**Содержание**

[Введение 5](#_Toc167086078)

[1 Аналитический обзор литературы 6](#_Toc167086079)

[1.1 Аналог «PapaJohns» 6](#_Toc167086080)

[1.2 Аналог «FoxPizza» 7](#_Toc167086081)

[1.3 Аналог «DoDoPizza» 8](#_Toc167086082)

[1.4 Выводы 9](#_Toc167086083)

[2 Проектирование приложения 9](#_Toc167086084)

[2.1 Проектирование базы данных 9](#_Toc167086085)

[2.2 Описание основных информационных объектов и ограничений целостности](#_Toc167086086)

[**2.2.1 Коллекция user\_detail** 10](#_Toc167086087)

[**2.2.2 Коллекция menu\_item\_detail** 10](#_Toc167086088)

[**2.2.3 Коллекция offer\_detail** 11](#_Toc167086089)

[**2.2.4 Коллекция order\_detail** 11](#_Toc167086090)

[2.3 Проектирование мобильного приложения 12](#_Toc167086091)

[2.4 Выводы 13](#_Toc167086092)

[3 Реализация приложения 13](#_Toc167086093)

[3.1 Технические средства разработки 14](#_Toc167086094)

[3.2 Разработка мобильного приложения 15](#_Toc167086095)

[3.3 Выводы 16](#_Toc167086096)

[4 Тестирование приложения 16](#_Toc167086097)

[4.1 Выводы 18](#_Toc167086098)

[5 Руководство по использованию 18](#_Toc167086099)

[5.1 Руководство гостя 19](#_Toc167086101)

[5.2 Руководство пользователя 23](#_Toc167086102)

[5.3 Выводы 27](#_Toc167086103)

[Заключение 28](#_Toc167086104)

[Список использованных источников 29](#_Toc167086105)

[ПРИЛОЖЕНИЕ А. UML диаграмма использований 30](#_Toc167086106)

[ПРИЛОЖЕНИЕ Б. Класс Globs 31](#_Toc167086107)

# Введение

Мобильные приложения играют важную роль в современном бизнесе. Они обеспечивают удобство и доступность для клиентов, позволяя им взаимодействовать с брендом в любое время и в любом месте. Это особенно важно в эпоху цифровизации, когда большинство людей используют смартфоны для доступа к интернету. Мобильные приложения также могут улучшить взаимодействие с клиентами, предоставляя персонализированный опыт и улучшая обслуживание клиентов.

Мобильное приложение для пиццерии может принести множество преимуществ. Во-первых, оно может увеличить продажи, предоставляя клиентам удобный способ просмотра меню и заказа пиццы. Во-вторых, приложение может помочь укрепить отношения с клиентами, предлагая им персонализированные рекомендации и специальные предложения, такие как скидки на любимые блюда или бонусы за частые заказы. Наконец, мобильное приложение может собирать ценную информацию о предпочтениях клиентов, которую можно использовать для улучшения меню и обслуживания. В целом, мобильное приложение может стать мощным инструментом для увеличения конкурентоспособности и роста вашего бизнеса.

Были поставлены следующие цели разработки проекта:

* Реализовать функции пользователя и администратора;
* Реализовать регистрацию и авторизацию пользователей;
* Обеспечить управление товарами;
* Реализовать корзину пользователя;
* Обеспечить управление профилем для пользователей.

Основной целью курсового проекта является разработка приложения пиццерии. В качестве языка был выбран Dart в связке с фреймворком Flutter. В качестве провайдера авторизации используется удаленный сервер MySQL, а взаимодействие пользовательского интерфейса и сервера отвечает NodeJS.

# 1 Аналитический обзор литературы

Обзор аналогов является критически важным этапом в процессе разработки программного обеспечения. Он позволяет разработчикам получить представление о существующих решениях, их функциональности, преимуществах и недостатках. Это помогает избежать повторения ошибок, сделанных другими, и определить уникальные особенности, которые можно внедрить в новое ПО. Аналоги рассмотрены ниже.

## Аналог «PapaJohns»

Приложение PapaJohns - это официальное мобильное приложение, разработанное для удобного заказа пиццы и других блюд на телефонах и планшетах, представлено на рисунке 1.1. В приложении PapaJohns представлен полный ассортимент меню, включая разнообразные пиццы, закуски, десерты и напитки. Вы найдете подробные описания блюд, фотографии, отзывы клиентов и актуальные цены.

