МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Пиццерия»

Выполнил студент Адамович Антон Максимович

(Ф.И.О.)

Руководитель проекта ст.преподаватель Блинова Е.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: ст. преподаватель Блинова Е.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Блинова Е.А (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

**Минск 2022**

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc135684247)

[**1. Обзор аналогов приложений «Пиццерия»** 4](#_Toc135684248)

[**1.1 Приложение «Додо пицца»** 4](#_Toc135684249)

[**1.2 Приложение «Доминос пицца»** 8](#_Toc135684250)

[**1.3 Вывод по главе «Обзор аналогов приложений «Пиццерия»** 10](#_Toc135684251)

[**2. Проектирование программного продукта** 12](#_Toc135684252)

[**2.1 Проектирование мобильного приложения** 12](#_Toc135684253)

[**2.2 Проектирование локальной базы данных** 14](#_Toc135684254)

[**2.3 Проектирование глобальной базы данных** 15](#_Toc135684255)

[**2.4 Проектирование веб-приложения** 18](#_Toc135684256)

[**2.5 Проектирование сервера** 18](#_Toc135684257)

[**2.6 Выводы по главе «Проектирование программного продукта»** 19](#_Toc135684258)

[**3. Программная реализация приложений** 20](#_Toc135684259)

[**3.1 Технические средства для разработки** 20](#_Toc135684260)

[**3.2 Разработка баз данных** 20](#_Toc135684261)

[**3.3 Разработка мобильного приложения** 20](#_Toc135684262)

[**3.4 Разработка сервера** 25](#_Toc135684263)

[**3.5 Разработка веб-приложения** 26](#_Toc135684264)

[**3.6 Вывод по главе «Программная реализация приложений»** 26](#_Toc135684265)

[**4. Обеспечение безопасности информационной системы** 27](#_Toc135684266)

[**4.1 Вывод по главе «Обеспечение безопасности информационной системы»** 27](#_Toc135684267)

[**5. Руководство пользователя** 28](#_Toc135684268)

[**5.1 Вывод по главе «Руководство пользователя»** 33](#_Toc135684269)

[**ЗАКЛЮЧЕНИЕ** 34](#_Toc135684270)

[**Список использованных источников** 36](#_Toc135684271)

[**ПРИЛОЖЕНИЕ А** 37](#_Toc135684272)

[**ПРИЛОЖЕНИЕ Б** 38](#_Toc135684273)

# **ВВЕДЕНИЕ**

В современном мире люди ведут активный образ жизни, и у них остается меньше времени на приготовление пищи. Фаст-фуд стал удобным вариантом для многих, а пицца — одним из самых популярных блюд быстрого питания во всем мире. С появлением онлайн-заказов и служб доставки на дом спрос на пиццу только увеличился. Следовательно, существует потребность в удобном, эффективном и простом в использовании приложении для заказа пиццы, которое могло бы предоставить клиентам приятный и беспроблемный опыт.

Приложение для пиццерии позволит клиентам заказывать пиццу и настраивать ее в соответствии со своими предпочтениями. Также нужны такие функции, как отслеживание заказов в режиме реального времени и планирование доставки. Цель — разработать приложение, которое обеспечивает отличный пользовательский интерфейс и максимально упрощает процесс заказа пиццы.

Курсовой проект включает в себя 7 разделов: введение, содержание, обзор аналогов, проектирование программного продукта, программная реализация продукта, обеспечение информационной безопасности системы, руководство по использованию, заключение, список литературных источников.

# **1. Обзор аналогов приложений «Пиццерия»**

## **1.1 Приложение «Додо пицца»**

«Додо пицца» – это мобильное приложение, которое позволяет клиентам заказывать пиццу в местном ресторане «Додо Пицца» с доставкой или самовывозом.

Приложение доступно для устройств iOS и Android и может быть бесплатно загружено из соответствующих магазинов приложений. После загрузки пользователи могут создать учетную запись, просматривать меню, настраивать свой заказ и отслеживать статус доставки своей пиццы в режиме реального времени.

Приложение поддерживает следующие функции:

* регистрация и авторизация пользователя;
* поиск по еде;
* конфигурация пиццы;
* корзина товаров;
* начисление баллов и покупка еды за баллы.

На рисунках 1.1–1.3 представлен интерфейс приложения.

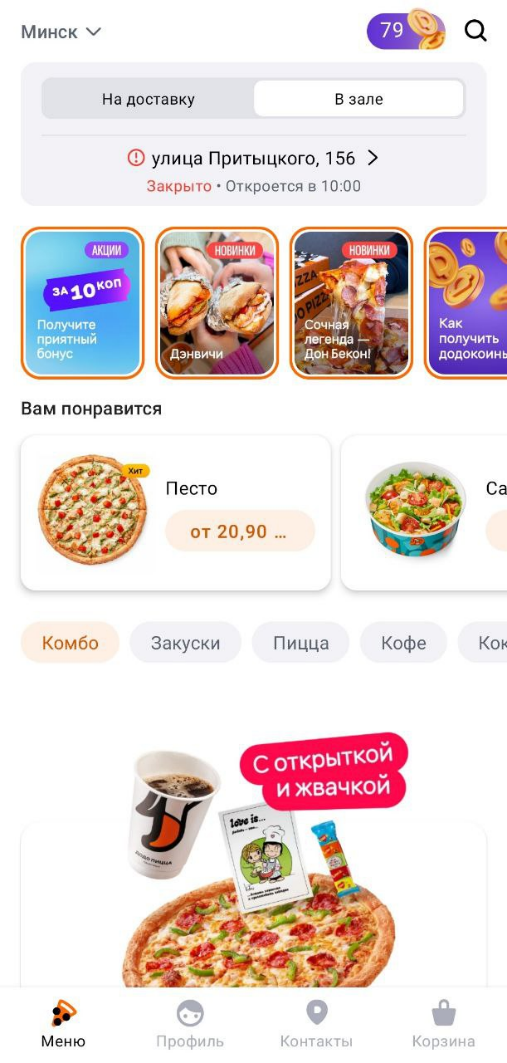


Рисунок 1.1 – Пример интерфейса приложения «Додо пицца»

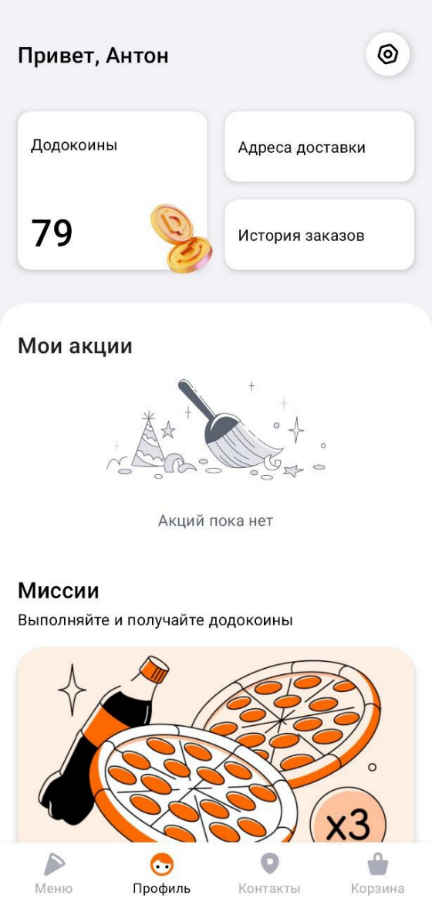


Рисунок 1.2 – Пример интерфейса приложения «Додо пицца»

