Министерство образования Республики Беларусь

Учреждения 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**

**ДИПЛОМНОГО ПРОЕКТА НА ТЕМУ:**

Мобильное приложение «Интернет-магазин спортивной обуви»

Обучающийся 4 курс, 7 группа И.А. Ероховец

курс, группа дата подпись И. О. Ф.

Руководитель

дипломного проекта асс. Н.И. Уласевич

должность, ученая степень, ученое звание дата подпись И. О. Ф.

И.о. заведующего

кафедрой ст. преп. Е.А. Блинова

должность, ученая степень, ученое звание дата подпись И. О. Ф.

Консультант ст. преп. А.С. Соболевский

должность, ученая степень, ученое звание дата подпись И. О. Ф.

Нормоконтролер асс. А.Н. Николайчук

должность, ученая степень, ученое звание дата подпись И. О. Ф.

Дипломный проект защищен с оценкой

Председатель ГЭК к.т.н., доцент В.К. Дюбков

должность, ученая степень, ученое звание дата подпись И. О. Ф.

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

УТВЕРЖДАЮ

И. о. заведующего кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Блинова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

**ЗАДАНИЕ**

**на дипломный проект студенту**

Ероховцу Ивану Александровичу

(фамилия, имя, отчество)

1. Тема проекта: Мобильное приложение «Интернет-магазин спортивной обуви»

2. Тема утверждена приказом по университету от 08.05.25 № 143-с

3. Срок сдачи студентом законченного проекта: 05.06.2025 г.

4. Исходные данные к проекту (требования к системе):

4.1 Назначение: *мобильное приложение, предоставляющее возможность выбора и приобретения спортивной обуви.*

4.2 Основные функциональные возможности: *регистрация и авторизация пользователей, просмотр обуви, добавление выдранной обуви в корзину, оформление заказа, просмотр статуса заказов, выбор адреса, сортировка, редактирование, создание и удаление товара, написание отзывов, заявки на поставщика, просмотр личного кабинета, программа лояльности, уведомления об изменениях, поделиться ссылкой на товар.*

4.3 Пользовательские роли: *Администратор (удаление товаров, модерирование комментариев, выдача доступа к созданию товаров, добавление категории, добавление адреса); Пользователь (просмотр одежды, добавление одежды в корзину, оформление заказа, просмотр статуса заказов, сортировка, поделиться товаром, просмотр мои заказы, управление аккаунтом), Поставщик (создание и редактирование своего товара); Продавец (выдача товара со склада, изменения статуса заказа).*

4.4 Целевая аудитория: *люди, владеющие базовыми навыками использования смартфона.*

4.5 Программная платформа: *Android Studio 2024, Firebase. Flutter 3.29, Dart 3.7.2.*

5. Содержание расчетно-пояснительной записки:

1. *Реферат*
2. *Содержание*
3. *Введение*
4. *Раздел 1: обзор аналогов и постановка задачи*
5. *Раздел 2: проектирование мобильного приложения*
6. *Раздел 3: программная реализация приложения*
7. *Раздел 4: тестирование мобильного приложения*
8. *Раздел 5: руководство пользователя*
9. *Раздел 6: анализ информационной безопасности приложения*
10. *Раздел 7: технико-экономическое обоснование проекта*
11. *Заключение*
12. *Список использованных источников*
13. *Приложения и графическая часть*

6. Перечень графического материала (с точным указанием обязательных чертежей):

1. *Диаграмма развертывания*
2. *Логическая схема базы данных*
3. *Диаграмма вариантов использования*
4. *Блок-схема алгоритма оформления заказа*
5. *Блок-схема алгоритма смены пароля*
6. *Таблица расчета экономических показателей*
7. *Скриншот главной страницы*

7. Консультанты по проекту с указанием относящихся к ним разделов проекта

|  |  |
| --- | --- |
| Раздел | Консультант |
| *Технико-экономическое обоснование проекта* | *А. С. Соболевский* |

8. Дата выдачи задания:

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.И. Уласевич

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.А. Ероховец

(подпись)

9. Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов дипломного проекта | Срок выполнения этапов проекта | Примечание |
| *1* | *Обзор аналогов и постановка задачи* | 25.03–31.03.24 |  |
| *2* | *Проектирование мобильного приложения* | 01.04–07.04.24 |  |
| *3* | *Программная реализация приложения* | 08.04–28.04.24 |  |
| *4* | *Анализ информационной безопасности приложения* | 29.04–02.05.24 |  |
| *5* | *Тестирование мобильного приложения* | 03.05–06.05.24 |  |
| *6* | *Руководство пользователя* | 07.05–12.05.24 |  |
| *7* | *Технико-экономическое обоснование проекта* | 13.05–19.05.24 |  |
| *8* | *Оформление пояснительной записки* | 20.05–28.05.24 |  |
| *9* | *Рецензирование дипломного проекта* | 17.06–20.06.24 |  |
| *10* | *Допуск рабочей комиссии к защите в ГЭК* | 17.06–21.06.24 |  |
| *11* | *Защита дипломного проекта* | 24.06–26.06.24 |  |

Дипломник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руководитель проекта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

**Реферат**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

Утв.

Реферат

Лит.

Листов

1

*-, 2025*

У

**Abstract**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

Утв.

Abstract

Лит.

Листов

1

*-, 2025*

У

Содержание

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Ероховец И.А*

Пров.

*Уласевич Н.И*

Н. контр.

.

.

Утв.

Содержание

Лит.

Листов

2

*-, 2025*

У

[Введение 6](#_Toc194299233)

[1 Обзор аналогов и постановка задачи 7](#_Toc194299234)

[1.1 Обзор аналогов проектируемого мобильного приложения 7](#_Toc194299235)

[1.1.1 Приложение «Lamoda» 7](#_Toc194299236)

[1.1.2 Мобильное приложение «Mark Formelle» 9](#_Toc194299237)

[1.1.3 Мобильное приложение «Ostin» 10](#_Toc194299238)

[1.2 Постановка задачи 11](#_Toc194299239)

[1.3 Выводы по разделу 11](#_Toc194299240)

[2 Проектирование мобильного приложения 13](#_Toc194299241)

[2.1 Выбор технологий и средств 13](#_Toc194299242)

[2.1.1 Среда разработки Android Studio 13](#_Toc194299243)

[2.1.2 Язык программирования Dart 14](#_Toc194299244)

[2.1.3 Облачная база данных Firebase 15](#_Toc194299245)

[2.1.4 Фреймворк Flutter 15](#_Toc194299246)

[2.2 Проектирование базы данных 16](#_Toc194299247)

[2.3 Проектирование мобильного приложения 20](#_Toc194299248)

[2.4 Диаграммы и блок-схемы приложения 22](#_Toc194299249)

[2.4.1 Диаграмма вариантов использования 22](#_Toc194299250)

[2.5.1 Диаграмма развертывания 24](#_Toc194299251)

[2.5.2 Блок-схема оформления заказа из заведения 24](#_Toc194299252)

[2.7 Выводы по разделу 25](#_Toc194299253)

# Введение

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

.

.

.

.

Утв.

Введение

Лит.

Листов

1

*---, 2025*

У

В современном мире, где время является одним из самых ценных ресурсов, люди ежедневно сталкиваются с необходимостью решать повседневные задачи, требующие значительных временных затрат. Одной из таких задач является выбор и покупка одежды, в том числе и обуви. С каждым годом все больше людей стремятся ускорить этот процесс, минимизировать возникающие проблемы и максимально упростить удовлетворение своих потребностей.

В ответ на эти вызовы возникла необходимость в создании мобильных приложений, которые не только упрощают процесс выбора и покупки, но и предоставляют доступ к товарам из разных уголков мира. Такие приложения способны значительно экономить время пользователей, делая процесс более удобным и эффективным.

Целью данного дипломного проекта является разработка мобильного приложения для онлайн-покупки спортивной обуви. Приложение будет не только выполнять базовые функции интернет-магазина, но и предлагать дополнительные возможности: помощь в выборе одежды на основе предпочтений пользователя, доступ к отзывам, а также рекомендации, учитывающие стиль и вкусы клиента.

Для достижения цели дипломного проекта, необходимо решить ряд задач:

* изучить уже существующие аналоги и выявление их преимуществ и недостатков;
* подбор инструментов и платформ для реализации приложения;
* создание базы, содержащей всю необходимую информацию о товарах, пользователях и заказах;
* создание архитектуры, обеспечивающей удобство использования и стабильность работы;
* проектирование интуитивно понятного и визуально привлекательного интерфейса с возможностью фильтрации товаров;
* разделить пользователей системы на четыре роли: администратор, продавец, поставщик и пользователь.

Приложение будет интегрировано с базой данных, что позволит хранить и обрабатывать все необходимые данные о товарах и пользователях. Особое внимание будет уделено созданию удобного интерфейса, который обеспечит комфортное взаимодействие с приложением.

Реализация данного проекта призвана упростить процесс выбора и покупки одежды, сделав его быстрым, удобным и доступным для пользователей. Актуальность проекта обусловлена ростом популярности онлайн-покупок, а также постоянным спросом на одежду и делает разработку такого приложения особенно важной.

# 1 Обзор аналогов и постановка задачи

ФИО

Подпись

Дата

Лист

1

БГТУ 01.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

.

.

Утв.

1 Обзор аналогов и постановка задачи

Лит.

Листов

5

*-, 2025*

У

Для составления технических требований к проекту был проведен полный анализ схожих мобильных приложений. На основе анализа были выделены ключевые факторы, которые стоит учесть при разработке данного мобильного приложения.

## 1.1 Обзор аналогов проектируемого мобильного приложения

В настоящее время представлено достаточно количество мобильных приложений для приобретения обуви онлайн. Основным назначением такого рода приложения – предоставление возможности приобретения какого-либо товара любого предоставленного размера и других критериев. Однако не все приложение следят за актуальностью некоторых товаров, а также их наличия у поставщиков. Важными аспектами при выполнении оценки таких приложений является их функциональность и удобство использования. Также важно, чтобы приложение работало стабильно при высокой нагрузке. Это ещё больше обеспечит комфортное использование для большого количества пользователей одновременно и может кратно увеличить доходы.

Наиболее популярными на данных момент мобильные приложения с похожим функционалом являются:

* Lamoda;
* Zenden;
* Kari.

### 1.1.1 Приложение «Lamoda»

«Lamoda» – это сервис для онлайн покупок, которые предлагает большой выбор одежды и обуви.

Пользователям предоставляется выбор одежды различных брендов и категорий, таких как женская, мужская и детская, а также спортивная обувь. Lamoda предлагает покупателям разные способы доставки, можно как забрать самостоятельно, так и заказать курьера на дом. Присутствуют и различные способы оплаты, включая как оплату наличными так и онлайн-платежи.

Lamoda активно развивает своё приложение, регулярно следит за новыми трендами и пополняется новыми брендами. За счёт удобного интерфейса и функционала, приложение стало популярным среди пользователей.

На главной странице приложения пользователю предлагается выбор своей страны и дальнейший вход в систему. После входа в систему пользователь может ознакомиться с популярными предложениями, которые разбиты на определённые категории, а именно «Одежда» «Обувь» «Аксессуары» и другие, что делает поиск нужных товаров более простым и быстрым.

Приложение стало популярным, благодаря своему удобству использования. На рисунке 1.1 представлена страница со списком товаров.

