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**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

**«Мобильное приложение «Интернет-магазин»»**

Выполнил студент Ритчик Дарья Сергеевна

(Ф.И.О.)

Руководитель проекта асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

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

Консультант: асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Нистюк О. А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2023

**Содержание**

ВВЕДЕНИЕ 3

1. Постановка задачи и аналитический обзор литературы 4

1.1 Приложение «Wildberries» 4

1.2 Приложение «Ozon» 5

1.3 Выводы по главе «Обзор аналогов приложений» 7

2. Проектирование информационной системы 8

2.1 Локальная и глобальная база данных 8

2.2 Проектирование мобильного приложения 14

2.3 Проектирование веб-сервера 15

2.4 Вывод по главе «Проектирование информационной системы» 16

3. Разработка информационной системы 17

3.1 Разработка базы данных 17

3.2 Разработка мобильного приложения 18

3.3 Разработка веб-приложения 19

3.4 Вывод по главе «Разработка информационной системы» 21

4. Обеспечение безопасности информационной системы 22

Заключение 23

Список использованных источников 24

Приложение А 25

Приложение B 26

# ВВЕДЕНИЕ

С каждым годом интернет-технологии занимают все более важное место в повседневной жизни людей. Сейчас подавляющее большинство магазинов различных направлений имеют свои онлайн-версии. Это позволяет наиболее эффективным образом продавать и доставлять товары, а также удобно хранить данные о продажах. В сети можно купить практически все, но остается множество магазинов, которые только начинают свой онлайн-путь. В интернет-магазине, разработанном в качестве курсового проекта, можно приобрести практически всё – начиная от книг и предметов роскоши, заканчивая одеждой.

Мобильное приложение будет сделано с помощью Android Studio на языке Dart и использовании фреймворка Flutter.Сервер написан на языке Node.js, в качестве базы данных использована PostgreSQL.

Оплату в приложении можно легко произвести по карте – такую возможность даёт встроенный Google pay.

# Постановка задачи и аналитический обзор литературы

## 1.1 Приложение «Wildberries»

Важным этапом создания различных проектов является ознакомление с разных литературных источников и аналитический разбор.

Были проанализированы несколько разных программных средств связанных с моей тематикой.

Приложение Wildberries (рис. 1.1) позволяет совершать интернет-покупки одежды, обуви и аксессуаров. У магазина Wildberries есть и свое приложение для смартфонов, которое очень удобно использовать. Самый большой плюс этого приложения – это простой и понятный интерфейс. В плане навигации и поиска товаров, приложение Wildberries оставляет только положительные эмоции. Большой ассортимент товаров, простая и понятная фильтрация, наличие акций и скидок – все это делает приложение Wildberries одним из лучших в виртуальном мире.

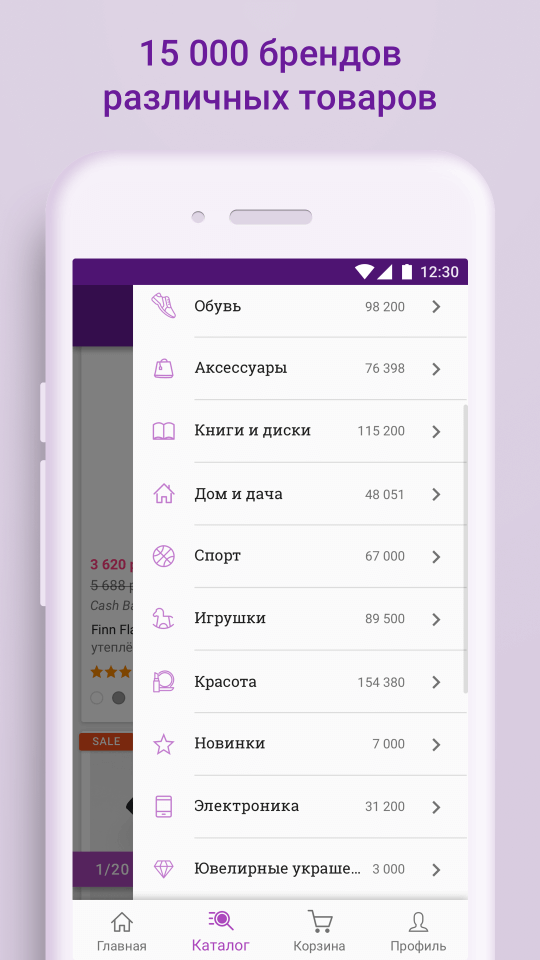


Рис. 1.1 – Мобильное приложение.

Данное приложение так же имеет и десктопную версию, которая показана на рисунке ниже (рисунок 1.2)