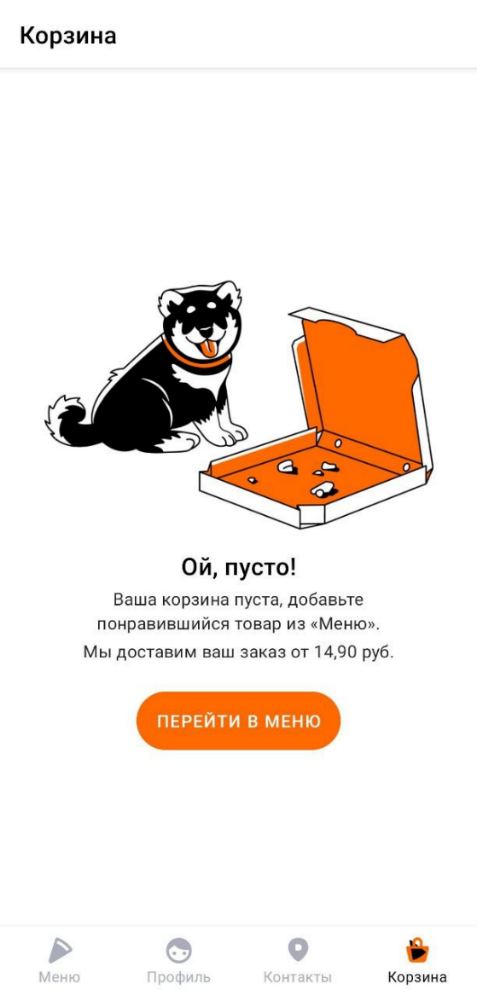


Рисунок 1.3 – Пример интерфейса приложения «Додо пицца»

Одной из уникальных особенностей приложения «Додо Пицца» является использование технологий. Например, клиенты могут использовать приложение, чтобы настроить свою пиццу до мельчайших деталей, включая тип корочки, начинки и даже соус. Они также могут сохранять свои любимые заказы и быстро менять их порядок всего несколькими нажатиями. Данная особенность представлена на рисунке 1.4.

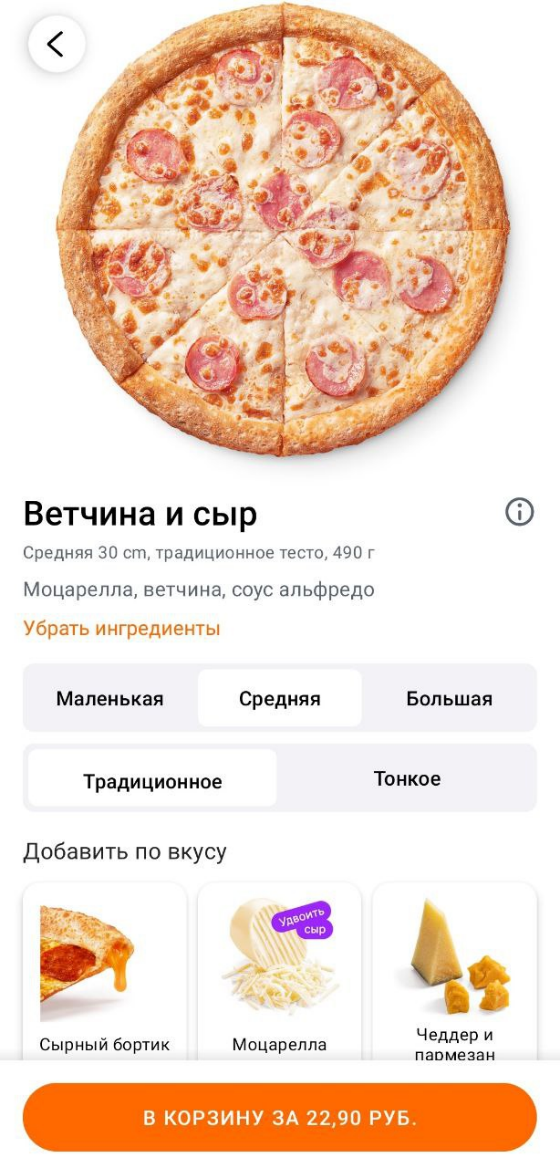


Рисунок 1.4 – Пример конфигурации заказа

Кроме того, приложение предлагает программу лояльности, которая вознаграждает клиентов баллами за каждый размещенный ими заказ. Эти баллы можно обменять на бесплатную пиццу, напитки и другие товары. Реализация показана на рисунке 1.5.

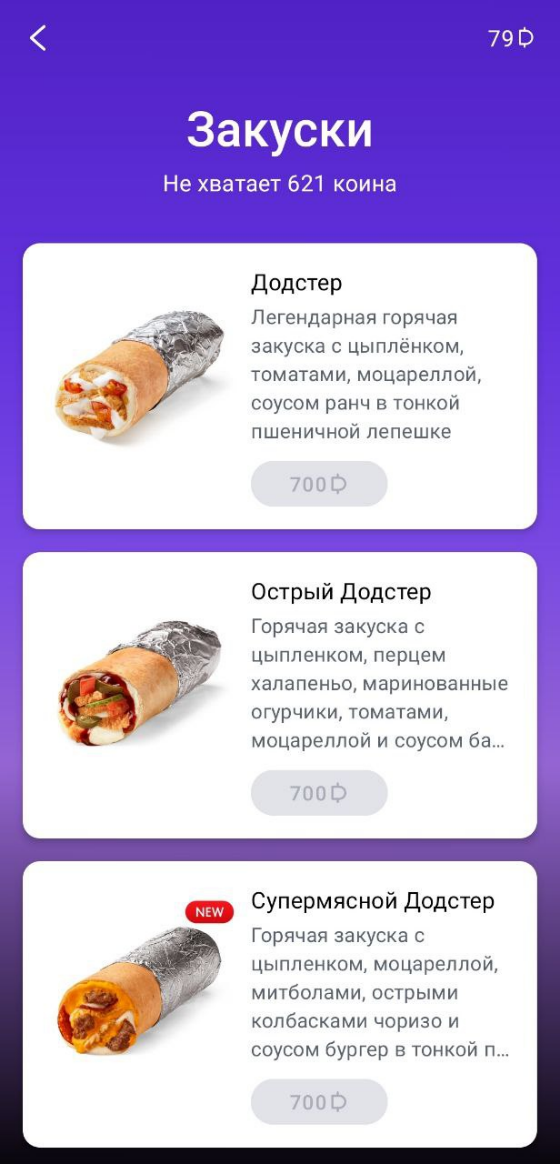


Рисунок 1.5 – Реализация покупки еды за баллы

В целом, приложение «Додо Пицца» — это удобный способ для клиентов заказать пиццу в местном ресторане «Додо Пицца». Он сочетает в себе технологии с отличным меню пиццы и программой вознаграждений, чтобы создать плавный и приятный опыт.

## **1.2 Приложение «Доминос пицца»**

«Доминос пицца» – это мобильное приложение, которое позволяет клиентам заказывать пиццу и другие блюда из местного ресторана Domino's Pizza с доставкой или самовывозом.