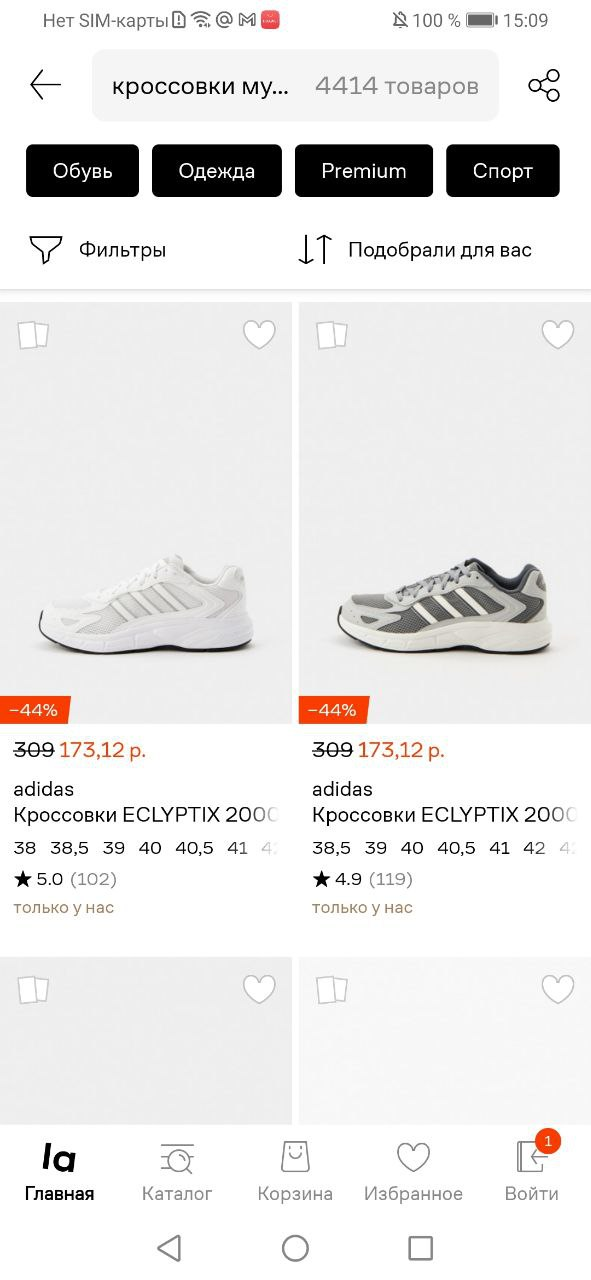


Рисунок 1.1 – Каталог товаров в приложении «Lamoda»

Каждый товар в приложении имеет свою уникальную страницу, на которой представлена вся необходимая информация для потенциального покупателя. Страница товара включает подробное описание товара, доступные размеры, актуальные цены, а также отзывы от других пользователей. Отзывы играют важную роль, так как они позволяют покупателю ознакомиться с мнением людей, которые уже приобрели данный товар. Это помогает сформировать более полное представление о качестве продукции и сделать осознанный и взвешенный выбор при покупке.

Процесс оформления заказа также хорошо продуман и максимально удобен для пользователя. После добавления товаров в корзину покупателю предлагается выбрать наиболее подходящий способ доставки, а также вариант оплаты. Для удобства пользователей предусмотрено несколько способов оплаты: это может быть оплата онлайн, с использованием банковской карты или электронного кошелька, либо оплата наличными при получении товара. Такой подход позволяет каждому покупателю выбрать наиболее комфортный для него вариант.

На рисунке 1.2 представлен интерфейс страницы с товаром.

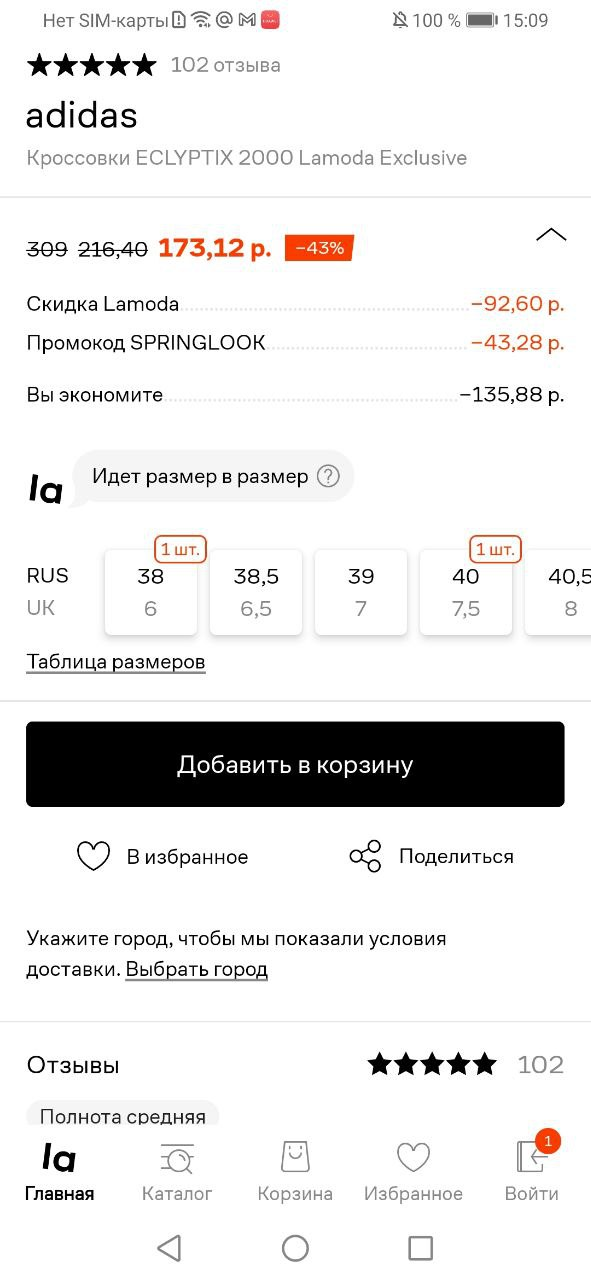


Рисунок 1.2 – Страница товара в приложении «Lamoda»

Также у приложения реализована удобная и функциональная корзина, с которой сможет разобраться любой пользователь. Интерфейс интуитивно понятен, можно быстро добавлять и изменять товары и их количество. Пример корзины представлен на рисунке 1.3.

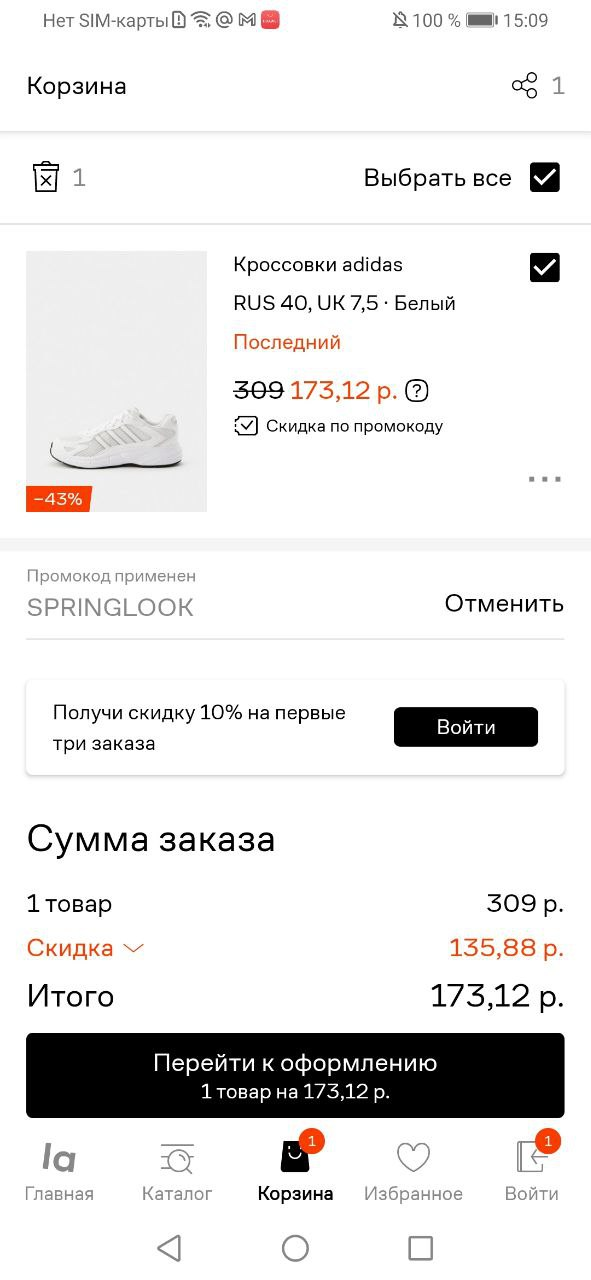


Рисунок 1.3 – Страница корзины в приложении «Lamoda»

Ключевые особенности приложения Lamoda:

* широкий ассортимент обуви;
* удобный поиск и навигация по категориям;
* рейтинг и отзывы;
* возможность оформления заказа и оплаты онлайн;
* интеграция с картами для отслеживания доставки;
* гармонично выбранные цвета и дизайн.

К основным минусам можно отнести большое количества товара, что иногда усложняет навигацию.

В целом, мобильное приложение Lamoda является удобным и функциональным, особенно для тех, кто предпочитает шопинг не выходя из дома.

### 1.1.2 Мобильное приложение «Zenden»

Мобильное приложение «Zenden» предлагает пользователям удобные возможности для покупки обуви, в том числе стильных и спортивных моделей. Основная задача приложения – сделать выбор подходящей пары максимально приятный и быстрым, а также привлечения новых клиентов.

Приложение отличается рядом полезных возможностей, которые делают шопинг комфортным и увлекательным. Среди ключевых функций можно выделить:

* широкий ассортимент обуви для любого случая, включая спортивную;
* возможность добавлять товары в избарнное;
* получение пользовательских рекомендаций на основе предпочтений;
* доступ к эксклюзивным скидкам и акциям.

Однако, несмотря на множество преимуществ, приложение обладает и некоторыми недостатками, которые могут повлиять на впечатления от использования. Например, для полного доступа ко всем функциям, таким как оформление заказов и сохранение товаров в избранное, требуется обязательная регистрация. Это может стать препятствием для пользователей, которые предпочитают делать покупки без создания личного аккаунта.

Еще одним недостатком является отсутствие возможности просматривать историю покупок. Такая функция значительно упростила бы повторное оформление заказов на уже приобретённую обувь.

На рисунке 1.4 представлена страница со списком товаров.

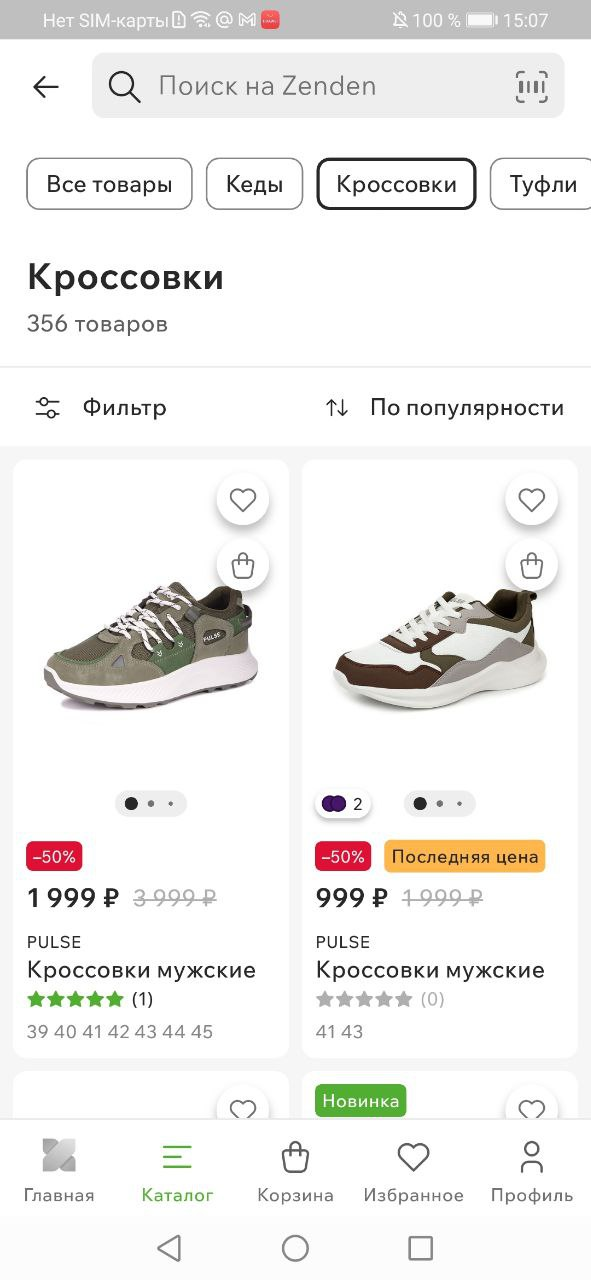


Рисунок 1.4 – Страница с каталогом в приложении «Zenden»

Кроме того, ассортимент приложения ограничивается исключительно продукцией бренда Zenden, что может не подойти пользователям, ищущим широкий выбор товаров разных производителей. На рисунке 1.5 представлена страница товара.