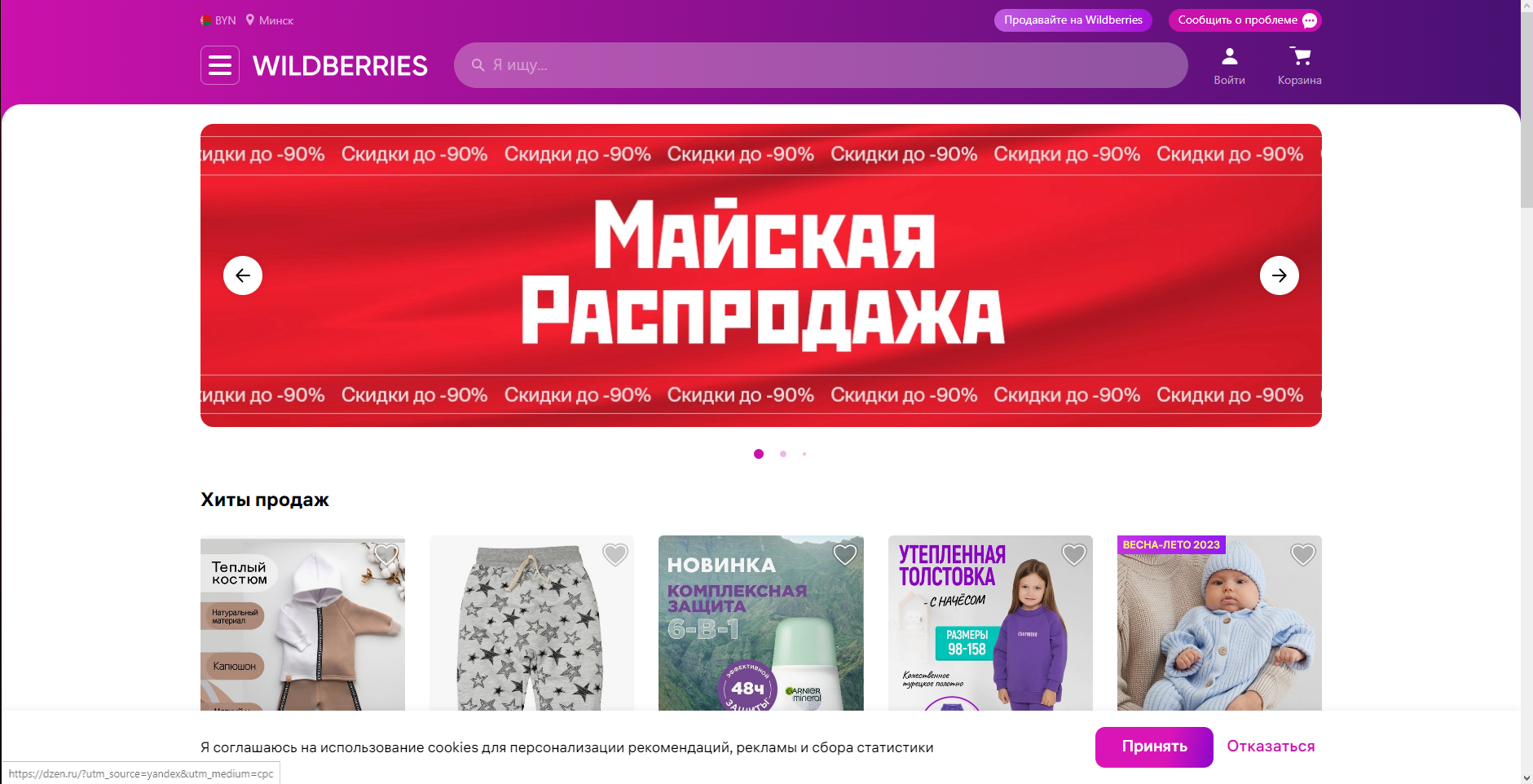


Рис. 1.2 – Версия для компьютеров.

Дизайн выполнен аналогично мобильной версии, в той же цветовой палитре.

## 1.2 Приложение «Ozon»

Приложение «Ozon».

Это многофункциональный интернет-магазин, который предлагает широкий ассортимент товаров, начиная от бытовой техники, продуктов питания, книг и заканчивая парфюмерией и косметикой. Один из недостатков данного интернет-магазин – это не слишком удобный интерфейс.Однако на Ozon можно быстро найти то, что вам нужно, настроив при этом фильтры так, как вы считаете удобным. Также на сайте и в приложении Ozon есть множество акций и скидок, которые могут сохранить ваш бюджет. Удобной функцией является возможность оплаты заказа при получении товара и приложение может работать при разных ротациях с разными разрешениями экрана.

Далее будут рассмотрены некоторые страницы данного приложения (Рис. 1.3 и 1.4) именно с этого приложения было почерпнуто вдохновение на реализацию списка категорий и просмотра информации в соответствии с ним. Список является исчерпывающим, и позволяет пользователю быстрее дойти до конечного пункта своих поисков.

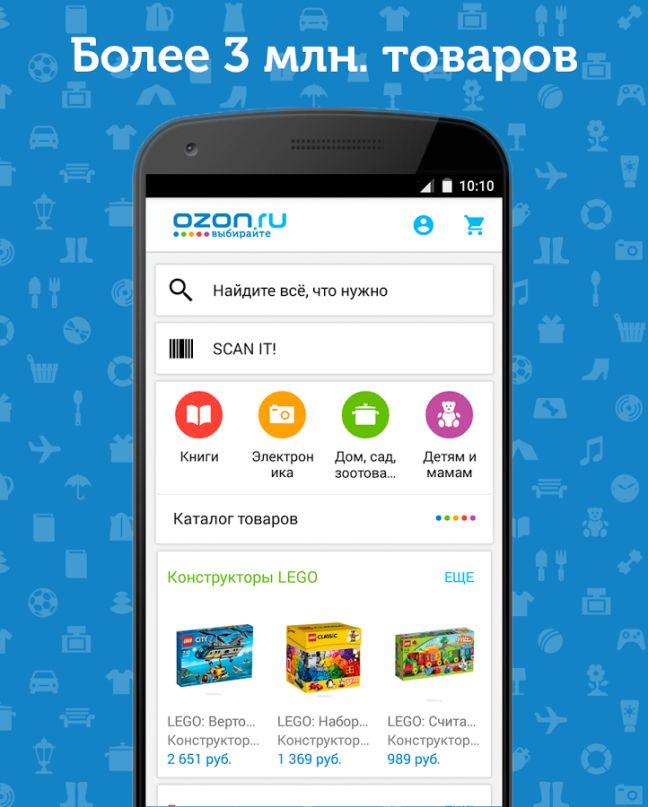


Рис. 1.3 – Ozon – дизайн под телефон.