Приложение поддерживает следующие функции:

* регистрация и авторизация пользователя;
* поиск по еде;
* конфигурация пиццы;
* корзина товаров.

На рисунках 1.6 и 1.7 представлен интерфейс приложения.

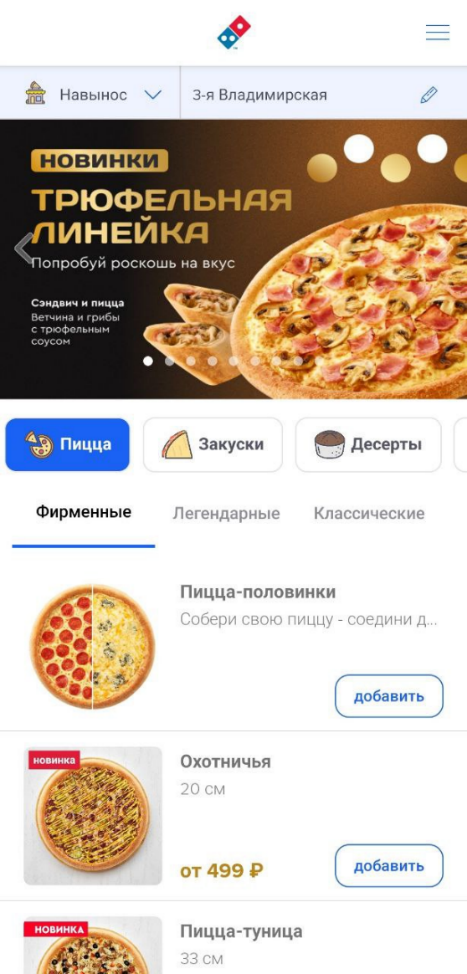


Рисунок 1.6 – Пример интерфейса приложения «Доминос пицца»

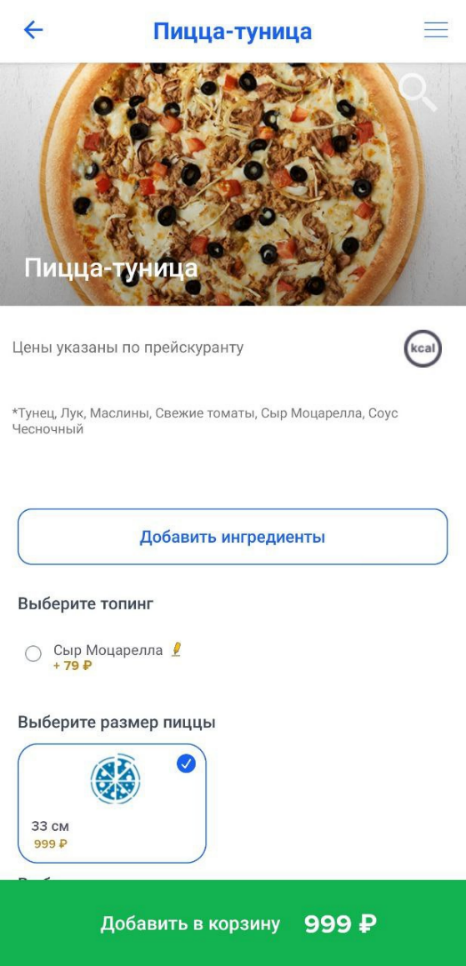


Рисунок 1.7 – Пример интерфейса приложения «Доминос пицца»

В общем, приложение «Доминос пицца» поддерживает такие же функции, как и приложение «Додо Пицца».

## **1.3 Вывод по главе «Обзор аналогов приложений «Пиццерия»**

В заключение отметим, что аналоги приложений для пиццерий предоставляют ряд возможностей и функций, которые помогают упростить процесс заказа и улучшить качество обслуживания клиентов. Эти приложения позволяют клиентам просматривать меню, настраивать свои заказы и отслеживать доставку в режиме реального времени с помощью своих мобильных устройств.

Более того, программы лояльности, предлагаемые многими из этих приложений, стимулируют повторные покупки, что делает их ценным инструментом для повышения лояльности клиентов и увеличения продаж.

В целом популярность этих приложений подчеркивает важность цифровых решений для современного бизнеса, особенно в сфере производства продуктов питания и напитков. Поскольку технологии продолжают развиваться, вполне вероятно, что в будущем мы увидим еще более сложные и инновационные приложения для пиццерий и других типов ресторанов.

Проанализированные преимущества будут учтены при разработке курсового проекта.

# **2. Проектирование программного продукта**

Данный раздел будет описывать все основные части данного приложения. К основным частям относятся:

* локальная база данных;
* глобальная база данных;
* серверное приложения;
* веб-приложение;
* мобильное приложения.

## **2.1 Проектирование мобильного приложения**

В данной курсовой работе необходимо было разработать приложение, позволяющее пользователям осуществлять заказ пиццы.

Если пользователь первый раз запускает приложение на устройстве или не авторизован, то открывается экран входа. У пользователя есть возможность ввести логин и пароль, если он зарегистрирован в приложении, либо зарегистрироваться.

При входе на страницу регистрации пользователь вводит свои логин и придумывает пароль. После сохранения данных, пользователь переходит на страницу авторизации.

В случае ввода верных логина и пароля пользователю открывается домашний экран, где отображаются все пиццы, которые доступны для заказа. Сверху находится кнопка поиска, которая позволяет найти нужную пиццу. Также присутствует нижняя навигационная панель, позволяющая пользователю переходить между различными страницами приложения.

При нажатии на пиццу в списке открывается окно с конфигуратором для выбора конфигурации пиццы: размера, теста и дополнительного сыра. При выборе конфигурации можно нажать на кнопку Добавить и товар добавится в корзину.

При переходе на экран профиля пользователь может изменить свои данные включая номер телефона и дату рождения. Также там находятся кнопка выхода из приложения, кнопка управления адресами и кнопка с отображением истории заказов.

При переходе на страницу с адресами, на которой есть поле и кнопка с добавлением адреса и список с адресами. В списке при свайп-жесте адрес удаляется из списка.

При переходе на страницу с историей отображается список выполненных заказов текущего пользователя.

При переходе на страницу с заказами отображается список заказов, находящихся в доставке.

На странице с корзиной находится список добавленных товаров для заказа. Есть возможность выбрать количество для определенного товара. Далее пользователь выбирает адрес для заказа и нажимает кнопку Заказать.

При входе в аккаунт происходит синхронизация мобильной базы данных с глобальной базой данных.

В случае, если пользователем будет админ, то ему предоставиться возможность открыть страницу с добавлением пиццы, а также удалением пиццы из списка с помощью свайп-жеста.

Если же пользователем будет доставщик, то у него открывается окно со всеми заказами и возможностью с помощью свайпа отметить его готовность.

Можно ознакомиться с UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. UML-диаграмма представлена на рисунке 2.1.