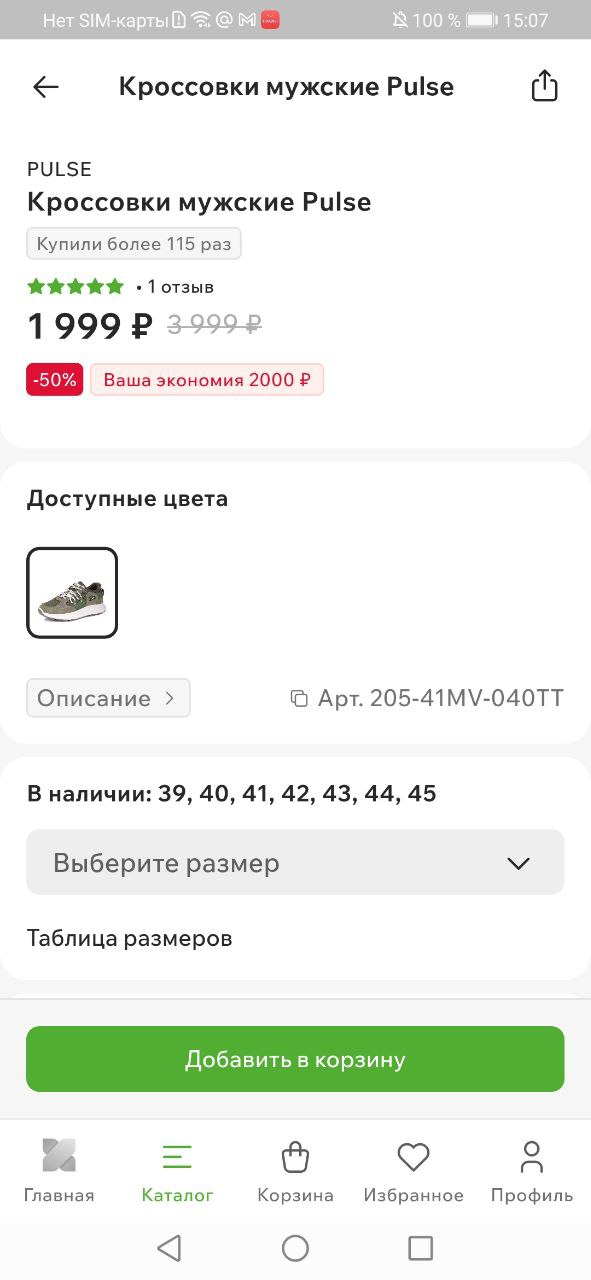


Рисунок 1.5 – Страница товара в приложении «Zenden»

Карточка товара в мобильном приложении Zenden разработана так, чтобы пользователь мог получить всю необходимую информацию о выбранной обуви быстро и удобно. На отдельной странице для каждой модели представлено несколько качественных фотографий, позволяющих рассмотреть обувь со всех сторон и детально изучить материалы, отделку и особенности дизайна.

Для удобства покупателей в карточке отображается актуальная информация о наличии выбранного размера, стоимости, а также о действующих скидках или акциях. При необходимости можно сразу добавить товар в корзину и продолжить оформление заказа.

На рисунке 1.6 представлена страница корзины.

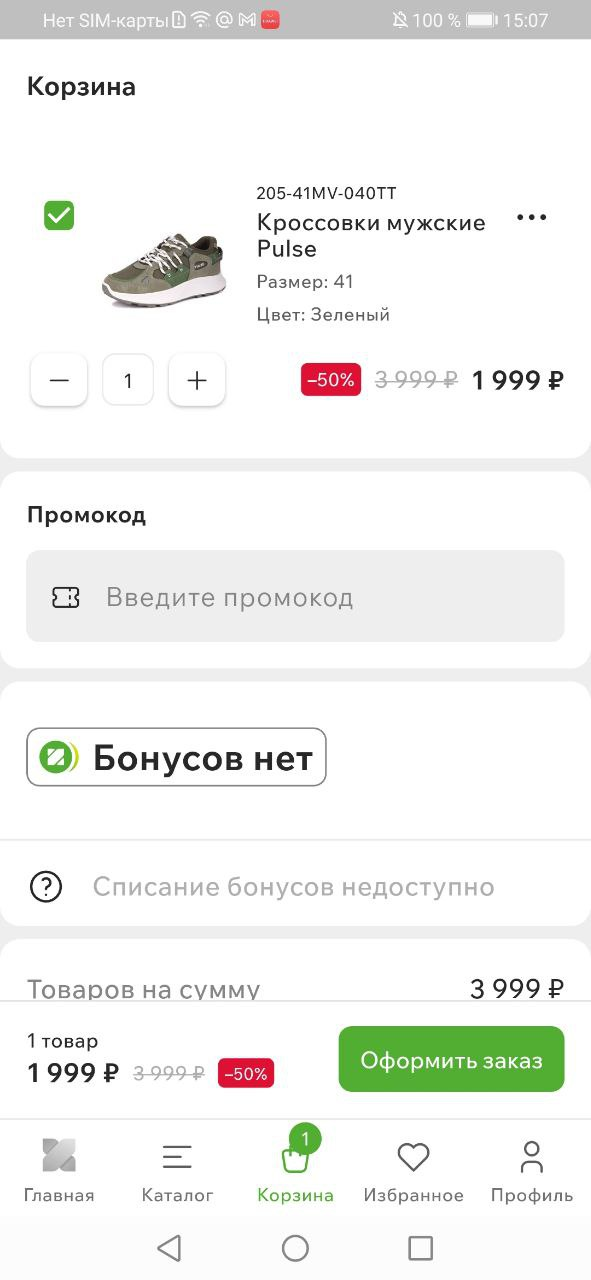


Рисунок 1.6 – Страница корзины в приложении «Zenden»

В целом, мобильное приложение Zenden — это современный и удобный инструмент для покупки обуви, который сочетает в себе богатый функционал и привлекательный дизайн. Тем не менее, такие недостатки, как обязательная регистрация и ограничение ассортимента только на одной марке, могут быть устранены в будущем, что сделает приложение ещё более удобным и привлекательным для покупателей.

### 1.1.3 Мобильное приложение «Kari»

Мобильное приложение для заказа обуви и аксессуаров бренда «Kari» — это удобный и современный инструмент для онлайн-шопинга, который предоставляет пользователям широкий набор полезных функций. Основная задача приложения — обеспечить быстрый и комфортный доступ к ассортименту обуви, включая популярные спортивные модели, а также упростить процесс выбора и оформления заказа.

Интерфейс приложения соответствует актуальным стандартам мобильной коммерции, что делает навигацию по каталогу простой и интуитивно понятной. Пользователь легко может перейти к интересующим его категориям: женская, мужская, детская или спортивная обувь, а также найти аксессуары для завершения образа. Поиск нужной пары осуществляется по названию, параметрам или с помощью фильтров, что особенно удобно при большом ассортименте.

На рисунке 1.7 представлена страница с каталогом.

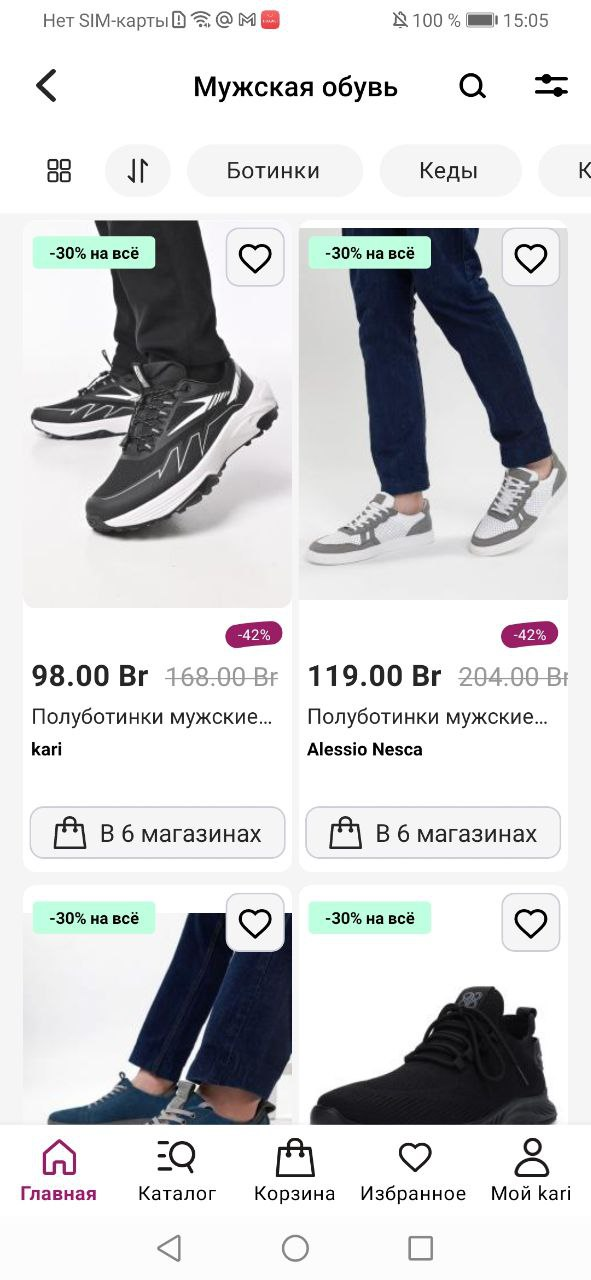


Рисунок 1.7 – Страница с каталогом в приложении «Ostin»

Карточка товара в приложении содержит детальное описание выбранной обуви, фотографии с разных ракурсов, информацию о материалах, размерах и наличии, а также отзывы других покупателей. Это помогает сделать осознанный выбор и подобрать наиболее подходящую модель.

К главным особенностям данного приложения можно отнести:

* широкий выбор обуви;
* возможность отслеживания истории покупок;
* простой процесс оформления и изменения заказов.

На рисунке 1.8 можно увидеть страницу с товаром.



Рисунок 1.8 –Страница товара в приложении «Ostin»

В приложении реализована корзина, где можно управлять выбранными товарами: изменять их количество, удалять ненужные позиции и видеть итоговую сумму заказа. После этого достаточно нажать кнопку «Оформить заказ», чтобы перейти к быстрой и понятной процедуре покупки. На всех этапах оформления заказа пользователь может изменить детали, выбрать способ доставки и оплаты, а также проверить актуальность данных.

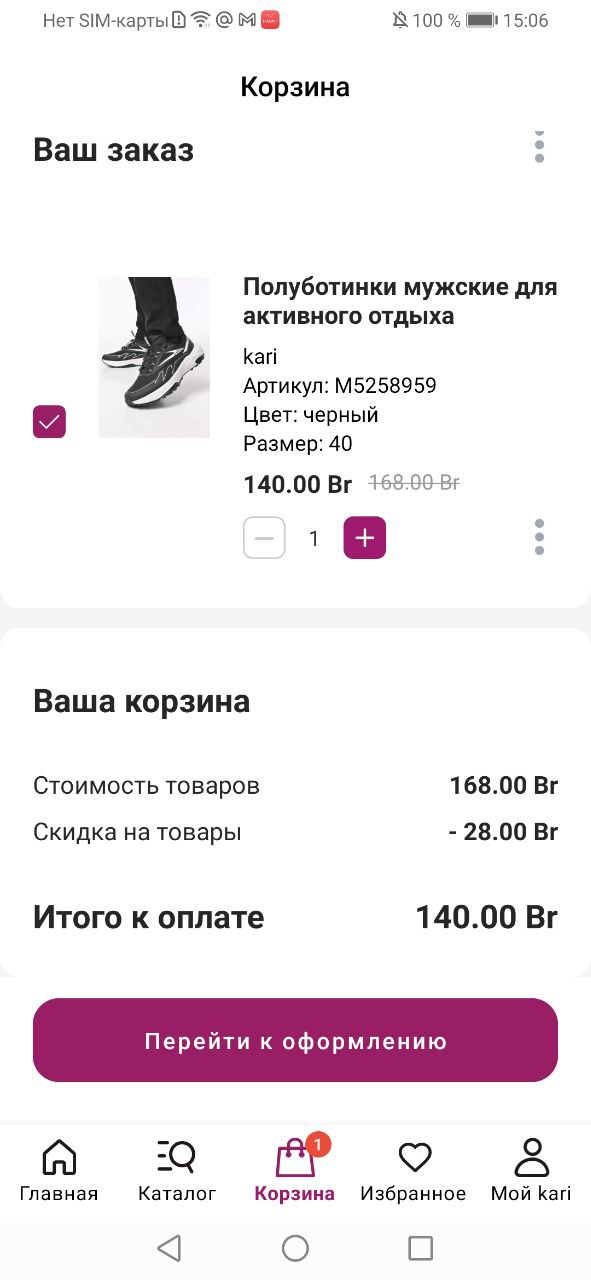
`

Рисунок 1.9 – Страница корзины в приложении «Kari»

Несмотря на множество преимуществ, у приложения есть и некоторые недостатки. Так, для его корректной работы требуется стабильное подключение к интернету. К тому же информация о наличии некоторых моделей и актуальных акциях может обновляться с задержкой, что иногда вызывает неудобство у покупателей. Тем не менее, возможность просматривать и редактировать детали заказа перед подтверждением покупки обеспечивает дополнительный контроль и удобство.

В целом, приложение «Kari» обладает широким функционалом, который удовлетворяет основные потребности современных покупателей обуви. Лаконичный дизайн, отсутствие лишних элементов и продуманные функции делают его эффективным инструментом для быстрой и приятной покупки обуви, в том числе спортивной. Улучшение скорости обновления данных об ассортименте и скидках может сделать этот сервис ещё более привлекательным для пользователей в будущем.