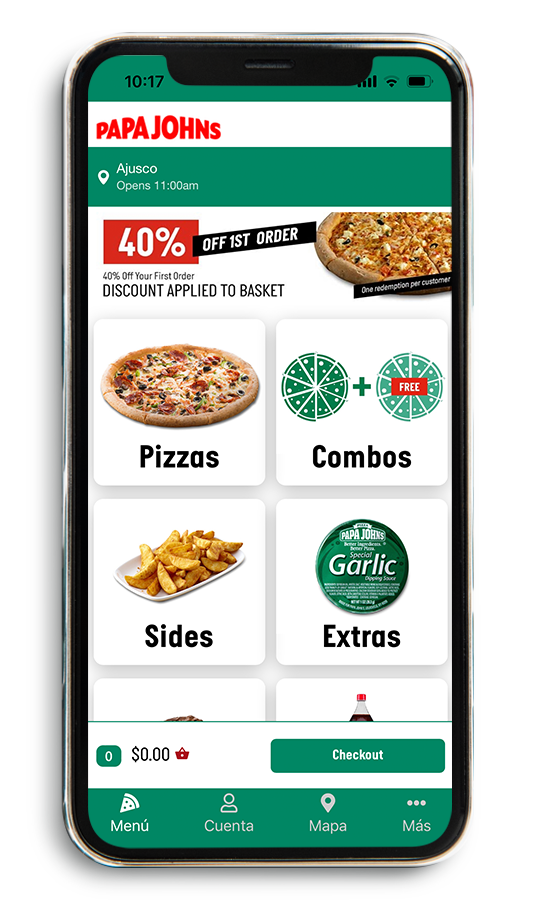


Рисунок 1.1 – Приложение «PapaJohns»

Приложение PapaJohns обладает рядом преимуществ, которые делают его незаменимым инструментом для любителей пиццы. Оно предлагает удобный и быстрый способ заказа с доставкой на дом или самовывозом. Интерфейс приложения адаптирован под мобильные устройства, что делает его простым и понятным в использовании. Вы также можете воспользоваться функцией персонализированных рекомендаций и специальных предложений, чтобы получить скидки на любимые блюда и бонусы за частые заказы.

Однако, несмотря на все его преимущества, приложение PapaJohns имеет и свои ограничения. Некоторые функции, доступные на веб-сайте, могут отсутствовать в мобильном приложении. Это означает, что пользователи могут не получить полный функционал, доступный на веб-сайте.

В целом, приложение PapaJohns предоставляет пользователям удобный и эффективный способ заказа пиццы, улучшая их опыт и удовлетворение.

## 1.2 Аналог «FoxPizza»

Мобильное приложение FoxPizza - это официальное приложение от сети пиццерий FoxPizza, представленное на рисунке 1.2, предназначенное для удобного заказа пиццы и других блюд на мобильных устройствах. Приложение FoxPizza предоставляет полный каталог меню с подробными описаниями блюд, фотографиями и отзывами клиентов. Дизайн приложения напоминает сайт, что обеспечивает привычный и удобный интерфейс для пользователей.

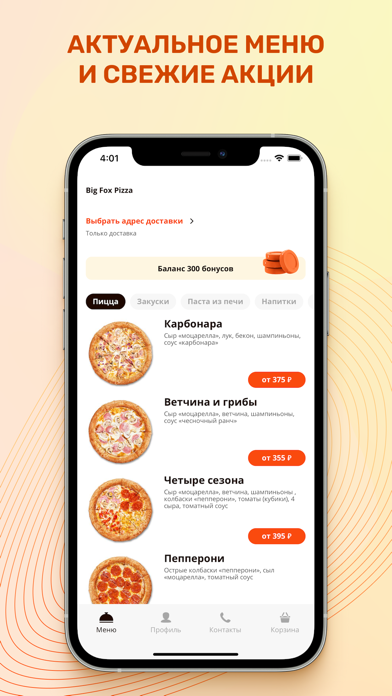


Рисунок 1.2 – Приложение «FoxPizza»

Среди преимуществ приложения FoxPizza можно выделить быстрый и удобный поиск блюд, что позволяет пользователям легко находить и заказывать любимые пиццы и закуски. Приложение также предлагает инструмент для сравнения блюд, что помогает в выборе. Пользователи могут отслеживать изменения цен и наличие акционных предложений, добавляя их в список избранного. Кроме того, приложение поддерживает персонализированные рекомендации и специальные предложения, что делает процесс заказа еще более выгодным и приятным.