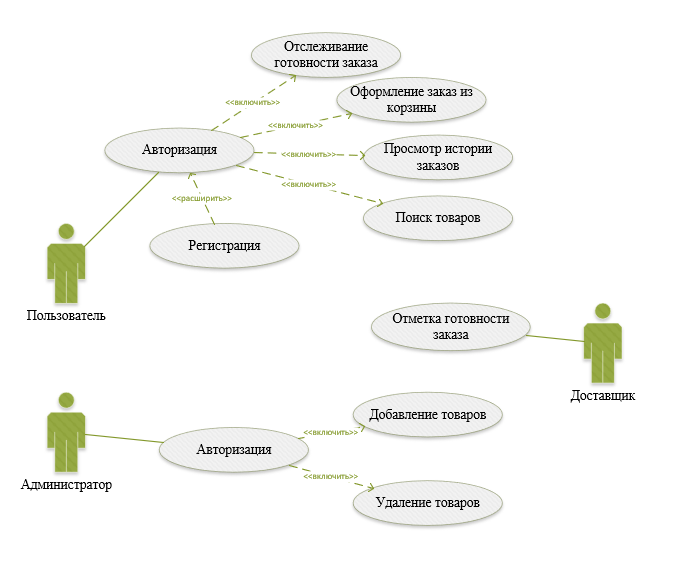


Рисунок 2.1 – Диаграмма вариантов использования

Таким образом приложение должно выполнять функции:

* регистрация пользователя;
* авторизация пользователя;
* выбор конфигурации пиццы и добавление в корзину;
* добавлять и удалять пиццы;
* добавлять и удалять новые адреса;
* просматривать историю заказов;
* оформлять заказ на доставку;
* отмечать доставку заказа.

## **2.2 Проектирование локальной базы данных**

Локальная база данных будет создана для использования приложения в автономном режиме и сохранения корзины локально. При отключении интернета пользователь сможет просматривать историю заказов и товары в корзине.

В базе данных мобильного приложения будут созданы для использования 5 таблиц. Скрипт создания БД и таблиц будет представлен в Приложении А. Ниже представлена их характеристика.

В таблице «Cart» содержится информация о корзине. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Pizza\_Id – вторичный ключ, ссылается на первичный ключ таблицы «Pizzas», служит для связи этих таблиц.
* Quantity – содержит количество пиццы, целочисленный тип данных.

В таблице «Pizzas» содержится информация о пиццах из корзины. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Name – содержит имя пиццы, строковый тип данных;
* Size – содержит количество товара, целочисленный тип данных;
* Dough – содержит размер товара, строковый тип данных;
* Cheese – содержит информацию о сыре, строковый тип данных;
* Price – содержит информацию о цене, плавающий с точкой численный тип данных;
* Image – содержит ссылку на картинку, строковый тип данных.

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

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* User\_Id – целочисленный тип данных, внешний идентификатор;
* Delivery\_Id – целочисленный тип данных, внешний идентификатор;
* Created\_At – содержит информацию о дате заказа, строковый тип данных;
* Status – содержит информацию о статусе, строковый тип данных.

В таблице «OrderItems» содержится информация о товарах заказа. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Order\_Id – целочисленный тип данных, внешний идентификатор;
* Product\_Id – целочисленный тип данных, внешний идентификатор;
* Product\_Details\_Id – целочисленный тип данных, внешний идентификатор;
* Quantity – содержит информацию о количестве, целочисленный тип данных.

Схема локальной базы данных представлена на рисунке 2.2.

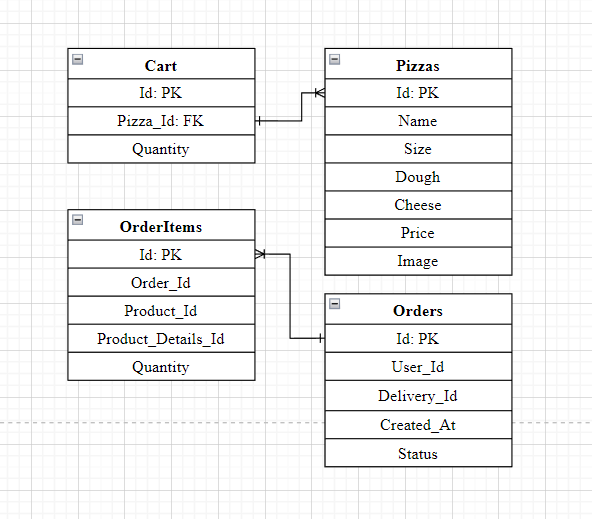


Рисунок 2.2 – Схема локальной базы данных

Таким образом, локальная база данных обеспечит сохранение данных при автономной работе приложения конкретного пользователя.

## **2.3 Проектирование глобальной базы данных**

В глобальной базе данных созданы для использования 7 таблиц. Глобальная база данных обеспечивает хранение данных всех пользователей. Скрипт создания БД и таблиц находится в Приложении Б. Ниже представлена их характеристика.

В таблице «User» содержится информация о пользователях. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Rold\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Username – содержит имя пользователя, строковый тип;
* Password – содержит пароль пользователя, строковый тип;
* Phone\_Number – содержит номер телефона, строковый тип;
* Birthday – содержит день рождения пользователя, строковый тип.

В таблице «Role» содержится информация о ролях. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Name – содержит название роли, строковый тип.

В таблице «DeliveryAddress» содержится информация о корзине. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* User\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Address – содержит адрес, строковый тип данных.

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

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* User\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Delivery\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Created\_At – содержит дату создания заказа, строковый тип данных;
* Status – содержит статус заказа, строковый тип данных.

В таблице «OrderDetail» содержится информация о деталях заказа. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Order\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Product\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Pizza\_Detail\_Id – целочисленный тип данных, уникальный идентификатор, является вторичным ключом;
* Quantity – содержит количество заказанного товара, целочисленный тип данных.

В таблице «Pizza» содержится информация о пиццах. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Name – содержит имя пиццы, строковый тип данных;
* Description – содержи описание пиццы, строковый тип данных;
* Price – содержит информацию о цене, плавающий с точкой численный тип данных;
* Image – содержит ссылку на картинку, строковый тип данных.

В таблице «PizzaDetails» содержится информация о корзине. Столбцы данной таблицы:

* Id – целочисленный тип данных, уникальный идентификатор, является первичным ключом;
* Size – содержит количество товара, целочисленный тип данных;
* Dough – содержит размер товара, строковый тип данных;
* Cheese – содержит информацию о сыре, строковый тип данных.

Схема глобальной базы данных представлена на рисунке 2.2.

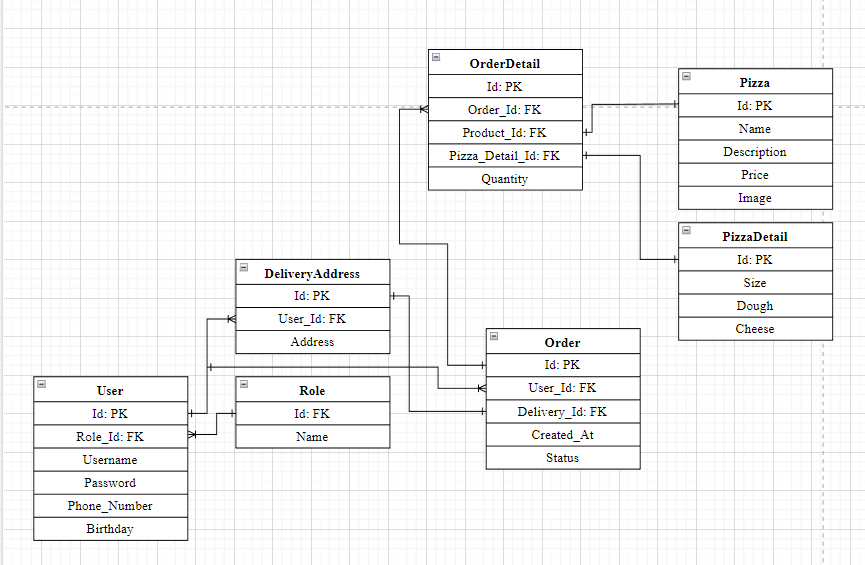


Рисунок 2.3 – Схема глобальной базы данных

Таким образом, глобальная база данных обеспечит сохранение всех данных приложения каждого пользователя при использовании приложения с подклчение к сети Интернет.