## 1.2 Постановка задачи

Разрабатываемый проект, является мобильным приложением для выбора и приобретения спортивной обуви, оформление, заказ, а так-же отслеживания товара и у приложения должен быть следующий функционал:

* регистрация и авторизация пользователей;
* возможность просмотра товара;
* возможность фильтрации товара;
* возможность выбора товара по размеру и количества;
* возможность поделиться товаром;
* возможность добавления товара в корзину;
* возможность оформления заказа и его доставку;
* получать уведомления;
* управление личным кабинетом;
* возможность пользователю просматривать список всех его заказов.

## 1.3 Выводы по разделу

В данной главе были разобраны три самых популярных приложения, которые являются аналогами приложения мобильное приложение «Интернет-магазин спортивной обуви»: «Lamoda», «Kari» и «Zenden». Мы остановились на трёх приложениях, так как остальные практически не отличаются по функционалу и пользовательскому интерфейсу.

Важно отметить, что рынок мобильных приложений для онлайн-покупки обуви сегодня очень динамичен и конкурентен. Каждый разработчик стремится внедрять новые решения для повышения удобства пользователей, делая упор не только на базовый функционал, но и на детали интерфейса, индивидуальные предложения и дополнительные возможности. Благодаря этому пользователи могут выбирать те сервисы, которые максимально соответствуют их ожиданиям по скорости работы, простоте оформления заказов и качеству обслуживания.

Кроме того, современные тренды в разработке мобильных приложений диктуют особое внимание к визуальному оформлению и пользовательскому опыту. Красивый и минималистичный дизайн, удобная навигация, своевременные уведомления и прозрачная система отслеживания заказов становятся неотъемлемой частью успешного продукта. Именно поэтому анализ аналогичных решений позволяет не только выявить сильные и слабые стороны конкурентов, но и сформировать собственное видение идеального приложения для интернет-магазина спортивной обуви.

Из приложений «Lamoda» я выделю то, что нужно работать во всех направлениях, а именно сотрудничать с разными брендами и сделать личный кабинет со всей информацией о заказе и пользователе, а из «Kari» и «Zenden» – взять за идею карточку товара, которая сделана достаточно просто и красиво, без лишних деталей.

Проанализировав все вышесказанное можно выделить основной функционал разрабатываемого в курсовом проекте мобильного приложения для спортивной обуви:

* регистрация и авторизация пользователей;
* наличие нижней навигационной панели;
* возможность просмотра информации в личном кабинете;
* поиск и сортировка товара по разным параметрам;
* просмотр заказов и их статусов.

Стоит отметить, что анализ аналогичных приложений позволяет выявить лучшие практики и технологии, применяемые другими разработчиками. Это может помочь оптимизировать функциональность и удобство использования нашего приложения, что подчеркивает важность такого обзора.

# 2 Проектирование мобильного приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

.

.

.

.

Утв.

2 Проектирование мобильного приложения

Лит.

Листов

15

*-, 2025*

У

## 2.1 Выбор технологий и средств

При реализации дипломного проекта использовались язык программирования Dart и фреймворк Flutter.

В качестве средств разработки, использовались программные средства Android Studio, Visual Studio Code.

### **2.1.1 Среда разработки Android Studio**

Android Studio — это мощная и бесплатная среда разработки, разработанная компанией Google специально для создания приложений на платформе Android. Она предоставляет обширный набор инструментов, которые значительно упрощают процесс разработки и развертывания мобильных приложений.

Android Studio предлагает широкий набор функций, направленных на повышение производительности разработчика. Среди них поддержка подсветки синтаксиса, автодополнения кода, мощные средства отладки и профилирования, а также интеграция с системами контроля версий, такими как Git. Эти возможности помогают разработчикам сосредоточиться на написании кода, снижая затраты времени на рутинные задачи.

Одним из главных преимуществ Android Studio является встроенная поддержка Gradle-системы автоматизации сборки. Gradle позволяет эффективно управлять зависимостями, конфигурациями сборки и процессом развертывания, что особенно полезно при работе с крупными проектами.

Для упрощения тестирования приложений, Android Studio предоставляет мощные инструменты, такие как эмуляторы устройств, позволяющие запускать и тестировать приложения на различных конфигурациях Android без необходимости использования физического устройства. Это значительно ускоряет процесс отладки и улучшения качества приложения.

Одной из ключевых особенностей Android Studio является его тесная интеграция с Android SDK, что обеспечивает разработчикам доступ ко всем новейшим API и библиотекам. Это позволяет быстро адаптироваться к изменениям в платформе Android и использовать новейшие возможности операционной системы.

Android Studio разработана с учетом производительности и ресурсоэффективности, что позволяет комфортно работать с ней даже на компьютерах со скромными характеристиками. Кроме того, среда разработки поддерживает настройку пользовательского интерфейса, что позволяет разработчикам адаптировать ее под свои индивидуальные предпочтения.

На протяжении многих лет среда разработки Android Studio зарекомендовала себя как стандарт для разработки приложений под Android, благодаря своей стабильности, функциональности и поддержке сообщества. Она предоставляет все необходимые инструменты для создания высококачественных мобильных приложений, что делает ее незаменимым инструментом для Android-разработчиков. Помимо этого, регулярные обновления и улучшения Android Studio помогают разработчикам оставаться в курсе новейших технологий и методик, что способствует созданию инновационных и современных приложений.

### 2.1.2 Язык программирования Dart

Dart — это язык программирования общего назначения, разработанный компанией Google, предназначенный прежде всего для создания прикладных приложений. Его основной сферой применения на сегодняшний день является разработка графических приложений с использованием кроссплатформенного фреймворка Flutter. Flutter позволяет создавать клиентские веб-приложения, мобильные приложения под Android и iOS, а также десктопные приложения для Windows, macOS и Linux. Это означает, что мы можем написать программу на Dart с общим кодом и скомпилировать её сразу под все поддерживаемые платформы, включая Android, iOS, Windows, Linux, macOS и веб-браузеры. Однако применение языка Dart не ограничивается только фреймворком Flutter.

В своем развитии Dart испытал влияние более ранних языков, таких как Smalltak, Java, JavaScript. Его синтаксис похож на синтаксис других си-подобных языков. При этом Dart - объектно-ориентированный язык. Все значения, которые используются в программе на Dart, представляют объекты.

Dart предоставляет мощные инструменты и функции, которые существенно упрощают создание мобильных и веб-приложений. Он гарантирует отличную производительность и удобство использования благодаря статической и динамической типизации, что дает возможность выявлять ошибки на этапе компиляции и одновременно обеспечивает гибкость в написании кода. Эти особенности способствуют созданию более надежного и безопасного кода программистами, позволяя сократить время на его отладку.

Одним из ключевых преимуществ Dart является его асинхронная модель программирования, которая позволяет эффективно обрабатывать параллельные задачи, что особенно полезно в приложениях с высокой нагрузкой. Это делает Dart идеальным выбором для разработки отзывчивых пользовательских интерфейсов и работы с сетевыми запросами.

Dart также предлагает обширную стандартную библиотеку, содержащую множество полезных функций и классов для работы с коллекциями, файлами, сетевыми запросами и другими задачами. Данная библиотека существенно уменьшает время разработки, позволяя сконцентрироваться на логике приложения.

### **2.1.3 Облачная база данных Firebase**

Firebase — платформа для разработки мобильных и веб-приложений, разработанная компанией Google. Она представляет собой обширный инструментарий и сервисы, позволяющие разработчикам оперативно создавать, тестировать и внедрять приложения, а также управлять их функциональностью и производительностью.

Основным элементом Firebase является облачное хранилище данных Firestore, предоставляющее возможности хранения и синхронизации данных в режиме реального времени. Firestore упрощает создание и управление базами данных, обеспечивая высокую доступность и масштабируемость приложений. Поддержка оффлайн-режима гарантирует возможность работы с приложением даже при временном отсутствии интернет-соединения.

Firebase также предлагает мощные средства аутентификации пользователей, поддерживающие различные методы, включая электронную почту, социальные сети и анонимный вход. Это упрощает управление пользователями и повышает безопасность приложений.

В дополнение, Firebase содержит инструменты аналитики, которые помогают разработчикам отслеживать поведение пользователей и оценивать производительность приложений. Инструменты аналитики предоставляют ценную информацию о взаимодействии пользователей с приложением, что позволяет принимать обоснованные решения по его улучшению.

Firebase предоставляет возможность отправки уведомлений в реальном времени посредством Firebase Cloud Messaging (FCM). Этот сервис позволяет разработчикам отправлять push-уведомления пользователям на мобильных устройствах и в веб-приложениях, что способствует повышению вовлеченности и удержанию пользователей.

Другой значимой функцией Firebase является хостинг, позволяющий разработчикам быстро разворачивать статические и динамические веб-приложения с поддержкой HTTPS. Хостинг Firebase обеспечивает высокую скорость загрузки и легкость управления, что делает его отличным выбором для веб-разработки.

Firebase также интегрируется с другими сервисами Google, такими как Google Analytics и Google Cloud, что позволяет разработчикам использовать мощные инструменты для анализа и масштабирования приложений.

За последние годы Firebase подтвердил свою репутацию надежной платформы для разработки приложений, благодаря своей функциональности, удобству использования и поддержке сообщества. Регулярные обновления и новые функции делают Firebase современным инструментом, который помогает разработчикам создавать качественные и высокопроизводительные приложения.

### **2.1.4 Фреймворк Flutter**

Flutter — это бесплатный и невероятно удобный инструмент разработки приложений, созданный компанией Google. С его помощью можно создавать приложения не только для мобильных платформ Android и iOS, но и для веба и настольных систем, что делает его идеальным вариантом для разработчиков,стремящихсякэкономиивремениисил,разрабатываякроссплатформенные решения.

Главное преимущество Flutter – его обширный функционал. Он включает в себя богатую библиотеку виджетов, а также функцию "горячей перезагрузки", позволяющую мгновенно видеть изменения в коде. Это значительно ускоряет процесс разработки. К тому же, интеграция с различными инструментами тестирования и отладки облегчает поиск и исправление ошибок.

Flutter также обеспечивает отличную производительность благодаря своей системе рендеринга. Он использует собственный движок для графики и анимации, поэтому приложения выглядят одинаково хорошо на всех платформах и работают плавно и быстро.

Для тестирования Flutter предоставляет мощные инструменты, включая фреймворки для модульного и интеграционного тестирования. Это позволяет проверять каждую часть приложения, обеспечивая его качество на всех этапах разработки.

Ещё одно значительное преимущество Flutter — возможность использования единой кодовой базы для всех платформ. Разработчики пишут код один раз, и он работает на Android, iOS, вебе и десктопах. Это значительно экономит время и усилия, избавляя от необходимости адаптировать приложение под каждую платформу отдельно.

Не стоит забывать и о поддержке сообщества. Flutter постоянно обновляется, а разработчики могут использовать множество пакетов и библиотек, упрощающих работу. Все эти обновления и улучшения способствуют созданию современных, удобных и функциональных приложений.

В целом, Flutter стал важным инструментом для многих разработчиков благодаря своей гибкости, производительности и активному сообществу. Этот фреймворк позволяет создавать отличные кроссплатформенные приложения, и он подходит для разработчиков, желающих работать быстро и эффективно.

## 2.2 Проектирование базы данных

Первым шагом в разработке приложения является определение предметной области, которой оно будет охватывать, а также проектирование общей архитектуры базы данных. Необходимо четко понимать, какие именно данные будут обрабатываться, где они будут размещены и в каком формате будут представлены.

Логическая схема базы данных со всеми связями представлена на рисунке 2.2.