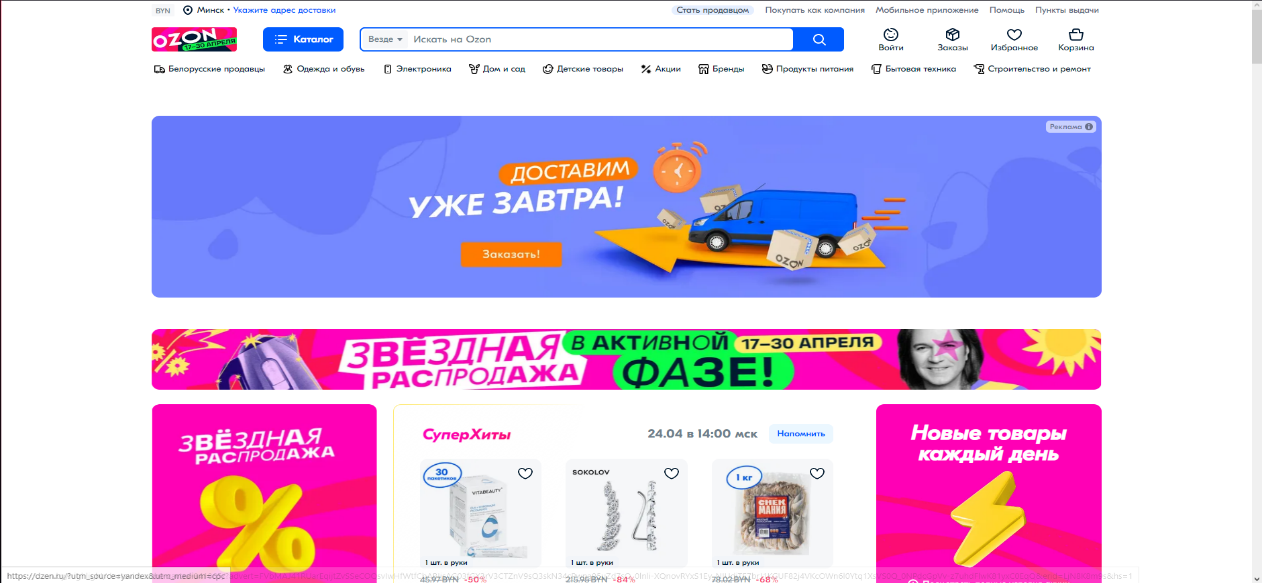
Как можно заметить десктопная версия значительно уступает мобильной – неудачные цветовые решения не цепляют глаз.

Рис. 1.4 – Интерфейс пользователя и администратора.

В моём приложении товары будут удобно разбиты по категориям, что позволит легко и быстро найти необходимые позиции.

## Выводы по главе «Обзор аналогов приложений»

В данной главе были показаны два популярных приложения, которые являются аналогами приложения «DoRi Mall»: «Wildberries» и «Ozon». В результате подытожим сравнения.

Wildberries позволяет с легкостью найти и заказать желаемый товар, обеспечивая при этом легкую навигацию, множество скидок и акций. Ozon же направлен на усовершенствование своего сервиса и расширение ассортимента товаров. Несмотря на особенности каждого магазина, оба из них предлагают высокое качество обслуживания. Выбор между ними зависит только от потребностей и предпочтений покупателя.

В течение анализа были выявлены слабые и сильные стороны обоих приложений, реализация некоторых решений будет представлена также и в версии DoRi mall.

# Проектирование информационной системы

## 2.1 База данных

Задачей курсового проекта создать приложение, которое будет иметь заданный функционал:

* просмотр информации о товаре
* заказ товара
* авторизация и регистрация
* управление заказами в корзине;

В базе данных данного приложения созданы для использования 3 таблицы. Скрипт создания БД и таблиц находится в Приложении А. Ниже представлена их характеристика.

Схема базы данных представлена в таблице 2.1 – 2.3.

**Таблица 2.1- схема таблицы *users.***

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Типы данных** | **Назначение** |
| id | UUID | Первичный ключ. Определяет уникальный идентификатор записи. |
| name | Varchar (255) | Имя пользователя |
| email | Varchar (255) | Электронная почта |
| password | Varchar (255) | Пароль |
| address | Varchar (255) | Адрес, по которому осуществляется доставка |
| type | Varchar (255) | Тип пользователя – админ или обычный пользователь |
| cart | Jsonb | Хранит товары, находящиеся в корзине пользователя |

В таблице «users» содержится информация для авторизации пользователей. Столбцы данной таблицы:

* id – уникальный идентификатор, который генерируется на основе UUID, является первичным ключом;
* name – строковый тип данных;
* email – электронная почта пользователя
* password – строковый тип данных, хешируется на сервере и на мобильном приложении с помощью bcrypt
* address –строковый тип данных, определяющий адрес доставки товаров, имеет право быть не заполненным при регистрации
* type – Определяет права доступа для обычных пользователей или админов, имеет 2 состояния (user || admin)
* cart – Json поле,

**Таблица 2.2 – схема таблицы *products*.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| id | UUID | Первичный ключ. Определяет уникальный идентификатор записи. |
| name | Varchar (255) | Наименование товара |
| description | Varchar (255) | Описание товара |
| images | Jsonb | Картинка товара |
| quantity | Integer | Количество товара |
| price | Integer | Цена товара |
| category | Varchar (255) | Категория |
| ratings | Jsonb | Рейтинг |

В таблице «***products»*** содержатся товары маркетплейса. Столбцы данной таблицы:

* id – первичный ключ, который автоматически генерируется для каждой записи с помощью UUID;
* name – строковый тип, хранящий название товара;
* description – дополнительное описание товара, доступ к которому можно получить перейдя на индивидуальную страницу товара;
* images – Json поле, которое хранит картинку товара
* quantity – целочисленный тип данных для хранения количества товара
* price – целочисленный тип данных для хранения стоимости товара
* category – строковый тип данных, предназначенный для хранения категории, к которой относится товар, для дальнейшего просмотра товаров по категориям
* ratings – Json поле которое хранит вычисляемое значение для определения среднего рейтинга товара

**Таблица 2.3 – схема таблицы orders.**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип данных** | **Назначение** |
| id | UUID | Первичный ключ. Определяет уникальный идентификатор записи. |
| products | Jsonb | Товары |
| total­\_price | Numeric (10,2) | Полная стоимость |
| address | Varchar (255) | Адрес доставки |
| user\_id | UUID | Уникальный идентификатор пользователя |
| ordered\_at | BigInt | Дата заказа |
| status | Integer | Статус заказа |