## **2.4 Проектирование веб-приложения**

Веб-приложение представляет из себя тоже самое, что и мобильное приложение по функционалу. Отличие будет лишь в том, что веб-приложение не имеет автономного режима. Веб-приложение должно выполнять следующие функции:

* регистрация пользователя;
* авторизация пользователя;
* просмотр, добавление и удаление пицц;
* отметка о доставке заказа.

## **2.5 Проектирование сервера**

Для синхронизации двух и более клиентов необходимо серверное приложение. Серверная часть приложения должна предоставлять WEB API для доступа к каждой сущности в базе данных. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON.

Серверное приложение должно быть построено на основе архитектурного стиля REST и будет использоваться для взаимодействия компонентов распределённого приложения в сети.

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.

## **2.6 Выводы по главе** **«Проектирование программного продукта»**

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 клиентом и сервером, формат передачи данных, обозначили HTTP методы, которые будут использованы в работе с серверной частью.

При проектировании веб-приложения были определен основной функционал, который предусматривает тот же функционал, что и мобильное приложения, за исключением автономного режима.

# **3. Программная реализация приложений**

## **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQFLite – компактная встраиваемая реляционная база данных.

PostgreSQL – система управления реляционными БД.

Flutter – комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, веб-приложений, а также настольных приложений под Windows, macOS и Linux с использованием языка программирования Dart.

Node.js – программная платформа, основанная на движке V8, превра-щающая JavaScript из узкоспециализированного языка в язык общего назначения.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

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

WebStorm – интегрированная среда разработки на JavaScript, CSS и HTML.

Postman – набор инструментов тестирования API.

## **3.2 Разработка баз данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite. Для сервера будет использована СУБД PostgreSQL.

Код создания структуры мобильной базы данных представлен в Приложении А, серверной – в Приложении Б.

Локальная база данных содержит в себе информацию о корзине и истории заказов пользователя и используется для автономной работы.

Глобальная база данных содержит в себе информацию о всех пользователях и их данных. Используется в качестве основного хранилища данных и находится в облачном сервисе.

## **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Dart и фреймворк Flutter, позволяющий создавать приложения для различных платформ. Вся разработка будет вестись через создание файлов для отдельных страниц и с упором на мобильное приложение в плане дизайна.

Графическая структура проекта представлена на рисунках 3.1 и 3.2.

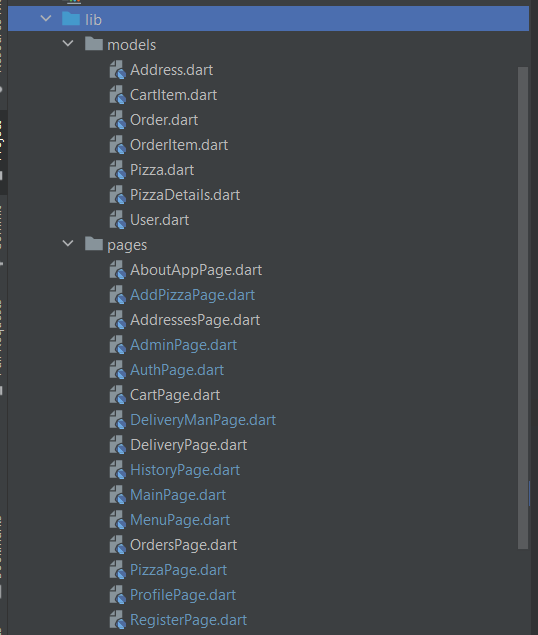


Рисунок 3.1 – Графическая структура проекта

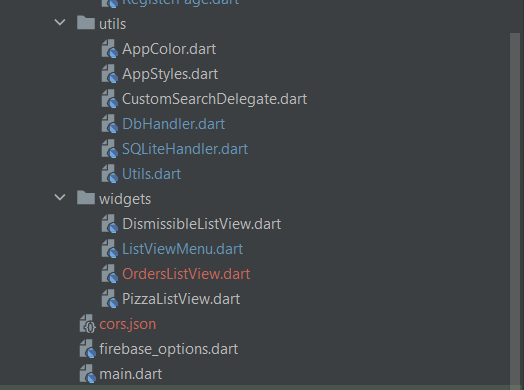


Рисунок 3.2 – Графическая структура проекта

Пакет «Pages» хранит в себе классы страниц приложения, которые позволяют пользователю зарегистрироваться и авторизоваться, просматривать, добавлять и удалять рецепты, добавлять адреса, добавлять и удалять рецепты из списка избранного и просматривать информацию о пользователях. Основные методы классов данного пакета приведены в таблице 3.1.

**Таблица 3.1 – Содержание пакета «Pages»**

|  |  |
| --- | --- |
| Класс MainPage – главная страница приложения | |
| initState() | Метод позволяет инициализировать некоторые из компонентов приложения. |
| build() | Создание основного визуала страницы |
| Класс AuthPage – страница авторизации пользователя | |
| initState() | Метод позволяет инициализировать некоторые из компонентов приложения. |
| build() | Создание основного визуала страницы |
| updateTextValue() | Обновляет данные пользователя для входа |
| Класс RegisterPage – страница регистрации пользователя | |
| initState() | Метод позволяет инициализировать некоторые из компонентов приложения. |
| build() | Создание основного визуала страницы |
| updateTextValue() | Обновляет данные пользователя для входа |
| Класс MenuPage – страница с списком пицц | |
| initState() | Метод позволяет инициализировать некоторые из компонентов приложения. |
| build() | Создание основного визуала страницы |
| \_initLocalDb() | Инициализирует локальную базу данных |
| \_tryToGetOrdersHistory() | Попытка синхронизации локальный и глобальной баз данных |
| \_fillDbWithOrders() | Заполнение локальной БД заказами |
| \_fillDbWithOrderItems () | Заполнение локальной БД предмет заказа |
| Класс AdminPage – страница для админа | |
| initState() | Метод позволяет инициализировать некоторые из компонентов приложения. |
| build() | Создание основного визуала страницы |
| \_loadPreferences() | Загрузка Shared Preferences |
| \_loadUser() | Загрузка пользователя из SP |

Пример создания нижнего меню представлен в листинге 1.

int \_currentChildIndex = 0;  
  
final List<Widget> \_children = [  
 MenuPage(),  
 ProfilePage(),  
 OrdersPage(),  
 CartPage()  
];  
  
void onBnbTapped(int index) {  
 setState(() {  
 \_currentChildIndex = index;  
 });  
}  
  