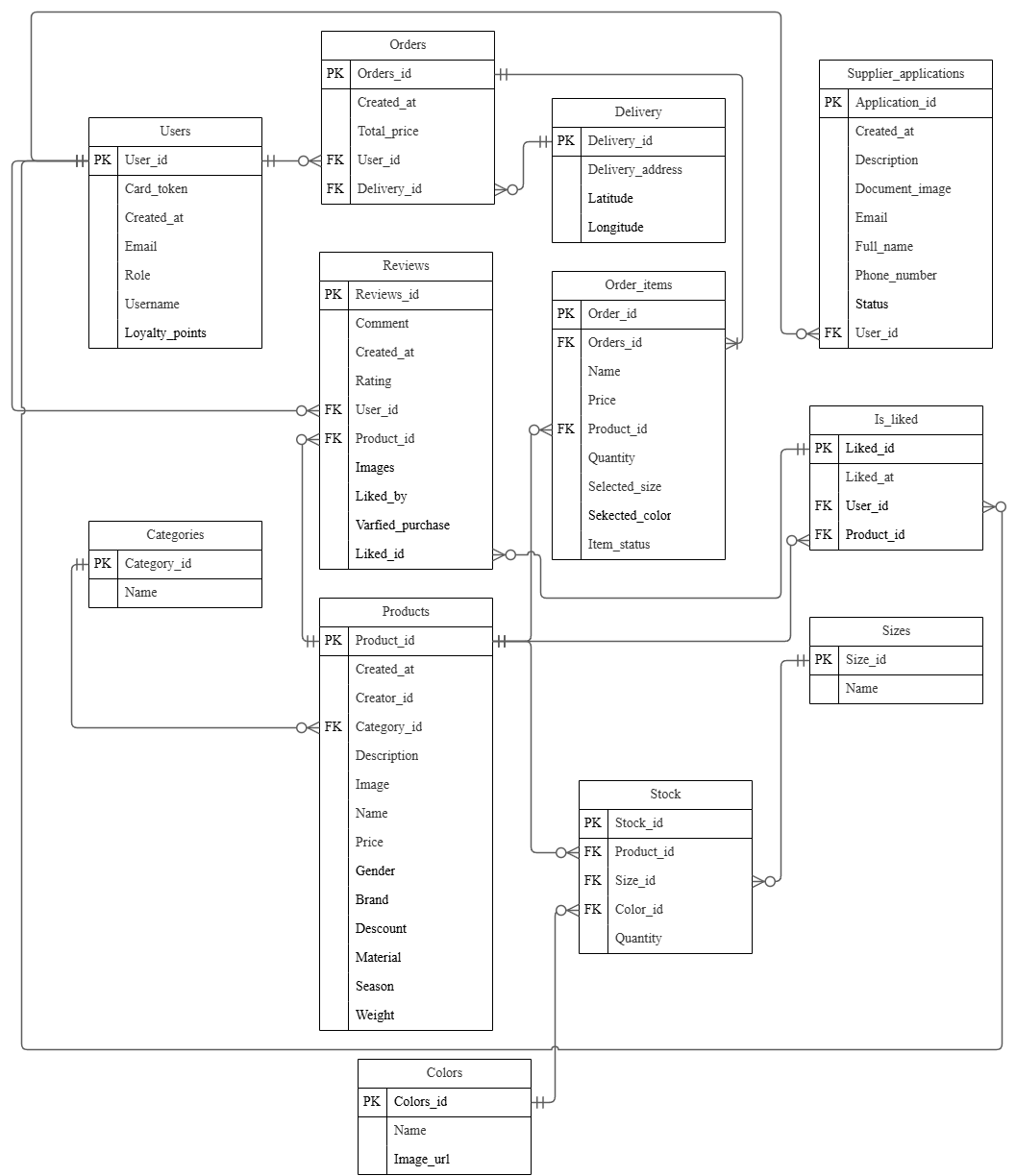


Рисунок 2.2 – Схема базы данных

Всего в базе данных 12 таблиц. Название и назначение каждой спроектированной коллекции приведено в таблице 2.1.

Таблица 2.1 – Таблицы базы данных

|  |  |
| --- | --- |
| Название таблицы | Назначение таблицы |
| Users | Хранит данные о пользователях |
| Orders | Данные о списке всех заказов |
| Products | Хранит информацию о товаре |
| Reviews | Хранит отзывы о товаре |
| Delivery | Хранит возможные места доставки товара |
| OderItems | Данные о каждом отдельном заказе |
| Categories | Хранит доступные категории товаров |
| SupplierAppliccat | Заявка на поставщика |
| IsLiked | Хранит лайки н товарах |
| Sizes | Хранит размеры товара |
| Stock | Хранит общую информацию о размерах, цветах и количестве |
| Colors | Хранит информацию о цвете |

Подробное описание каждой коллекции приведено в таблицах 2.2 – 2.8. А именно описаны типы и названия полей, а также приведено их описание и назначение.

Таблица 2.2 – Структура таблицы Delivery

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| DeliveryId | string | Уникальный идентификатор |
| DeliveryAddress | string | Точный адрес доставки |

В таблице Delivery ( таблица 2.2 ) содержится информация о возможных местах доставки товара. Благодаря данной таблице можно ознакомиться с пунктами доставки, а также содержит в себе внешний ключ для связи с таблицей Orders (таблица 2.7 ).

При создании товара создаются вспомогательные таблицы, которые по итогу вносятся в таблицу Stock ( таблица 2.3 ) и помогают хранить количество товаров, доступные цвета, размеры и количество.

Таблица 2.3 – Структура таблицы Stock

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| StockId | string | Уникальный идентификатор хранилища |
| ProductId | number | Уникальный идентификатор продукта |
| SizeId | string | Уникальный идентификатор размера |
| ColorId | string | Уникальный идентификатор цвета |
| Quantity | number | Количество |

Вся информация о товаре, хранится в таблице Products ( таблица 2.4 ). Благодаря данной таблице, продавцы могу добавлять, изменять или удалять информацию о товарах, тем самым всегда поддерживать актуальность в наличии или точной стоимости каждой позиции.

При создании товара, когда продавец вводит данные товара, они сохраняются в базе данных в таблице Products (таблица 2.4). Каждая строка в таблице Products представляет собой отдельный товар и содержит следующую информацию: идентификатор товара, дату создания товара, идентификатор категории товара, идентификатор создавшего товар, описание товара, изображение с товаром, название товара, цена, гендер товара, бренд, скидка, материал, сезон носки и вес.

Таблица 2.4 – Структура таблицы Products

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| ProductId | number | Уникальный идентификатор товара |
| CreatedAt | timestamp | Дата регистрации товара |
| CategoryId | number | Идентификатор категории |
| CreatorId | string | Идентификатор пользователя |
| Description | string | Дополнительная информация о товаре |
| Image | string | Изображение товара |
| Name | string | Полное название товара |
| Price | number | Цена товара |
| Gender | string | Для кого предназначен товар |
| Brand | string | Фирма товара |
| Descount | number | Скидка товара |
| Material | string | Материал товара |
| Season | string | Сезон носки |
| Weight | number | Вес товара |

Хранение данных в таблице Users ( таблица 2.5 ) обеспечивает возможность сохранения и извлечения информации о пользователях, что позволяет администратору просматривать общее количество зарегистрированных клиентов.

Таблица 2.5 – Структура таблицы Users

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| UserId | string | Уникальный идентификатор пользователя |
| CardToken | string | Токен карты клиента |
| CreatedAt | timestamp | Дата создания аккаунта |
| Email | string | Электронная почта пользователя |
| Role | string | Роль пользователя |
| Username | string | Никнейм пользователя |
| LoyaltyPoints | string | Баллы лояльности |

После оформления заказа, информация об идентификаторе пользователя, датой добавления товара, статусом заказа, общей суммой, идентификатор покупателя, адресом доставки и типом оплаты попадает в таблицу Orders (таблица 2.6).

Таблица 2.6 – Структура таблицы Sizes

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| SizeId | string | Уникальный идентификатор размера |
| Name | string | Размер товара |

В таблице Orders ( таблица 2.7 ) хранится информация о общем заказе.

Таблица 2.7 – Структура таблицы Orders

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| OrdersId | string | Уникальный идентификатор заказ(а/ов) |
| CreatedAt | timestamp | Дата и время заказа |
| TotalPrice | number | Общая сумма заказа |
| UserId | string | Статус заказа |
| DeliveryId | number | Адрес доставки заказа |

Также в таблице Reviews (таблица 2.8). хранится о отзывах, которые клиент загружает о приобретенном товаре.

Таблица 2.8 – Структура таблицы Reviews

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| ReviewsId | string | Уникальный идентификатор отзыва |
| Comment | string | Текст отзыва |
| CreatedAt | timestamp | Дата создания |
| Rating | number | Оценка товара от 0 до 5 |
| ProductId | number | Идентификатор товара |
| UserId | string | Идентификатор оставившего отзыв |
| Images | string | Изображение |
| LikedBy | string | Идентификатор пользователя |
| VarfiedPurch | boolean | Пользователь с покупкой |
| LikedId | string | Идентификатор лайка |

В таблице Categories ( таблица 2.9 ) представлена информация о категории товара, а именно его Id и название.

Таблица 2.9 – Структура таблицы Categories

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| CategoryId | number | Уникальный идентификатор категории товара |
| Name | string | Название категории |

В таблице isLiked ( таблица 2.10 ) хранится информация о лайкнувших определённый товар, а именно идентификатор лайка, время лайка, идентификатор пользователя и идентификатор продукта .

Таблица 2.10 – Структура таблицы IsLiked

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| LikedId | string | Уникальный идентификатор заказа |
| LikedAt | timestamp | Время лайка |
| UserId | string | Уникальный идентификатор |
| ProductId | number | Уникальный идентификатор товара |

В таблице OrderItems ( таблица 2.11 ) хранится информация о конкретном заказе, а именно идентификатор заказа, идентификатор заказов, название товара, цена, идентификатор товара, количество и его размер.

Таблица 2.11 – Структура таблицы OrderItems

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| OrderId | string | Уникальный идентификатор заказа |
| OrdersId | string | Идентификатор заказов |
| Name | string | Название товара |
| Price | number | Цена товара |
| ProductId | number | Идентификатор товара |
| Quantity | number | Количество товара |
| SelectedSize | string | Размер товара |
| ItemStatus | string | Статус товара |
| SelectedColor | string | Цвет товара |

В таблице Colors ( таблица 2.12 ) хранится информация о цветах, а также изображение товара в этом цвете.

Таблица 2.12 – Структура таблицы Colors

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| ColorsId | string | Уникальный идентификатор цвета |
| Name | string | Название цвета |
| ImageUrl | string | Ссылка на изображение |

В таблице SupplierApplications ( таблица 2.13 ) содержит информацию о заявках пользователей на становление поставщиком, а именно идентификатор заявки, дата создания, дополнительная информация, фото для документов, почта, ФИО, номер телефона, статус заказа и идентификатор оправителя.

Таблица 2.13 – Структура таблицы SupplierApplications

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| ApplicationId | string | Уникальный идентификатор заявки |
| CreatedAt | timestamp | Время создания |
| Description | string | Дополнительная информация |
| DocImage | string | Ссылка на фото с документами |
| Email | string | Почта пользователя |
| FullName | string | Полное ФИО пользователя |
| PhoneNumber | string | Номер телефона пользователя |
| Status | string | Статус заявки |
| UserId | string | Уникальный идентификатор пользователя |

По итогу мы получаем базу данных, которая соответствует как минимум первым трем формам нормализации.

## 2.3 Проектирование мобильного приложения

Для создания мобильного приложения был выбран язык программирования Dart и фреймворк Flutter [7]. Использование данного языка и фреймворка предоставляет ряд значимых преимуществ:

* высокая производительность благодаря компиляции в нативный код, что обеспечивает высокую скорость работы и быструю загрузку;
* встроенная поддержка асинхронного программирования, что позволяет управлять асинхронными задачами и повышает отзывчивость приложения;
* поддержка JIT компиляции для быстрой разработки и тестирования.

Фреймворк Flutter дополняет преимущества Dart следующими функциями:

* Hot Reload, который дает мгновенное применение изменений в коде и немедленный просмотр результата в режиме реального времени, что ускоряет процесс разработки и тестирования;
* использование готовых настраиваемых виджетов, которые следуют рекомендациям по дизайну Material Design (для Android) и Cupertino (для iOS), что позволяет быстро создавать качественные и стильные интерфейсы.

Кроме того, использование Dart и Flutter обеспечивает единообразие в разработке, поскольку Dart является основным языком программирования для Flutter, что упрощает командную работу и поддержку проекта. Подход Flutter к созданию интерфейса также позволяет создавать адаптивные приложения, которые могут автоматически адаптироваться под разные устройства и экраны. Благодаря активной и развивающейся сообществу разработчиков, Dart и Flutter постоянно обновляются и улучшаются, что обеспечивает доступ к новейшим технологиям и инструментам для создания высококачественных мобильных приложений.

Для создания мобильного приложения необходимо в терминале ввести команду «flutter create shop\_project», которая автоматически создает проект с разделением на папки и структурированным кодом.

Папка lib содержит файл main.dart, который является главным файлом приложения. Именно там начинается выполнение приложения. Остальные файлы в этой папке содержат коды страниц приложения. В каждом файле отдельная страница представлена в виде класса, в котором описывается разметка страницы, а также обработка всех функций и создание маршрутов для связи с сервером.

Папка test содержит тесты для приложения, где можно создавать файлы тестов для различных частей приложения, таких как виджеты или модели.

Папка build создается автоматически при сборке приложения и содержит артефакты сборки. Папка android содержит конфигурационные файлы и код для сборки приложения под Android. Папка ios содержит конфигурационные файлы и код для сборки приложения под iOS. Папка web содержит файлы для сборки приложения под веб. Здесь находятся файлы index.html, main.dart.js и другие ресурсы для веб-сборки. Папка macos содержит конфигурационные файлы и код для сборки приложения под macOS. Папка windows содержит конфигурационные файлы и код для сборки приложения под Windows. Папка linux содержит конфигурационные файлы и код для сборки приложения под Linux.