В таблице «orders» содержится информация о заказах. Столбцы данной таблицы:

* id –первичный ключ, генерируется автоматически на основе UUID;
* products –Jsonb тип, хранящий все товары, находящиеся в корзине;
* total\_price – numeric тип, хранящий полную стоимость корзины пользователя;
* address– строковый тип, хранящий адрес доставки;
* user\_id - уникальный идентификатор пользователя
* ordered\_at– bigint тип данных, описывающий время, в которое был заказан товар
* status– целочисленный тип данных, хранящий состояние заказа;

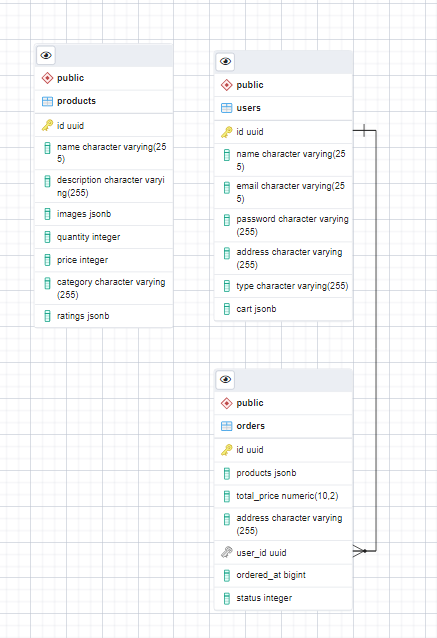
****

Рис. 2.4 – структура базы данных

В таблице users будут все данные о зарегистрированных пользователях: их имя и фамилия, пароль, электронная почта и также права доступа.

В таблице products содержатся товары и информация о них.

В таблице orders содержаться информация о заказах пользователей.

## 2.2 Проектирование мобильного приложения

В данной курсовой работе необходимо было разработать приложение, позволяющее пользователям совершать онлайн покупки. Изображение 2.5 показывает страницу регистрации, которая встречает каждого пользователя при входе в приложение