@override  
void initState() {  
 super.initState();  
}  
  
@override  
Widget build(BuildContext context) {  
 return Scaffold(  
 body: \_children[\_currentChildIndex],  
 bottomNavigationBar: BottomNavigationBar(  
 type: BottomNavigationBarType.fixed,  
 backgroundColor: Colors.*black*,  
 selectedItemColor: AppColor.*pumpkin*,  
 unselectedItemColor: Colors.*white*,  
 onTap: onBnbTapped,  
 currentIndex: \_currentChildIndex,  
 items: [  
 BottomNavigationBarItem(  
 icon: ImageIcon(AssetImage('assets/icons/pizzaBarIcon.png')),  
 label: 'Меню'  
 ),  
 BottomNavigationBarItem(  
 icon: ImageIcon(AssetImage('assets/icons/profileBarIcon.png')),  
 label: 'Профиль'  
 ),  
 BottomNavigationBarItem(  
 icon: ImageIcon(AssetImage('assets/icons/orders.png')),  
 label: 'Заказы'  
 ),  
 BottomNavigationBarItem(  
 icon: ImageIcon(AssetImage('assets/icons/cartBarIcon.png')),  
 label: 'Корзина'  
 )  
 ]  
 ),  
 );  
}

Листинг 1 – Создание нижнего навигационного меню

Пакет «Models» хранит в себе все классы, используемые в приложении. Дан-ный пакет включает в себя такие классы как «User», «Recipe», «Ingredient» и «Comment». Каждый из них содержит поля, хранящие информацию, конструкторы, а также методы для преобразования из json-формата в объекты класса.

Пакет «Utils» хранит в себе классы для хранения функций с дополнительным функционалом приложения и всеми запросами к серверному приложению (см. таб-лицу 3.2).

**Таблица 3.2 – Содержание пакета «Utils»**

|  |  |
| --- | --- |
| Класс DbHandler – список задач | |
| fetchUsers() | Получение всех пользователей |
| fetchPizzas() | Получение всех пицц |
| addUser() | Добавление пользователя |
| addOrder() | Добавление заказа |
| addPizza() | Добавление пиццы |
| addOrderDetails() | Добавление предмета заказа |
| login() | Вход пользователя |
| getUserByName() | Получение пользователя по его никнейму |
| updateUser() | Обновление данных пользователя |
| updateOrderStatus() | Обновление статуса доставки |
| getUserAddresses() | Получение адресов пользователя |
| getUserOrders() | Получение заказов пользователя |
| getOrders() | Получение заказов |
| getOrderItems() | Получение предметов заказа(по id) |
| getAllOrderItems() | Получение всех предметов заказа |
| addAddress() | Добавление адреса |
| deleteUserAddress() | Удалить адрес пользователя |
| deletePizza() | Удалить пиццу |
| getPizzaDetails() | Получить детали пицц |
| Класс SQLiteHandler – список задач | |
| clearPizzas() | Очистка таблицы с пиццами |
| clearHistory() | Очистка таблицы с заказами |
| getOrders() | Получение заказов |
| getOrderItems() | Получение предметов заказа |
| addOrder() | Добавление заказа |
| addOrderItem() | Добавление предмета заказа |
| addPizza() | Добавление пиццы |
| getPizzasInCart() | Получение пицц из корзины |
| incrementQuantity() | Увеличение количества пиццы |
| decrementQuantity() | Уменьшение количества пиццы |

Таким образом, были рассмотрены все основные компоненты мобильного приложения.

## **3.4 Разработка сервера**

Сервер было реализовано в среде разработки – JetBrains WebStorm. Для написания приложения была использована технология ExpressJs.

Графическая структура проекта представлена на рисунке 3.3.

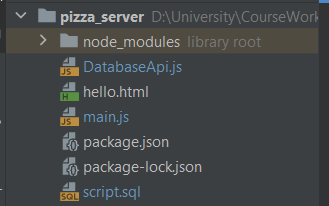


Рисунок 3.3 – Графическая структура серверного приложения

В файлах «DatabaseApi.js» находятся все функции для работы с базой данных, то есть вызов процедур из БД. В листинге 2 представлен пример подключения базы данных к серверному приложению:

constructor() {

this.pool = new postgres.Pool({

user: 'admin',

host: 'localhost',

database: 'pizzaria',

password: 'admin',

port: 5432,

});

}

Листинг 2 – Подключение к базе данных

Таким образом, были рассмотрены все основные компоненты серверного приложения.

## **3.5 Разработка веб-приложения**

Веб-приложение представляет из себя приложение с таким же функционалом, как и мобильное, так как разработка велась с помощью Flutter — фреймворк для разработки кроссплатформенных приложений.

## **3.6 Вывод по главе** **«Программная реализация приложений»**

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.

Третий этап – это разработка серверного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы.

Четвёртым этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

Заключительный этап - разработка веб-приложения, где была представлена графическая структура приложения, рассмотрены основные файлы приложения.

# **4. Обеспечение безопасности информационной системы**

При создании сервера и использованием аутентификации в приложении хорошей практикой будет шифрование пароля пользователя перед сохранением в базу данных.

В данной курсовой работе была использована библиотека crypto, которая добавляет модификатор входа хэш-функции, или же «соль», к паролю при помощи алгоритма SHA256 шифрует его.

Модификатор входа хэш-функции — строка данных, которая передаётся хеш-функции вместе с входным массивом данных для вычисления хэша. Используется для усложнения определения прообраза хэш-функции методом перебора по словарю возможных входных значений, включая атаки с использованием радужных таблиц.

Shared Preferences – это способ хранения и извлечения небольших объемов примитивных данных в виде пар ключ/значение в файл в хранилище устройства, таких как String, integer, float, Boolean, которые составляют ваши настройки в файле XML внутри приложения. на хранилище устройства.

## **4.1 Вывод по главе «Обеспечение безопасности информационной системы»**

В данной главе было рассмотрено обеспечение безопасности информационной системы проекта.

Так как в данном курсовом проекте присутствует авторизация пользователя, то было решено использовать хеширование паролей в качестве обеспечения информационной безопасности системы. Также были добавлены Shared Preferences для аутентификации пользователя.

# **5. Руководство пользователя**

Изначально открывается экран, на котором находится форма входа (рисунок 5.1).

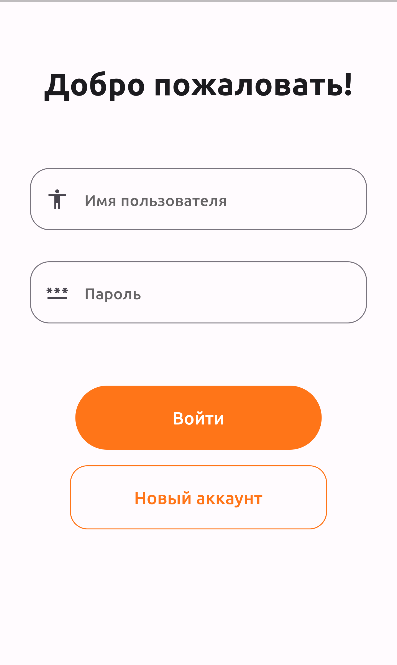


Рисунок 5.1 – Страница входа

Перед первым использованием приложения Вам необходимо зарегистрироваться. Для этого нужно нажать на подпись «Новый аккаунт», в открывшуюся форму ввести свой логин и пароль. После ввода данных Вам необходимо нажать кнопку «Создать». После успешной регистрации необходимо войти.

Для входа в аккаунт необходимо ввести логин и пароль, после чего нажать кнопку «Войти». После успешной авторизации Вы входите в приложение.

Сразу открывается главный экран, в котором отображаются все пиццы. Интерфейс представлен на рисунке 5.2.