Однако, как и любое другое приложение, мобильное приложение FoxPizza имеет свои недостатки. Несмотря на то, что приложение доступно для пользователей Android, релиз iOS-версии приложения произошел только в конце осени 2023 года, что было неудобно для пользователей Apple. В целом, приложение FoxPizza предоставляет удобный и эффективный способ заказа пиццы, улучшая пользовательский опыт и удовлетворение клиентов.

## 1.3 Аналог «DoDoPizza»

Мобильное приложение DoDoPizza — это официальное приложение от сети пиццерий DoDoPizza, разработанное для удобного заказа пиццы и других блюд на мобильных устройствах, представлено на рисунке 1.3. Приложение DoDoPizza представляет полный ассортимент меню с подробными описаниями блюд, фотографиями и отзывами клиентов. Дизайн приложения выполнен в узнаваемом стиле DoDoPizza, что обеспечивает привычный и удобный интерфейс для пользователей.

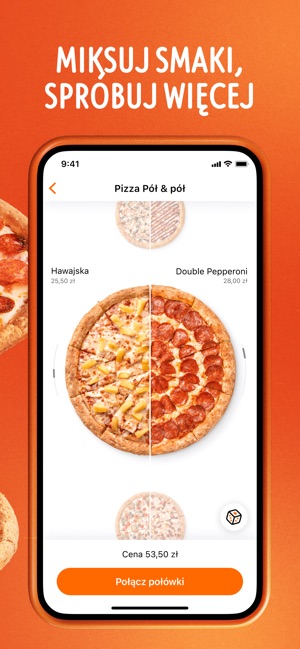


Рисунок 1.3 – Приложение «DoDoPizza»

Среди преимуществ приложения DoDoPizza можно выделить быстрый и удобный заказ пиццы с возможностью выбора различных размеров, начинок и добавок. Приложение также предлагает специальные акции и скидки для пользователей, делая процесс заказа еще более привлекательным. Пользователи могут отслеживать статус своего заказа в режиме реального времени и оплачивать его удобным способом, включая онлайн-оплату или оплату при получении.

Однако, как и у любого другого приложения, у мобильного приложения DoDoPizza есть свои ограничения. Некоторые функции могут быть недоступны в зависимости от региона или времени заказа. Однако команда разработчиков постоянно работает над улучшением приложения и добавлением новых возможностей для удовлетворения потребностей клиентов. В целом, приложение DoDoPizza предоставляет простой, быстрый и удобный способ заказа пиццы, делая опыт заказа еще более приятным для клиентов.

## 1.4 Выводы

Были рассмотрены аналоги проекта. После проведения анализа, стало ясно, что для создания хорошего приложения для пиццерии важен красивый и интуитивный дизайн приложения а также обширный функционал.

# 2 Проектирование приложения

Проектирование программного обеспечения - это процесс создания плана для разработки программного продукта. Оно включает в себя определение функциональности, структуры, архитектуры и интерфейсов программного продукта. Проектирование ПО также учитывает такие факторы, как производительность, надежность, удобство использования и безопасность. Этот процесс является ключевым этапом в жизненном цикле разработки программного обеспечения и служит основой для написания кода.

## Проектирование базы данных

Проектирование модели данных для NoSQL базы данных - это процесс определения структуры данных, которая наиболее эффективно отвечает требованиям приложения. В отличие от традиционных реляционных баз данных, NoSQL базы данных обычно не используют структуру таблиц и не требуют жесткой схемы данных. Это позволяет создавать более гибкие и масштабируемые модели данных. При проектировании модели данных для NoSQL базы данных важно учитывать тип базы данных (документо-ориентированная, колоночная, ключ-значение или графовая), требования к производительности, масштабируемости и надежности, а также специфику приложения, которое будет использовать эту базу данных.

Для данного курсового проекта была спроектирована база данных, логическая модель базы данных представлена на рисунке 2.1.