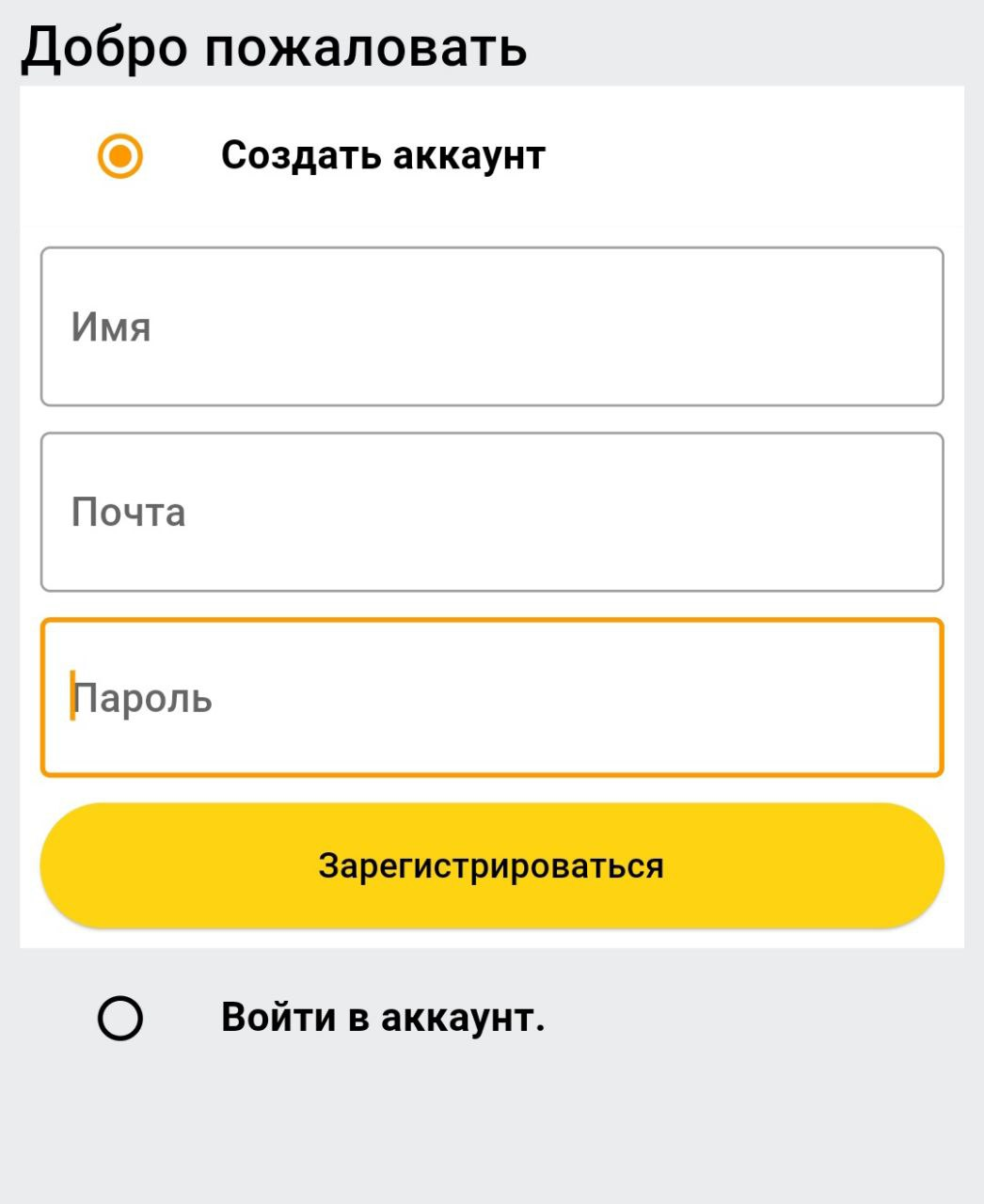


Рис. 2.5 – Страница регистрации

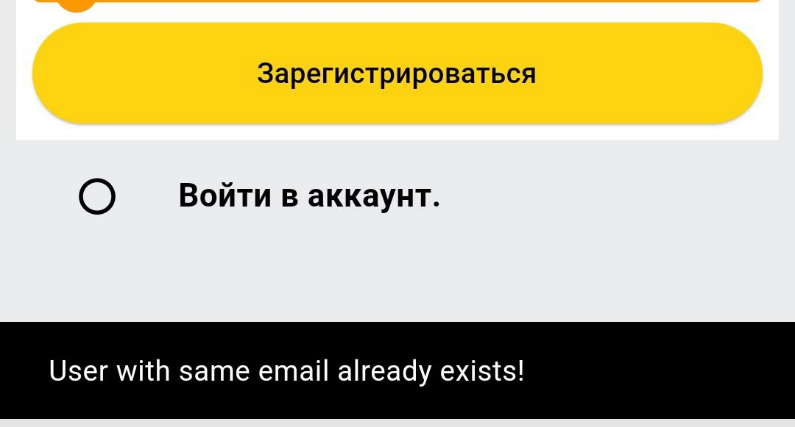
На этом экране можно зарегистрироваться или войти в уже существующий аккаунт. Для регистрации необходимо ввести имя пользователя, почту и пароль. Пользователь, чей email уже зарегистрирован не сможет создать ещё один аккаунт, вследствие чего будет выдано соответствующее сообщение (Рис. 2.6.). По умолчанию все пользователи регистрируются с типом “user”. Для входа в аккаунт необходимо ввести email и пароль. (Рис 2.7.)

Рис. 2.6. – Валидация при регистрации с same email

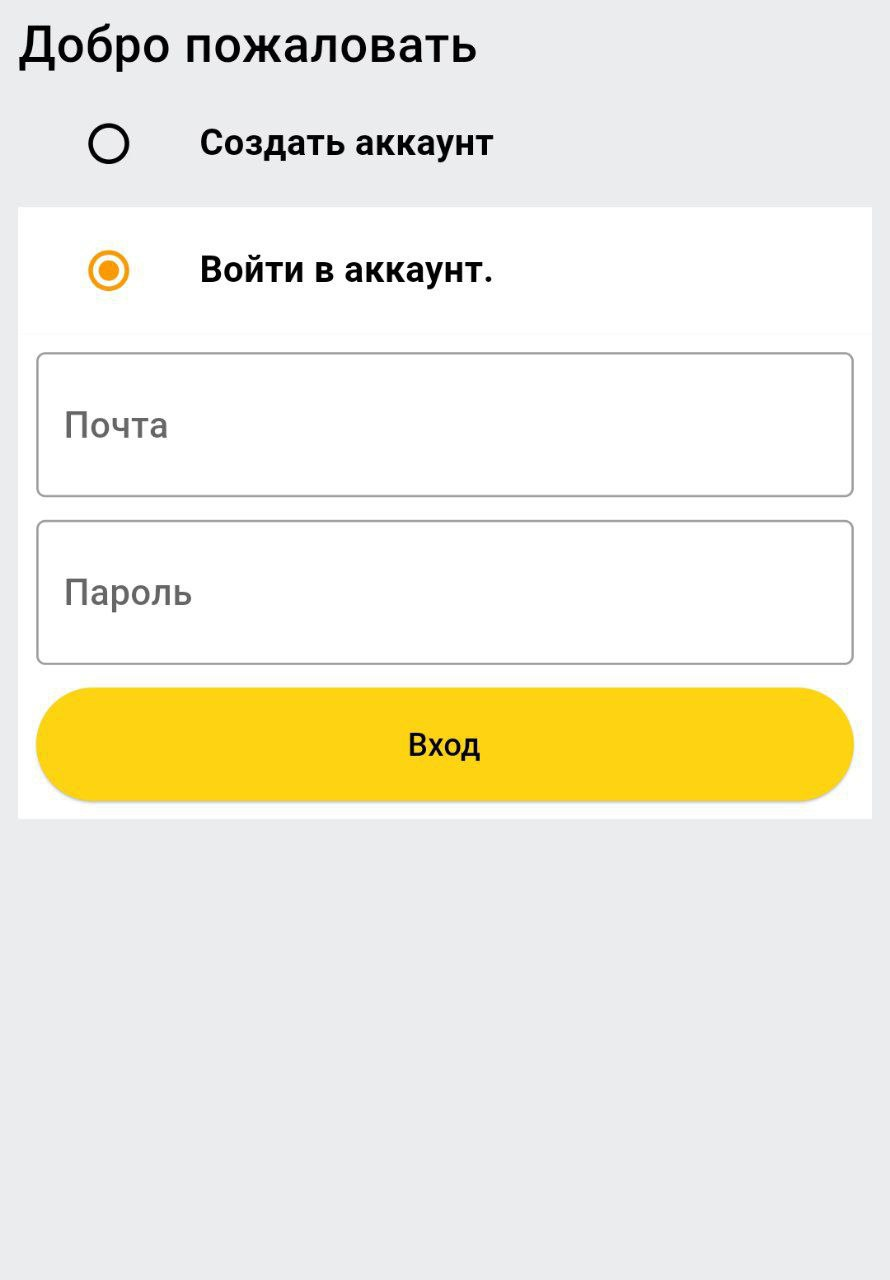


Рис 2.7 – Страница создания аккаунта

Пользователь необходимо ввести имя, почту и пароль, который должен быть не менее 8 символов. При введении пользователем некорректной информации будут выданы соответствующие сообщения. После успешной пройденной регистрации пользователь переходит на страницу со всеми товарами (Рис. 2.8.)