Рисунок 5.2 – Главная страница

При нажатии на кнопку поиска появляется поисковая строка. Снизу располагается навигационная панель. Отображение страницы профиля можно увидеть на рисунке 5.3.

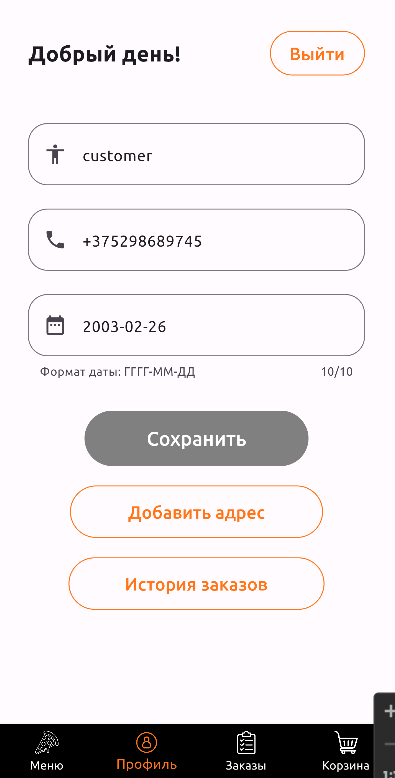


Рисунок 5.3 – Страница профиля

Добавление адреса и просмотр истории заказов можно увидеть на рисунках 5.4 и 5.5.

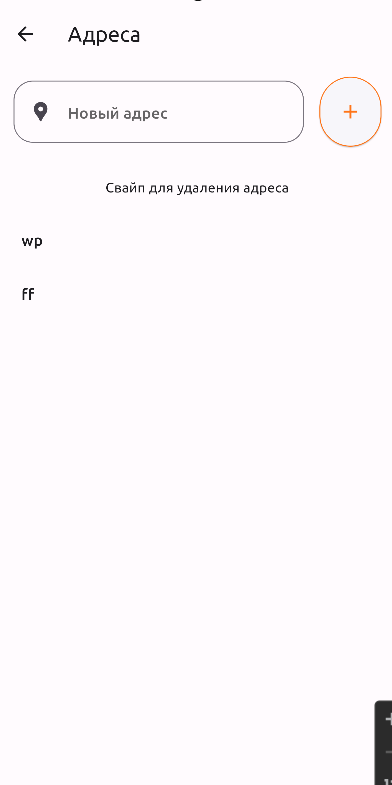


Рисунок 5.4 – Страница с добавлением адреса

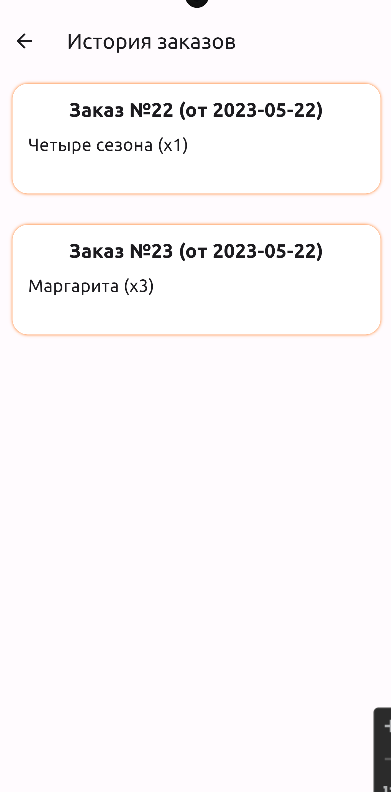


Рисунок 5.5 – Страница истории заказов

При нажатии на пиццу в списке появляется конфигуратор и при нажатии на кнопку добавить товар добавляется в корзину. Страницы пиццы и корзины можно увидеть на рисунках 5.6 и 5.7.



Рисунок 5.6 – Страница пиццы

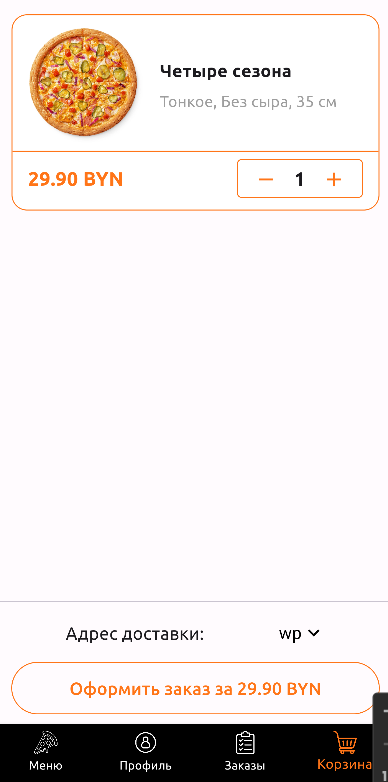


Рисунок 5.7 – Страница корзины

При нажатии кнопки Оформить заказ он добавляется в доставку и выводится на странице Заказы. Страницу Заказы можно увидеть на рисунке 5.8.

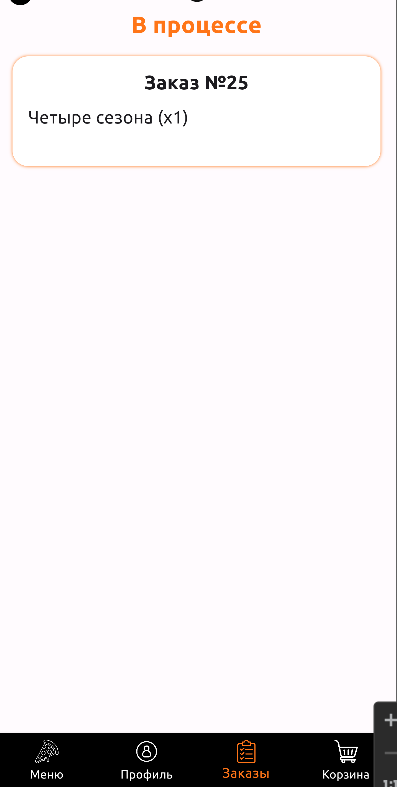


Рисунок 5.8 – Страница заказов

На рисунках 5.9 и 5.10 представлен примеры страниц для админа с отображаемым список пицц, кнопками добавления пиццы, выхода и полями для заполнения информации о пицце.