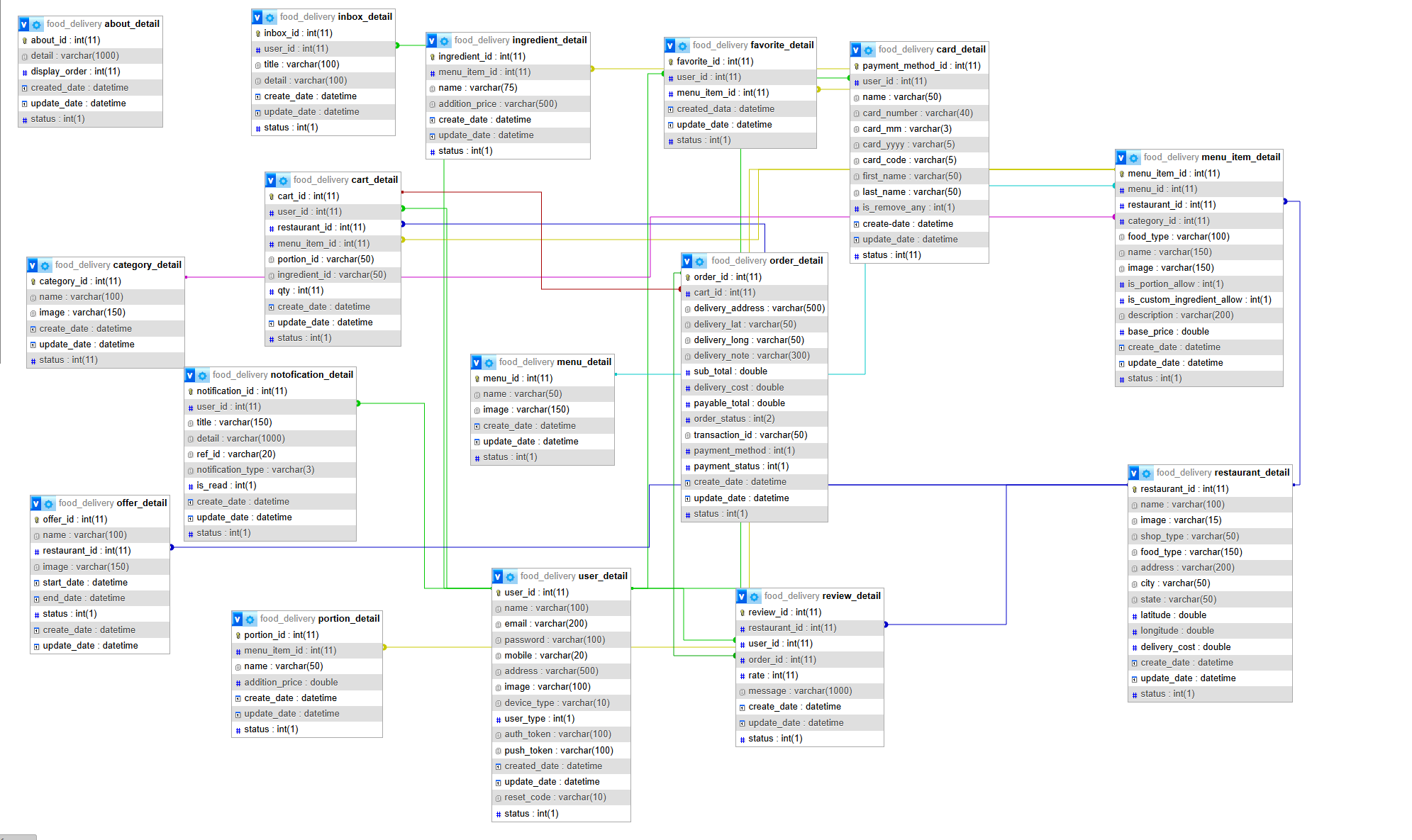


Рисунок 2.1 – Модель БД

Созданная база данных содержит в себе 16 коллекции: user\_detail, Restaurants Detail, Offer List Detail, Category Detail, Menu Detail, Menu Item Detail, Portion Detail, Ingredient Detail, Favorite Detail, Review Detail, Cart Detail, Order Detail, User Payment Method Detail, Notification Detail, Inbox Detail, About Detail.

2.2 Описание основных информационных объектов и ограничений целостности

Далее будет описана структура основных коллекций. В коллекции users хранятся данные для идентификации пользователей, зарегистрированных в приложении, menu\_item\_detail – данные товарах, favorites для хранения избранных товаров и orders для хранения заказов.