Файл pubspec.yaml используется для управления зависимостями проекта. Здесь указываются пакеты, которые используются в приложении, а также такие настройки, как название приложения, версия и ресурсы.

Файл README.md содержит описание проекта, инструкции по установке и использованию. В папке assets хранятся статические ресурсы приложения, такие как изображения, шрифты. Эти ресурсы указываются в файле pubspec.yaml.

Пример описанной архитектуры приведен на рисунке 2.3.

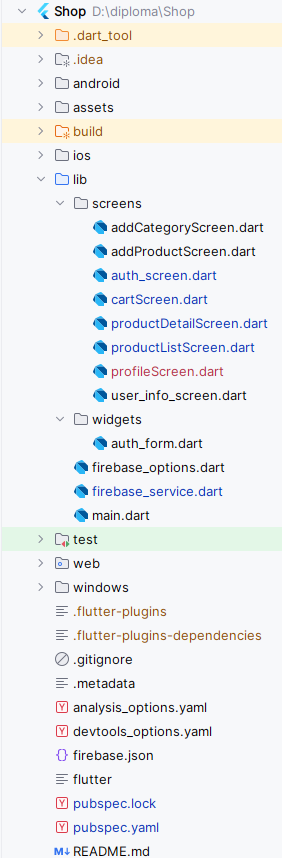


Рисунок 2.3 – Пример архитектуры файлов мобильного приложения

Страницы являются основополагающими компонентами приложения. На них ведет маршрутизация со стороны клиента. Каждая страница отвечает за отдельные функции, предназначенные именно ей.

В Flutter приложении страницы представляют собой виджеты, которые описывают пользовательский интерфейс для конкретного экрана. Эти виджеты могут включать в себя различные элементы: текст, изображения, кнопки и формы.

Маршрутизация между страницами осуществляется с помощью навигационных методов, таких как Navigator.push и Navigator.pop, которые позволяют перемещаться вперед и назад между страницами.

Каждая страница в приложении может иметь свою собственную логику и состояние, управляемое с помощью различных методов управления состоянием, таких как StatefulWidget, Provider или Bloc.

Страницы также могут взаимодействовать друг с другом, передавая данные через параметры маршрутизации или используя глобальные состояния.

Страницы были разделены на следующие логические сущности:

* addCategoryScreen – страница добавления категории;
* addProductScreen – страницы добавления товара;
* authScreen – страница авторизации;
* cartScreen – страница корзины;
* productDetailScreen –страница с товаром;
* productListScreen – страница со списком товаров;
* profileScreen – страница профиля;

Такое большое количество страниц было необходимо для обеспечения максимального комфорта и интуитивного взаимодействия пользователя с мобильным приложением. А также для разбиения классов, что удобно при написании кода, когда нужно обратиться к определенному классу. Всегда удобно, когда на определенную функцию приложения есть своя страница.

## **2.4 Диаграммы и блок-схемы приложения**

В данном подразделе будут представлены такие блок-схемы и диаграммы как:

* диаграмма вариантов использования;
* диаграмма компонентов;
* блок-схема оформления заказа.

### **2.4.1 Диаграмма вариантов использования**

В диаграмме вариантов использования, отображено разделение возможностей администратора, пользователя, поставщика и продавца.

Диаграмма вариантов использования администратора обозначена на рисунке 2.5.

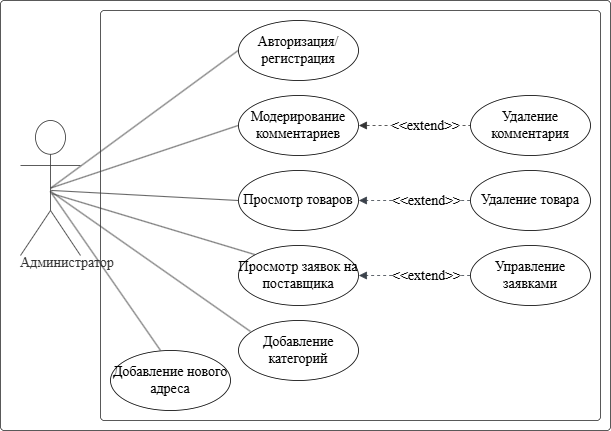


Рисунок 2.4 – Диаграмма вариантов использования администратора

Администратор может управлять заказами и их статусами, а также отзывами. Диаграмма вариантов использования пользователя представлена на рисунке 2.6.

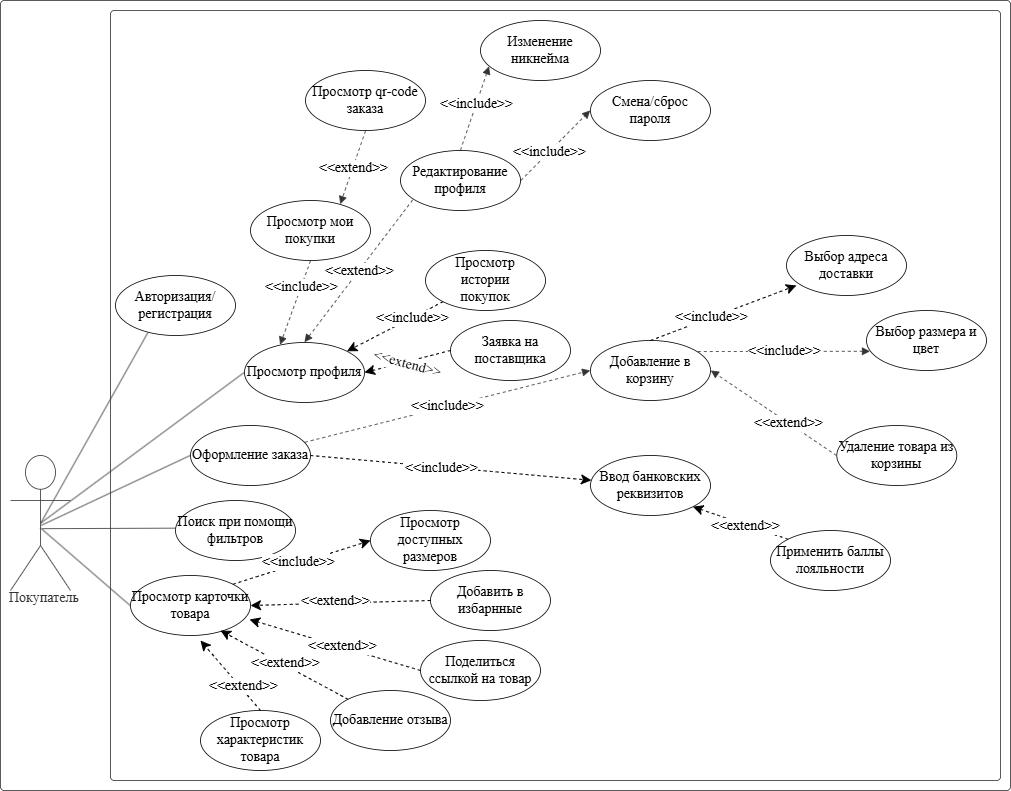


Рисунок 2.5 – Диаграмма вариантов использования пользователя

Пользователь может осуществлять поиск и добавлять необходимые ему товары в корзину, после чего оформлять заказ добавленных в корзину товаров, просматривать заказы.

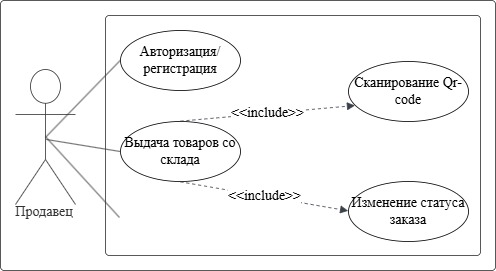


Рисунок 2.5 – Диаграмма вариантов использования продавца

Продавец может осуществлять редактирование и создания собственного товара.



Рисунок 2.6 – Диаграмма вариантов использования поставщика

Поставщик может осуществлять создание и редактирование собственных товаров, а также их удаление.

### **2.5.1 Диаграмма развертывания**

Диаграмма развертывания помогает визуализировать архитектуру системы и понять, как различные компоненты взаимодействуют друг с другом.

Для разработки мобильного приложения на основе операционной системы Android был задействован язык Dart. В качестве БД использовался Firebase Database.

Диаграмма развертывания представлена на рисунке 2.7.

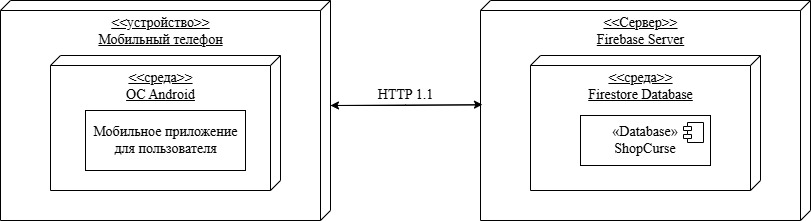


Рисунок 2.6 – Диаграмма развертывания

Цель диаграммы развертывания – предоставить обзорную картину архитектуры системы, выделить основные компоненты и их взаимодействие, а также помочь разработчикам и архитекторам понять и изменять систему.

### **2.5.2 Блок-схема оформления заказа из заведения**

Блок-схема является мощным инструментом, который позволяет визуализировать и упорядочить последовательность действий, потоки данных, принятие решений, ветвления и циклы в компактной и понятной форме. Она представляет собой графическое изображение, которое помогает лучше понять сложные процессы.

В блок-схеме каждое действие или компонент представлен прямоугольным блоком, в котором четко указано его название или описание. Эти блоки соединены линиями, которые обозначают потоки данных, управления или связи между компонентами системы. Линии могут иметь различные формы и типы стрелок, что позволяет указать на разные виды связей и направлений потока информации.

Использование блок-схемы позволяет не только визуализировать процесс, но и выявить потенциальные ошибки и исключения, что, в свою очередь, способствует повышению надежности системы. Благодаря этому разработчики могут заранее обнаружить слабые места в алгоритме и предусмотреть возможные проблемы.

Кроме того, блок-схема значительно облегчает коммуникацию между членами команды. Все участники проекта могут быстро и четко понять сложные процессы, что способствует более эффективному взаимодействию и совместной работе. Это особенно важно в командах с разными уровнями опыта и понимания.

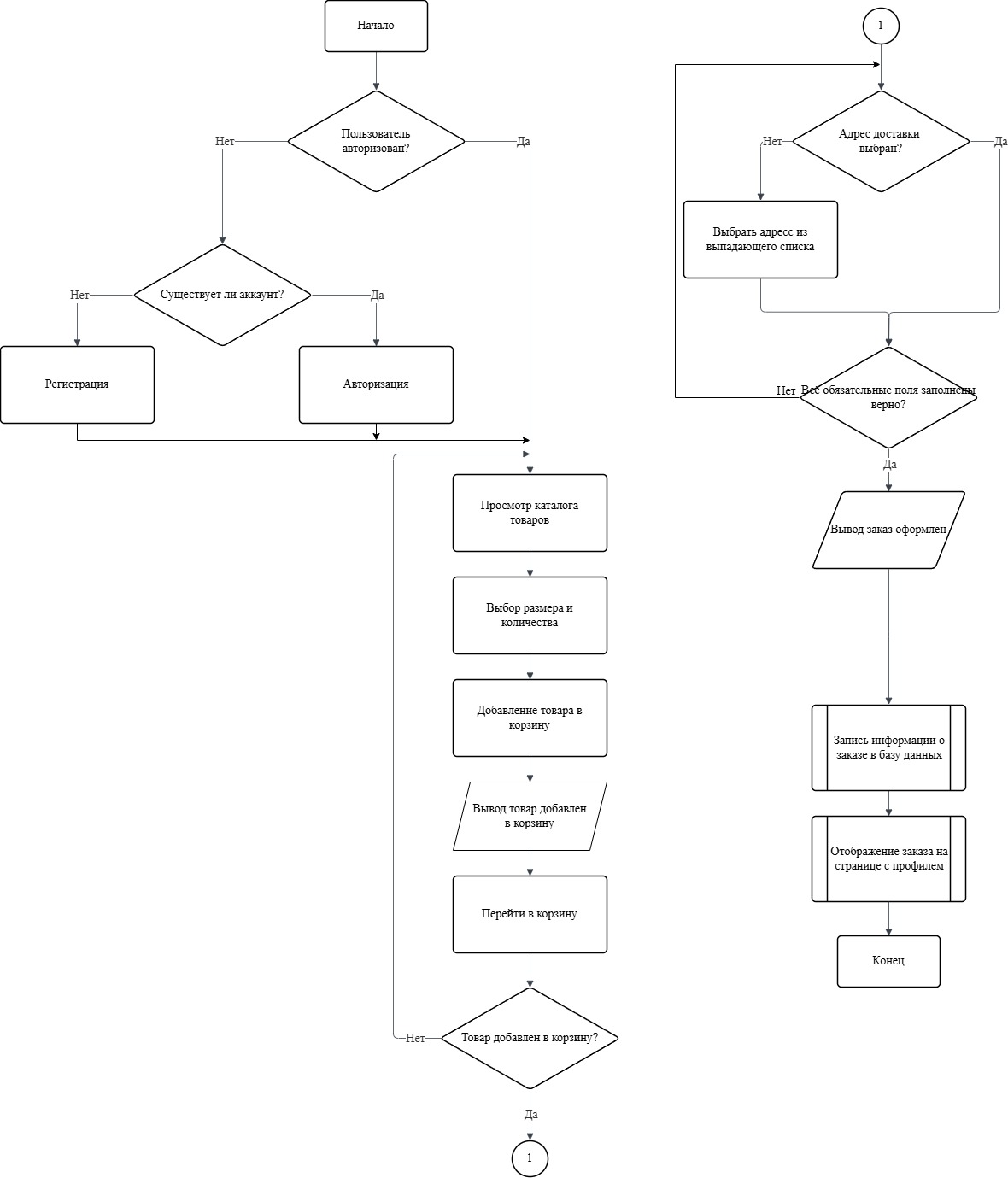


Рисунок 2.8 – Блок-схема оформления заказа

Блок-схема является удобным инструментом для анализа и проектирования алгоритмов программ, описания бизнес-процессов, планирования задач и разработки архитектуры систем.