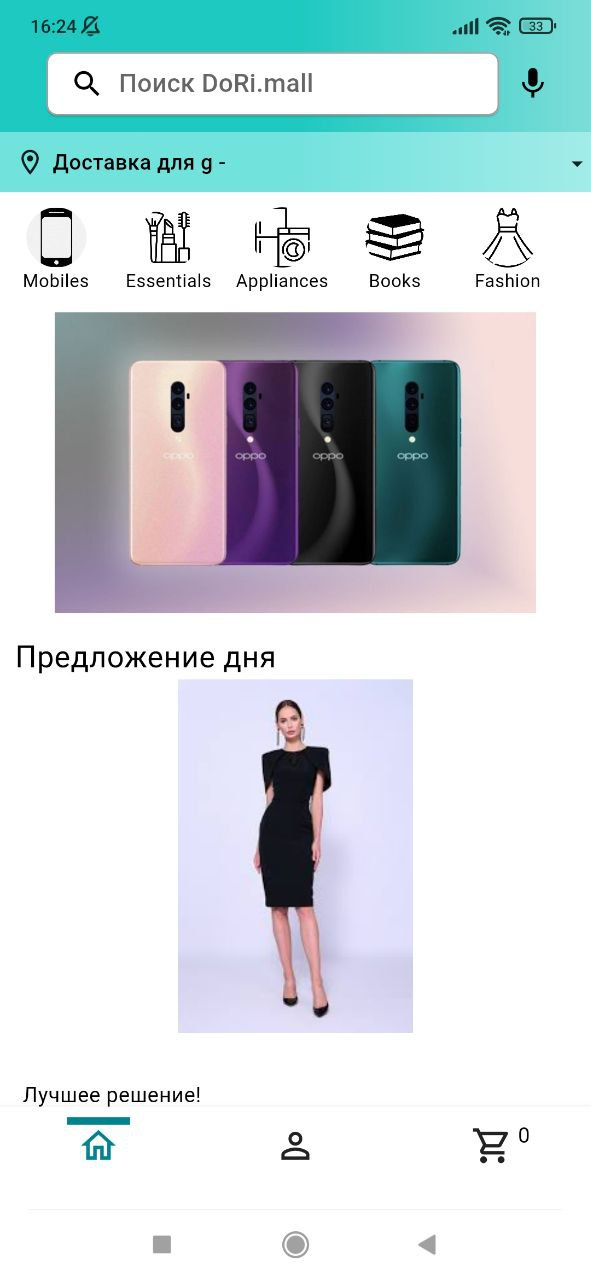
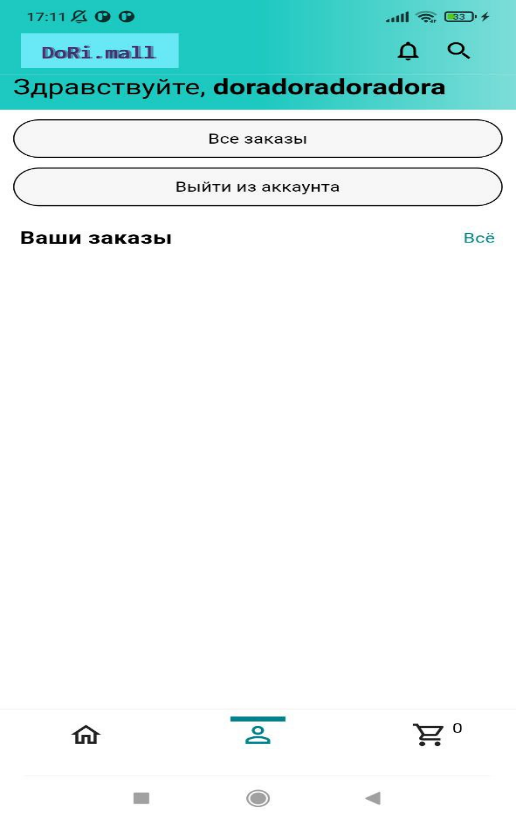


Рис 2.8 – Страница после авторизации

Пользователь, нажав на иконку, перейдет на страницу товара. На странице товара пользователь может добавить товар в корзину, купить или же выставить рейтинг. Нажав в навигационном меню на иконку человека, пользователь переходит на страницу юзера, в которой может просмотреть историю заказов, из аккаунта (рис 2.9). Нажав на картинку корзины пользователь переходит на страницу с оформлением заказа (Рис 2.10)

