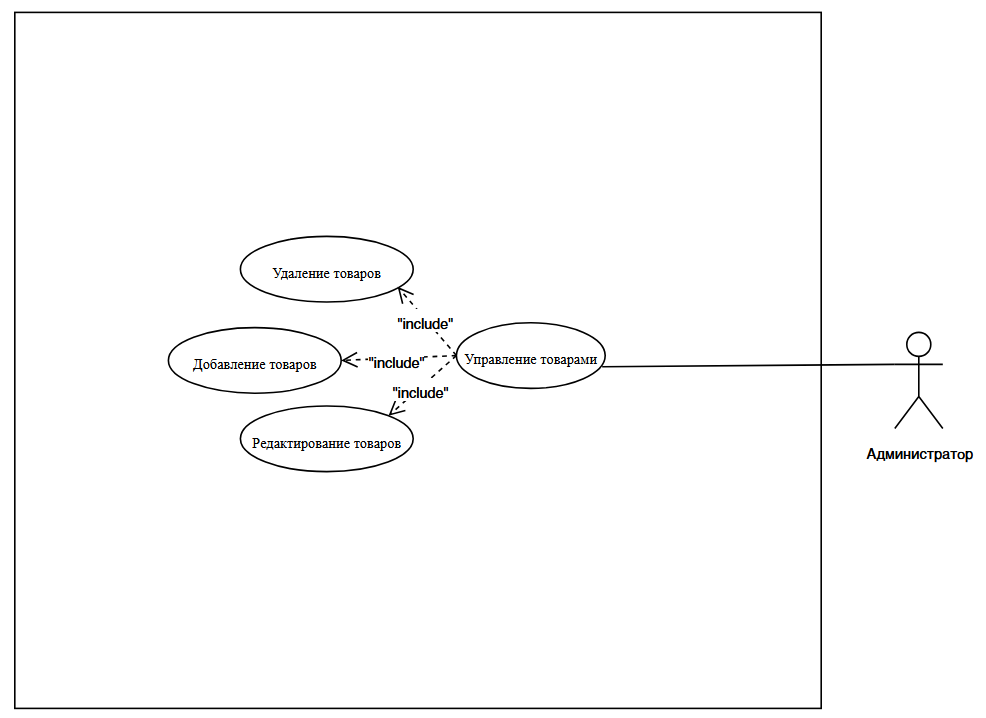
3) Документация Microsoft [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/en-us/answers/questions/1487090/how-to-integrate-firebase-analytics-in-net-maui Дата доступа: 22.04.2025.

4) Metanit [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/maui/ Дата доступа:22.04.2025.

# **Приложение А. Диаграмма вариантов использования пользователя**



# **Приложение Б. Диаграмма вариантов использования администратора**



# **Приложение В. Диаграмма схемы базы данных**