### **2.5.3 Блок-схема смены пароля**

Также на рисунке 2.8 представлена блок-схема смены пароля пользователям. Которая отражает все возможные способы смены пароля с различными статусами пользователя.

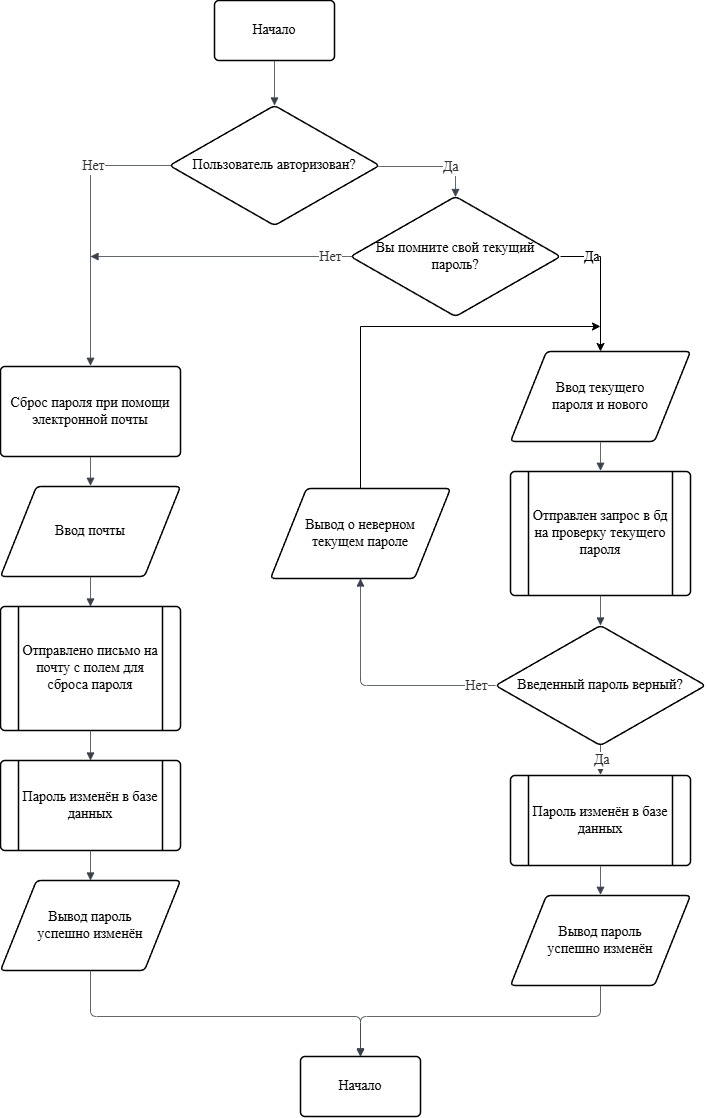


Рисунок 2.9 – Блок-схема смены пароля

Смена пароля одна из важных составляющих для сохранения конфиденциальности и безопасности пользователя приложения.

## 2.7 Выводы по разделу

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения. По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения, была также спроектирована база данных.

# 3 Реализция приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

.

.

.

.

Утв.

3 Реализция приложения

Лит.

Листов

15

*-, 2025*

У

В этом разделе будет рассмотрена реализация приложения, охватывающая серверную часть, клиентскую, а также взаимодействие между ними.

Реализация приложения проводилась в среде Android Studio 2024. Серверная часть реализована на языке программирования JavaScript с использованием среды выполнения Node.js. Для размещения сервера выбран хостинг Render, что позволяет быстро развернуть и масштабировать приложение при необходимости.

Клиентская часть приложения разработана в среде Android Studio с использованием языка программирования Dart и фреймворка Flutter. Такой выбор технологий позволил создать современный, удобный и отзывчивый пользовательский интерфейс для мобильных устройств на базе Android.

## 3.1 Разработка серверной части

На первый взгляд может показаться, что если проект построен вокруг облачного Firebase, то отдельная серверная часть не требуется. Однако на практике современное мобильное приложение — это больше, чем просто обёртка вокруг базы данных.

Серверная логика позволяет реализовать сложные сценарии, которые либо неудобны, либо невозможны только с помощью клиента и Firebase. Сервер даёт возможность централизованно управлять сложными процессами, отправлять кастомные push-уведомления, выполнять роль защиты с использование секретный ключей а также реализовать умные редиректы и обратку глубоких ссылок для мобильного приложения.

Когда администратор работает в мобильном приложении и видит список заявок от пользователей, которые хотят получить статус поставщика, он может просмотреть каждую заявку и изменить её статус — например, одобрить или отклонить. Для этого в интерфейсе приложения реализован специальный выпадающий список (или кнопки), с помощью которого администратор выбирает новый статус заявки.

Как только администратор выбирает новый статус (например, «Одобрено»), приложение формирует специальный HTTP POST-запрос на серверную часть. Сервер принимает этот запрос и начинает последовательное выполнение. Если новый статус одобрено, то сервер используя push-токен формирует персонализированное уведомление.

Часть кода использующего для уведомлений можно увидеть в листинге 3.1 .

if (newStatus === "approved") {

await admin.firestore().collection("users").doc(userId).update({ role: "Supplier" });}

const userDoc = await admin.firestore().collection("users").doc(userId).get();

const pushToken = userDoc.get("push\_token");

if (pushToken) {

let title, body;

if (newStatus === "approved") {

title = "Ваша заявка одобрена!";

body = "Поздравляем, вы стали поставщиком!";

} else if (newStatus === "rejected") {

title = "Ваша заявка отклонена";

body = "К сожалению, ваша заявка не прошла модерацию.";

} else {

title = "Статус вашей заявки обновлён";

body = "Проверьте статус в приложении.";

}

const message = {

token: pushToken,

notification: { title, body },

data: { status: newStatus },

};

await admin.messaging().send(message);

console.log(`Push sent to ${userId}: ${title}`);

}

Листинг 3.1 – Код для push-уведомлений

Сервер также используется для корректной работы кнопки поделиться, при нажатии кнопки поделиться формируется URL, при переходе на который открывается красивая страница и нас переадресует сразу в приложение на тот товар, которым поделились с нами. Если приложения нет, то откроется Google Play и предложит скачать его.

Пример генерации ссылки предложен в листинге 3.2 .

function isIOS() {

return /iphone|ipad|ipod/i.test(navigator.userAgent);

}

var productId = "${productId}";

var deepLink = "myapp://product?productId=" + productId;

if (!isIOS()) {

setTimeout(function() { window.location = deepLink; }, 450);

setTimeout(function() { window.location = "https://play.google.com/store/apps/details?id=com.example.shop"; }, 2200);

}

document.getElementById("openAppBtn").onclick = function(e) {

if (!isIOS()) {

window.location = deepLink;

setTimeout(function() {

window.location = "https://play.google.com/store/apps/details?id=com.example.shop";

}, 1800);}};

Листинг 3.2 – Код для генерации ссылок

Данный подход был выбран в связи с тем, что сервер может быть легко расширен для использования других сервисов, а также более безопасен для использования приватных ключей.

## 3.2 Взаимодействие с базой данных

Современные мобильные приложения требуют надёжного, масштабируемого и удобного механизма хранения и обмена данными между пользователями и сервером. Для решения этих задач в рамках моего дипломного проекта был выбран облачный сервис Firebase от Google, который предоставляет целый набор инструментов для аутентификации, хранения данных, обмена сообщениями и других типовых задач мобильной разработки. Firebase обладает рядом преимуществ: он не требует собственной инфраструктуры на старте, легко масштабируется, предоставляет мощные SDK для Flutter, а также гарантирует безопасность и скорость работы даже при большом количестве пользователей.

dependencies:

flutter:

sdk: flutter

cloud\_firestore: ^5.6.5

firebase\_core: ^3.12.1

firebase\_auth: ^5.5.1

firebase\_storage: ^12.4.4

firebase\_messaging: ^15.2.5

google\_sign\_in: ^6.3.0

shared\_preferences: ^2.5.2;

Листинг 3.3 – Пакеты для работы с Firebase

На листинге 3.3 можно увидеть пакеты которые использовались для работы с сервисом Firebase.

Для структурирования кода и удобства дальнейшей поддержки вся работа с Firebase вынесена в отдельный сервисный класс — FirebaseService. Это не просто набор функций, а полноценный singleton, то есть единый экземпляр сервиса, который используется во всём приложении. Такой подход гарантирует централизованный и последовательный доступ к базе данных и средствам авторизации.

class FirebaseService

{

static final FirebaseService \_singleton = FirebaseService.\_internal();

factory FirebaseService() => \_singleton;

FirebaseService.\_internal();

final auth = FirebaseAuth.instance;

final firestore = FirebaseFirestore.instance;

}

Листинг 3.4 – Объявление класса FirebaseService

В современном мобильном приложении, особенно если речь идёт о магазине, защита пользовательских данных и удобство входа — необходимые условия успеха. В моём проекте реализована многоуровневая система аутентификации на базе сервисов Firebase, что обеспечивает как безопасность, так и гибкость для пользователя.

Реализация вынесена в отдельный сервисный класс FirebaseService. Такой подход позволяет не дублировать логику и централизовать все процессы, связанные с аккаунтом пользователя — от регистрации до выхода.

Регистрация — первый шаг к персонализации работы в приложении. Пользователь вводит email, пароль и имя. Вся логика работы с аккаунтами реализована через сервис Firebase Auth, а профиль пользователя сохраняется в Firestore.

Код функцией регистрации представлен на листинге 3.5 .

Future<void> onRegister({

required String email,

required String password,

required String username,

String role = "user"

}) async {

final credential = await auth.createUserWithEmailAndPassword(

email: email,

password: password,

);

String userId = credential.user!.uid;

await firestore.collection('users').doc(userId).set({

'username': username,

'email': email,

'role': role,

'created\_at': FieldValue.serverTimestamp(),

'loyalty\_points': ''

});

await savePushToken(userId);

}

Листинг 3.5 – Функция регистрации

После регистрации пользователь может сразу войти в систему и пользоваться всеми возможностями магазина.

Для максимального удобства пользователей реализована авторизация через Google. Этот способ особенно удобен тем, что избавляет человека от необходимости придумывать и запоминать новый пароль для ещё одного сервиса. Всё, что требуется — выбрать свой Google-аккаунт и подтвердить действие. Процесс занимает буквально несколько секунд и не требует никаких дополнительных усилий: ни подтверждения почты, ни заполнения длинных форм. Пользователь нажимает на кнопку «Войти через Google» в приложении. Следом открывает стандартный диалог Google, где пользователь может выбрать любой из своих аккаунтов. Если пользователь уже входил ранее и выбрал действие запомнить меня, то вход выполнится автоматически без ожиданий действий от пользователя. Код с функцией входа через Google-аккаунт представлен на листинге 3.6 .