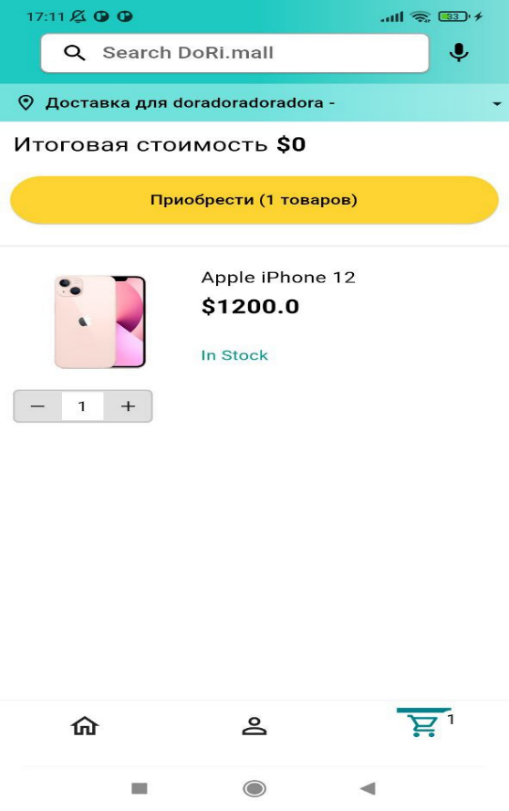
 Рис 2.9 – Страница аккаунта юзера

Рис 2.10. – Страница оплаты корзины

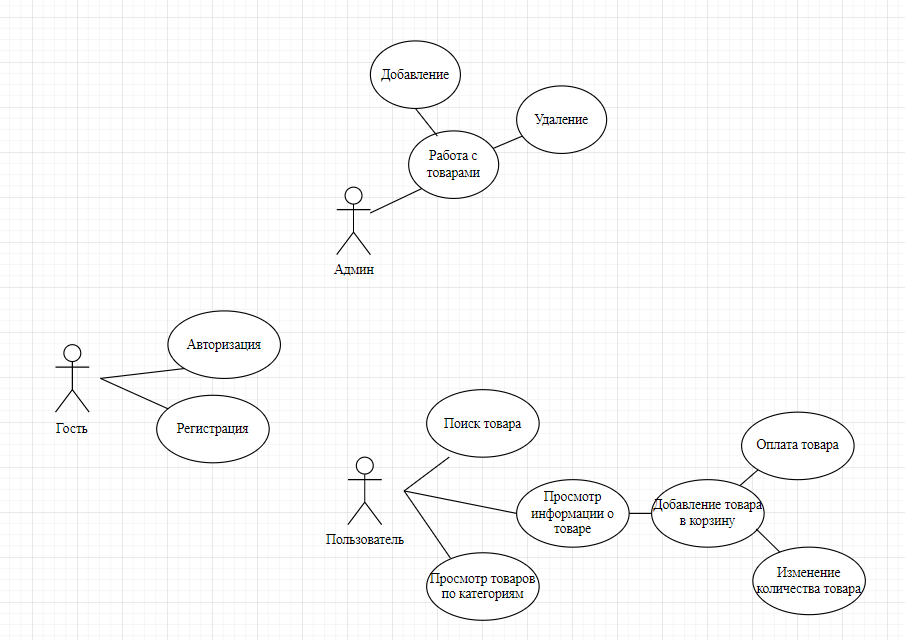


Рис. 2.9 – диаграмма использования мобильного приложения для пользователей

## 2.3 Проектирование веб-сервера

Веб-приложение - клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Веб приложение состоит из двух компонентов: веб сервера и клиента. Веб. Веб сервер имеет REST архитектуру и осуществляет взаимодействие с базой данных. Сервер содержит в себе реализованные сервисы для взаимодействия с каждой сущностью базы данных.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер будет возвращать данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь позволяет передать данные на сервер для последующей обработки. Метод PUT позволяет обновить соответствующие данные, переданные с клиента. Метод DELETE удаляет с базы данных заданную запись.

Веб-приложение состоит их двух частей: клиент и сервер. Клиент в свою очередь выступает в роле интерфейса веб-приложения.

## Вывод по главе «Проектирование информационной системы»

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 и описаны варианты использования программного средства.

Была также спроектирована база данных, где были определены необходимые таблицы, охарактеризованы поля каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, заключающиеся в создании первичных и внешних ключей.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между сервером и клиентом.

# Разработка информационной системы

Для разработки приложений курсового проекта были использованы следующие технические средства:

PostgreSQL - реляционная база данных.

SQL – декларативный язык программирования, применяемый для создания, модификации, и управления данными в РБД.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Visual Studio Code – редактор исходного кода.

Flutter - комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений

Dart –язык программирования.

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript.

Render – хостинг базы данных и сервера

## Разработка базы данных

Для хранения данных, поступающих из приложения, использована PostgreSQL.

Также были созданы хранимые процедуры, для взаимодействия с таблицами базы данных.

//admin/delete-product

CREATE OR REPLACE PROCEDURE delete\_product(id\_val UUID)

LANGUAGE SQL

AS $$

DELETE FROM products WHERE id = id\_val;

$$;

//admin-change-order=status

CREATE OR REPLACE PROCEDURE update\_order\_status(status\_val INTEGER, id\_val UUID)

LANGUAGE SQL

AS $$

UPDATE orders SET status = status\_val WHERE id = id\_val;

$$;

///api/signup

CREATE OR REPLACE PROCEDURE get\_user\_by\_email(email\_val VARCHAR(255))

LANGUAGE SQL

AS $$

SELECT \* FROM users WHERE email = email\_val; $$

Листинг 2 – примеры хранимых процедур

## 3.2 Разработка мобильного приложения

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на рисунке 3.

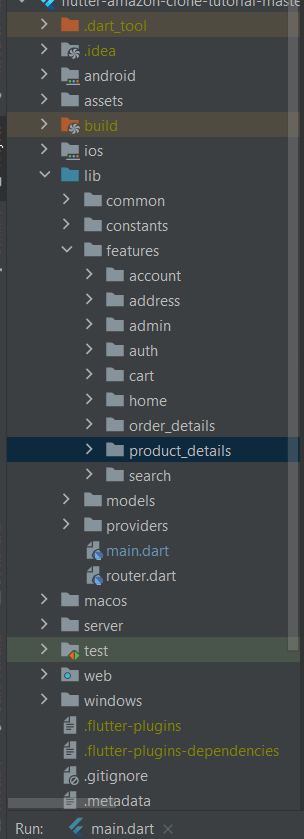


Рис. 3 – Графическая структура проекта

С помощю пакет http приложение асинхронно отправляет запросы к серверу (листинг 1)

http.Response res = await http.post(  
 Uri.https(uri,'/api/signin'),  
 body: jsonEncode({  
 'email': email,  
 'password': password,  
 }),  
 headers: <String, String>{  
 'Content-Type': 'application/json; charset=UTF-8',  
 },  
);

Листинг 1 – обращение к серверу со стороны клиента

После вызова http, через точку, прописывается метод для взаимодействия с сервером (get,post,put,delete)

Префикс $uri устанавливается в каждом пути для обращения к серверу

Это глобальная переменная, устанавливаемая в файле global\_variable, которая представляет собой переменную вида “String uri = 'server-coursework.onrender.com';” –представляет собой ссылку подключения к хостингу, на котором деплоится сервер.