Рисунок 5.9 – Страница админа, список пицц

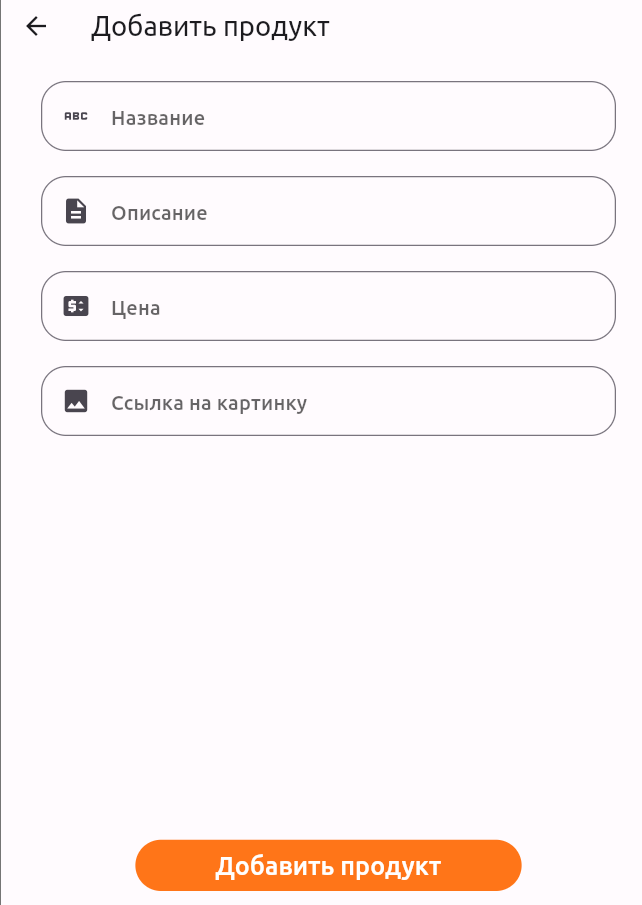


Рисунок 5.10 – Страница админа, добавление пиццы

На рисунке 5.11 представлен пример страниц доставщика, который с имеет список заказов и с помощью свайпа может отметить заказ как выполненный.

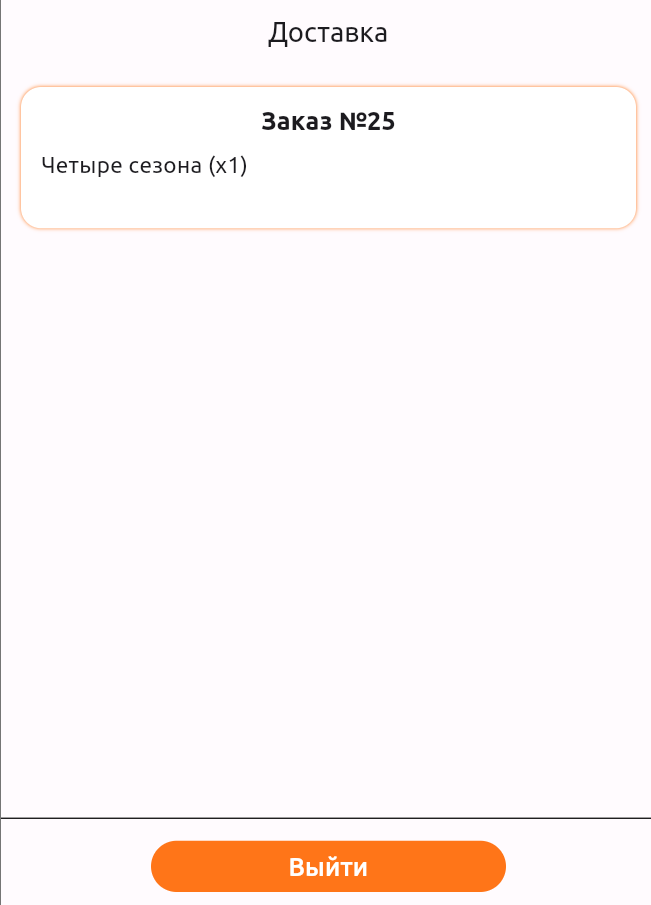


Рисунок 5.11 – Страница доставщика

Таким образом, были рассмотрены все основные функции приложения, а также описано как пользоваться приложением.

## **5.1 Вывод по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью курсового проекта было предоставить пользователям возможность удобного создавать и хранить все интересные пользователю рецепты.

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 мобильном 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. При проектировании веб-приложения был определен основной функционал.

– реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQАLite и PostreSQL, сред разработки Android Studio и JetBrains WebStorm, языков программирования Java и Dart, фреймворка Flutter, ExpressJs.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Pizzaria», который включает в себя мобильное приложение, веб-приложение и сервер.

– обеспечение информационной безопасности системы;

Для обеспечения информационной безопасности была выбрала библиотека crypto, которая позволит шифровать пароли пользователей перед записью в базу данных, что сохранит личные данные пользователя.

– разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

* регистрация пользователя;
* авторизация пользователя;
* выбор конфигурации пиццы и добавление в корзину;
* добавление и удаление пиццы;
* добавление и удаление новые адреса;
* просмотр истории заказов;
* оформление заказа на доставку;
* отметка доставки заказа.

# **Список использованных источников**

1. Интеграция с базами данных [Электронный ресурс]. / expressjs.com – Режим доступа: https://expressjs.com/ru/guide/database-integration.html
2. Http. Flutter Package [Электронный ресурс] / pub.dev/packages. – Режим доступа: https://pub.dev/packages/http
3. Архитектура REST [Электронный ресурс] – habr.com – Режим доступа: https://habr.com/ru/post/38730/
4. Solutions on Stackoverflow [Электронный ресурс] – Режим доступа: https://stackoverflow.com.

# **ПРИЛОЖЕНИЕ А**

CREATE TABLE Pizzas(

id INTEGER PRIMARY KEY,

name TEXT,

size TEXT,

dough TEXT,

cheese TEXT,

price REAL,

image TEXT

)

CREATE TABLE Cart (

id INTEGER PRIMARY KEY AUTOINCREMENT,

quantity INTEGER NOT NULL DEFAULT 1,

pizza\_id INTEGER NOT NULL,

FOREIGN KEY (pizza\_id) REFERENCES Pizzas(id)

)

CREATE TABLE Orders (

id INTEGER PRIMARY KEY,

user\_id INTEGER NOT NULL,

delivery\_id INTEGER NOT NULL,

created\_at TEXT NOT NULL,

status TEXT NOT NULL

)

CREATE TABLE OrderItems (

id INTEGER PRIMARY KEY,

order\_id INTEGER NOT NULL,

product\_id INTEGER NOT NULL,

pizza\_details\_id INTEGER NOT NULL,

quantity INTEGER NOT NULL

)

# **ПРИЛОЖЕНИЕ Б**

CREATE TABLE Roles (

id serial PRIMARY KEY,

name text

)

CREATE TABLE Users (

id serial PRIMARY KEY,

role\_id integer NOT NULL REFERENCES Roles(id),

username text NOT NULL UNIQUE,

password text NOT NULL,

phone\_number text DEFAULT(NULL),

birthday date DEFAULT(NULL)

)

CREATE TABLE DeliveryAddresses (

id serial PRIMARY KEY,

user\_id integer NOT NULL REFERENCES Users(id),

address text NOT NULL

)

CREATE TABLE Pizzas (

id serial PRIMARY KEY,

name text NOT NULL UNIQUE,

description text,

price numeric(8,2) NOT NULL,

image text

)

CREATE TABLE Orders (

id serial PRIMARY KEY,

user\_id integer NOT NULL REFERENCES Users(id),

delivery\_id integer NOT NULL REFERENCES DeliveryAddresses(id),

created\_at timestamp NOT NULL,

status text NOT NULL

)

CREATE TABLE PizzaDetails (

id serial PRIMARY KEY,

size text NOT NULL,

dough text NOT NULL,

cheese text NOT NULL

)

CREATE TABLE OrderDetails (

id serial PRIMARY KEY,

order\_id integer NOT NULL REFERENCES Orders(id),

product\_id integer NOT NULL REFERENCES Pizzas(id),

pizza\_details\_id integer NOT NULL REFERENCES PizzaDetails(id),

quantity integer NOT NULL DEFAULT(1)

)