Future<void> signInWithGoogle(BuildContext context) async {

final GoogleSignInAccount? googleUser = await googleSignIn.signIn();

if (googleUser == null) return;

final GoogleSignInAuthentication googleAuth = await googleUser.authentication;

final credential = GoogleAuthProvider.credential(

accessToken: googleAuth.accessToken,

idToken: googleAuth.idToken,

);

final UserCredential userCredential = await auth.signInWithCredential(credential);

final User? user = userCredential.user;

if (user != null) {

final DocumentSnapshot userDoc = await firestore.collection('users').doc(user.uid).get();

if (!userDoc.exists) {

await firestore.collection('users').doc(user.uid).set({

'username': user.displayName ?? 'Google User',

'email': user.email,

'role': 'user',

'created\_at': FieldValue.serverTimestamp(),

'loyalty\_points': ''

});

}

await savePushToken(user.uid);

await rememberUser(true);

Navigator.pushReplacementNamed(context, '/products');

}

}

Листинг 3.6 – Функция входа через Google

Работает это по такому принципу, что если у пользователя уже есть аккаунт, то происходит авторизацию, а если заходит впервые, то будет создавать новый профиль.

void onListenUser(void Function(User?)? doListen) {

auth.authStateChanges().listen(doListen);

}

Листинг 3.7 – Функция отслеживания статуса пользователя

При запуске приложения слушатель отслеживает, авторизован ли пользователь, и если нет — перенаправляет на экран входа. Благодаря методу onListenUser приложение всегда в курсе текущего состояния сессии.

## 3.3 Разработка клиентской части

При создании клиентской части приложения особое внимание уделялось не только технической реализации, но и удобству пользовательского опыта. Основная цель — сделать так, чтобы взаимодействие с магазином было интуитивным, современным и приятным независимо от возраста и опыта пользователя.

Интерфейс построен по принципу "mobile-first". Для оформления выбран тёмный стиль с акцентными цветами, что не только придаёт приложению современный внешний вид, но и способствует меньшей утомляемости глаз при длительной работе. В ходе проектирования дизайна были изучены популярные мобильные приложения в сфере e-commerce, чтобы учесть лучшие решения и создать свой узнаваемый стиль.

Основной экран приложения — это каталог товаров. Пользователь сразу видит витрину с удобной поисковой строкой, фильтрами по категориям, цветам, брендам, размеру, сезону и другим параметрам.

Реализован быстрый поиск по названию и описание товара, а также возможность гибкой фильтрации, чтобы найти нужный товар среди сотен позиций.

final filteredProducts = products.where((product) {

final matchesCategory = \_selectedCategory == null || product['category'] == \_selectedCategory;

final matchesColor = \_selectedColor == null || product['color'] == \_selectedColor;

final matchesPrice = product['price'] >= \_priceRange.start && product['price'] <= \_priceRange.end;

return matchesCategory && matchesColor && matchesPrice;

}).toList();

Листинг 3.8 – Фильтрация по нескольким условиям

На рисунке 3.1 продемонстрировано как выглядит фильтрация в самом приложении.

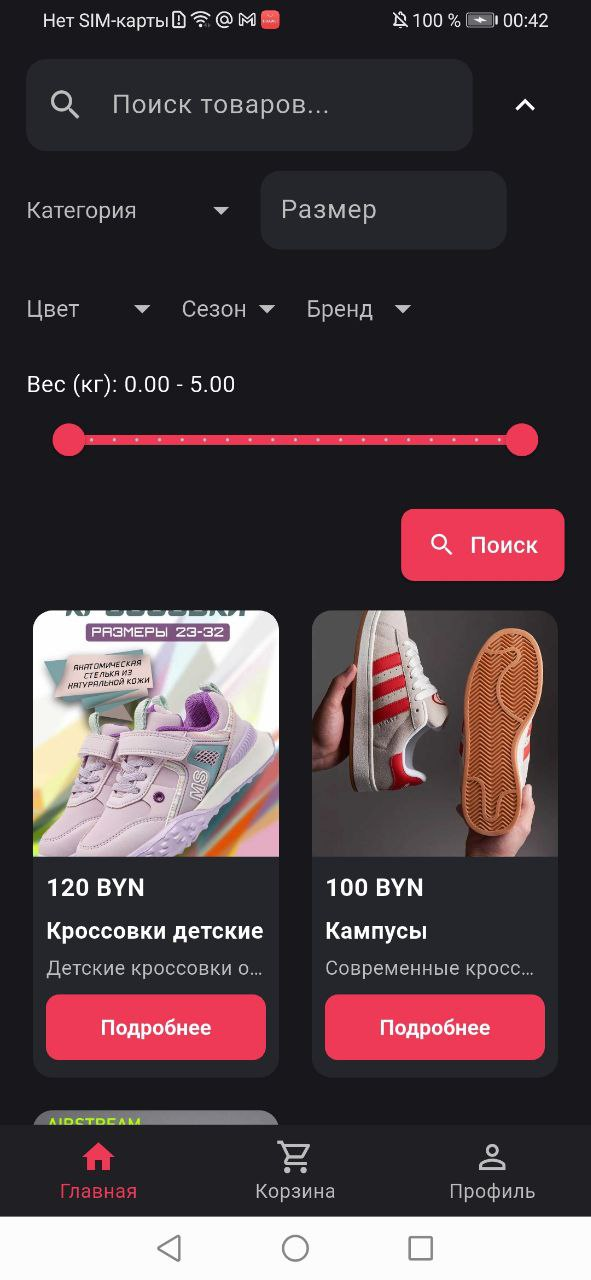


Рисунок 3.1 – Блок-схема смены пароля

Фильтрация происходит локально после загрузки данных, что обеспечивает моментальный отклик интерфейса.

После оформления заказа необходимо актуализировать остатки товаров, чтобы другие пользователи видели только реально доступные позиции. Для этого реализована функция обновления количества товара по выбранному размеру и цвету. Она вызывается при каждом успешном заказе или возврате, и работает напрямую с базой данных Firestore.

Future<void> updateProductStock(

int productId, String size, String color, int quantityChange) async {

final productQuery = await firestore

.collection('products')

.where('product\_id', isEqualTo: productId)

.limit(1)

.get();

if (productQuery.docs.isNotEmpty) {

final productDoc = productQuery.docs.first;

final productRef = productDoc.reference;

final productData = productDoc.data() as Map<String, dynamic>;

final sizes = productData['sizes'] as Map<String, dynamic>?;

if (sizes != null && sizes.containsKey(size)) {

final sizeData = sizes[size] as Map<String, dynamic>;

final colorQuantities = sizeData['color\_quantities'] as Map<String, dynamic>?;

if (colorQuantities != null && colorQuantities.containsKey(color)) {

final currentStock = colorQuantities[color] as int;

final newStock = currentStock + quantityChange;

if (newStock >= 0) {

await productRef.update({

'sizes.$size.color\_quantities.$color': newStock,

});

}

}

}

}

}

Листинг 3.9 – Обновление остатков товара

Данная функция находит нужный товар по его ID, далее — нужный размер и цвет внутри структуры в базе, и изменяет значение количества (например, уменьшает на 1 при покупке или увеличивает при возврате). Благодаря этому остатки всегда актуальны — и пользователь не сможет заказать больше, чем есть на складе. Также есть реализация в случае отказа от товара.

## 3.4 Выводы по разделу

В ходе разработки клиентской части приложения был реализован целый комплекс функций, обеспечивающих высокий уровень удобства, безопасности и интерактивности для конечного пользователя. Применение современного фреймворка Flutter позволило создать кроссплатформенное мобильное приложение с отзывчивым и адаптивным интерфейсом, актуальным для устройств разных размеров.

Использование облачных сервисов Firebase обеспечило быстрый обмен данными, надёжную аутентификацию и хранение пользовательской информации, а также гибкое масштабирование без необходимости поддержки собственной инфраструктуры. Вся работа с Firebase была вынесена в отдельный сервисный класс, что упростило сопровождение кода и позволило централизовать логику работы с данными и авторизацией.

Особое внимание было уделено таким ключевым функциям, как:

* Мгновенный поиск и фильтрация товаров;
* Динамическое обновление остатков на складе;
* Гибкая система аутентификации;
* Поддержка отзывов с фото и лайками;
* Интеграция с push-уведомлениями и deep links.

В результате клиентская часть приложения получилась не только функциональной и современной с точки зрения пользовательского опыта, но и гибкой для последующего развития. Такой подход обеспечивает конкурентоспособность решения на рынке мобильных приложений и позволяет быстро внедрять новые бизнес-функции по мере роста продукта.

# 7 Анализ информационной безопасности приложения

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Ероховец И.А.*

Пров.

Уласевич Н.И.

Н. контр.

.

.

.

.

.

Утв.

7 Анализ информационной безопасности приложения

Лит.

Листов

15

*-, 2025*

У

Важным аспектом при разработке мобильного приложения выступает обеспечение информационной безопасности. Приложение должно защищать данные пользователей от несанкционированного доступа, утечек и взлома.  
В рамках анализа безопасности моего приложения были рассмотрены следующие ключевые направления:

* защита пользовательских данных: реализация мер по защите аккаунтов, личной информации, истории заказов;
* разграничение прав пользователей: внедрение ролевой модели доступа к функциям приложения;
* регулярное обновление и контроль актуальности данных: обеспечение целостности информации на клиенте и сервере.

## 7.1 Защита пользовательских данных

Для хранения и передачи конфиденциальных данных пользователей применяется облачный сервис Firebase, отличающийся высоким уровнем безопасности и современными средствами шифрования.

Пароли и аутентификационные данные никогда не сохраняются в открытом виде — для хранения и проверки используется сервис Firebase Auth, где все пароли проходят хэширование с использованием современных криптоалгоритмов.  
Данные профиля, история заказов, бонусные баллы хранятся в базе данных Firestore, где доступ возможен только для авторизованных пользователей через защищённые каналы передачи данных, протокол HTTPS и встроенное шифрование на стороне Firebase.

Для дополнительной защиты реализованы следующие меры:

* возможность восстановления доступа только по подтверждённому email;
* автоматическая проверка безопасности паролей при регистрации.

Вся передача данных между приложением и сервером осуществляется исключительно по защищённому протоколу HTTPS, что исключает возможность перехвата данных злоумышленниками.

Файлы и изображения пользователей загружаются во внешнее облачное хранилище с использованием временных защищённых токенов доступа.

Также в приложении реализована система автоматического обновления токенов доступа и регулярная сверка состояния пользователя с базой данных. При обнаружении подозрительной активности осуществляется принудительный выход из аккаунта и оповещение пользователя.

## 7.2 Ограничение прав доступа пользователей

Для ограничения прав доступа пользователей реализована ролевая модель (RBAC — Role-Based Access Control). В приложении предусмотрены следующие основные роли:

* **Пользователь** имеет доступ к поиску, покупкам, просмотру и редактированию личных данных, отзывам.
* **Поставщик** имеет доступ к добавлению и редактированию собственных товаров.
* **Администратор** имеет расширенные функции: модерация отзывов, управление заявками поставщиков, добавление новых категорий и адресов.
* **Продавец** получает доступ к изменению статуса товара.

Управление доступом реализовано на двух уровнях:

* На клиенте скрываются элементы интерфейса, не предназначенные для текущей роли;
* На сервере и в Firebase Security Rules проводится дополнительная проверка прав при попытке выполнения чувствительных операций (например, добавление товара или модерация заявок).

Это позволяет минимизировать риски несанкционированного доступа к административным функциям и личным данным других пользователей.

## 7.3 Выводы по разделу

В данном разделе были рассмотрены и реализованы основные меры по обеспечению информационной безопасности мобильного приложения.

Использование технологий Firebase и Node.js позволило обеспечить высокий уровень защиты данных пользователей и устойчивость к основным видам атак.

Внедрение ролевой модели (RBAC) позволило гибко управлять правами доступа к различным функциям приложения, а централизованная обработка авторизации и проверка прав на сервере гарантируют сохранность и конфиденциальность пользовательских данных. Все эти меры в комплексе делают приложение надёжным и безопасным для конечных пользователей.

Список использованных источников

1. Главная страница сайта Lamoda [Электронный ресурс] / lamoda.ru– Режим доступа https://www.lamoda.ru/men-home/?sitelink=topmenuM – Дата доступа: 30.03.2025.
2. Главная страница сайта Mark Formelle [Электронный ресурс] / markformelle.by – Режим доступа https://markformelle.by/magaziny/– Дата доступа: 31.03.2025.
3. Главная страница сайта Ostin [Электронный ресурс] / ostin.com – Режим доступа: https://ostin.com/?utm\_referrer=https%3A%2F%2Fwww.google.com%2F/– Дата доступа: 25.03.2025.
4. Главная страница сайта Android Studio [Электронный ресурс] / developer.android.com – Режим доступа https://developer.android.com/studio/intro – Дата доступа: 15.03.2025.
5. Android documentation for app developers. The activity lifecycle [Электронный ресурс] / developer.android.com – Режим доступа: <https://developer.android.com/guide/components/activities/activity-lifecycle> – Дата доступа: 11.03.2025.
6. Введение в язык Dart [Электронный ресурс] / Metanit – 2025. – Режим доступа: https://metanit.com/dart/tutorial/1.1.php – Дата доступа: 20.03.2025.