## 3.3 Разработка веб-приложения

Веб-сервер разработан с использование языка программирование JavaScript и среды исполнения NodeJS. NodeJS — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Веб сервер разработан с использованием архитектуры REST. Каждая программная сущность определена в отдельный модуль. Папка middlewares содержит два файла – для аутентификаци администратора и авторизации с использованием токена

Routes содержит все маршруты и команды по 4 категориям

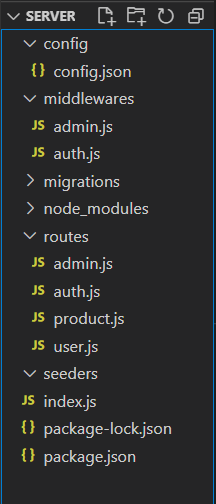


Рис. 3.5 – структура веб-сервера

В общем случае обращение к бд выглядит вызов конкретной процедуры и передача параметров, если они нужны (Листинг 2)

// get user data

authRouter.get("/", async (req, res) => {

  try {

    const query = 'CALL get\_user\_by\_id(1)';

    const values = [req.user];

    const { rows } = await pool.query(query, values);

    const user = rows[0];

    res.json({ ...user });

  } catch (e) {

    res.status(500).json({ error: e.message });

  }

});

Листинг 2 - пример обработки запроса к серверу

## Вывод по главе «Разработка информационной системы»

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третьим этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрены некоторые классы и их методы, в частности.

Четвертый этап – это разработка веб-приложения, где была представлена графическая структура веб-сервера и веб-клиента, рассмотрены некоторые классы и их методы.

# 4. Обеспечение безопасности информационной системы

Для обеспечения защиты персональных данных пользователей применялись методы хеширования и криптографии. Пароли пользователей информационной системы хранятся в базе данных в хешированном виде. Хеширование – это процесс преобразования входной строки переменной длины в выходную строку фиксированной длины, называемой хеш-кодом или дайджестом сообщения. Процесс хеширования реализуется путем использования криптографической хеш-функции (Приложение B). Хеширование паролей не позволяет злоумышленнику получить доступ к аккаунту пользователя информационной системы.

В данном приложении реализована функция хеширование пароля с помощью bcrypt. Он надежнее чем MD5, так как не может из хешированных данных получить пароль. Вместо этого он сравнивает два разных хеша и смотрит на совпадение. Если хеши совпадают, значит пароль введен правильно.

Для аутентификации юзеров используются токены

Токен — это ссылка (то есть идентификатор), которая сопоставляется с конфиденциальными данными через систему токенизации.

Пример листинга находится в приложении B

Когда пользователь проходит аутентификацию в приложении, сервер генерирует уникальный токен для этого пользователя. Этот токен затем сохраняется в БД, связанный с идентификатором пользователя. Когда пользователь пытается получить доступ к определенному ресурсу, приложение отправляет запрос на сервер с этим токеном. Сервер ищет токен в БД и проверяет его наличие и соответствие конкретному пользователю. Если токен существует и соответствует пользователю, сервер разрешает доступ к ресурсу.

При использовании токенов для аутентификации в БД важно обеспечить безопасность. Токены должны быть достаточно длинными и случайными, чтобы не могли быть подобраны злоумышленником. Кроме того, они должны быть защищены от утечки и изменения. Для этого токены могут быть зашифрованы и подписаны с помощью криптографических алгоритмов. Также необходимо регулярно проверять токены на предмет истечения срока действия и удаления их из БД после истечения этого срока.

# 

# Заключение

Целью моего курсового проекта было предоставить пользователям возможность легко и удобно покупать товары.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В веб-приложении был определен основной функционал, также была разработана диаграмма вариантов использования. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: PostgreSQL, сред разработки Android Studio и Visual Studio Code, языков программирования Dart и JS, фреймворка flutter и программной платформы node.js.В качестве хостинга – render.

Была реализована База Данных.

При решении данной задачи была достигнута поставленная цель, и был создан проект DoRi Mall, который включает в себя android-клиент, веб-клиент и веб-сервер.

# Список использованных источников

1. Архитектура REST [Электронный ресурс] – habr.com – Режим доступа: https://habr.com/ru/post/38730/
2. Про токены, JSON Web Tokens (JWT) - https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc
3. Хранение изображений приложения в БД https://habr.com/ru/articles/580044/
4. Dart, Flutter и кроссплатформа - https://habr.com/ru/articles/651065/

# 

# Приложение А

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

address VARCHAR(255) DEFAULT '',

type VARCHAR(255) DEFAULT 'user',

cart JSONB DEFAULT '[]'

);

CREATE TABLE ratings (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

userId VARCHAR(255) NOT NULL,

rating INTEGER NOT NULL

);

CREATE TABLE orders (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

products JSONB NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

address VARCHAR(255) NOT NULL,

user\_id UUID NOT NULL REFERENCES users(id),

ordered\_at BIGINT NOT NULL,

status INTEGER DEFAULT 0

);

CREATE TABLE products (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(255) NOT NULL,

description VARCHAR(255) NOT NULL,

images JSONB DEFAULT '[]',

quantity INTEGER NOT NULL,

price INTEGER NOT NULL,

category VARCHAR(255) NOT NULL,

ratings JSONB DEFAULT '[]'

);

# Приложение B

const auth = async (req, res, next) => {

  try {

    const token = req.header('x-auth-token');

    if (!token)

      return res.status(401).json({ msg: 'No auth token, access denied' });

    const decodedToken = jwt.verify(token, 'passwordKey');

    if (!decodedToken)

      return res

        .status(401)

        .json({ msg: 'Token verification failed, authorization denied.' });

    const { rows } = await pool.query('SELECT id FROM "users" WHERE id=$1', [

      decodedToken.id,

    ]);

    if (rows.length !== 1) {

      return res.status(401).json({ msg: 'User not found' });

    }

    req.user = decodedToken.id;

    req.token = token;

    next();

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

module.exports = auth;