### **2.2.1 Коллекция user\_detail**

Коллекция состоит из следующих полей:

* id: Уникальный идентификатор пользователя.
* email: Уникальный email пользователя.
* name: Имя пользователя.
* imageUrl: Ссылка на фотографию профиля пользователя.
* role: Роль пользователя.

### **2.2.2 Коллекция menu\_item\_detail**

Коллекция menu\_item\_detail состоит из следующих полей:

* menu\_item\_id: Уникальный идентификатор пункта меню.
* menu\_id: Идентификатор меню.
* restaurant\_id: Идентификатор ресторана.
* category\_id: Идентификатор категории блюда.
* food\_type: Тип блюда.
* name: Название блюда.
* image: Ссылка на изображение блюда.
* is\_portion\_allow: Флаг, позволяющий указать, доступна ли порция блюда.
* is\_custom\_ingredient\_allow: Флаг, позволяющий указать, доступен ли выбор ингредиентов пользователем.
* description: Описание блюда.
* base\_price: Базовая цена блюда.
* create\_date: Дата создания записи.
* update\_date: Дата последнего обновления записи.
* status: Статус записи.

### **2.2.3 Коллекция offer\_detail**

Коллекция offer\_detail состоит из следующих полей:

* offer\_id: Уникальный идентификатор предложения.
* name: Название акции.
* restaurant\_id: Идентификатор ресторана, предлагающего акцию.
* image: Ссылка на изображение акции.
* start\_date: Дата начала акции.
* end\_date: Дата окончания акции.
* status: Статус акции.
* create\_date: Дата создания записи.
* update\_date: Дата последнего обновления записи.

### **2.2.4 Коллекция order\_detail**

Коллекция orders состоит из следующих полей:

* order\_id: Уникальный идентификатор заказа.
* cart\_id: Идентификатор корзины.
* delivery\_address: Адрес доставки.
* delivery\_lat: Широта места доставки.
* delivery\_long: Долгота места доставки.
* delivery\_note: Примечание к заказу.
* sub\_total: Подытог заказа.
* delivery\_cost: Стоимость доставки.
* payable\_total: Общая сумма к оплате.
* order\_status: Статус заказа.
* transaction\_id: Идентификатор транзакции.
* payment\_method: Метод оплаты.
* payment\_status: Статус оплаты.
* create\_date: Дата создания заказа.
* update\_date: Дата последнего обновления записи.
* status: Статус заказа.

## 2.3 Проектирование мобильного приложения

Проектирование мобильных приложений - это сложный процесс, который включает в себя несколько ключевых аспектов, включая проектирование страниц и навигацию.

Проектирование страниц в мобильных приложениях требует учета ограниченного пространства экрана и специфики взаимодействия пользователя с мобильным устройством. Каждая страница должна быть интуитивно понятной и содержать только самую важную информацию, чтобы не перегружать пользователя. Дизайн должен быть простым и чистым, с хорошим балансом между текстом, изображениями и пространством.

Навигация в мобильных приложениях также играет важную роль. Она должна быть простой и интуитивной, чтобы пользователи могли легко переходить от одной страницы к другой. Обычно используются стандартные элементы навигации, такие как меню, вкладки и кнопки “Назад”. Однако важно учитывать контекст приложения и потребности пользователей при проектировании системы навигации.

При первом входе в приложения пользователя встречает экран, описывающий как пользоваться приложением. После этого он попадает на главный экран.

На главном экране представлены текущие предложения с возможностью перейти к каталогу товаров. В нем пользователь может выбрать товар, чтобы перейти на его страницу. Также, если пользователь является администратором, то он может перейти на страницу добавления нового товара. Внизу отображается таб-бар с навигацией на страницу профиля и корзины.

На странице товара представлена информация о товаре, с возможностью добавить товар в корзину и избранное. Администратор может изменить или удалить товар.

На странице корзины отображаются товары корзины, и если пользователь авторизован, то он может перейти на страницу оформления заказа для того, чтобы подтвердить его.

На странице профиля отображается текущая информация о пользователе или гостевая информация. Если пользователь авторизован, то он может просмотреть историю заказов. Также есть возможность просмотреть текущее избранные товары. Для авторизованных пользователей отображается возможность выйти, а для неавторизованных – войти. На странице входа можно перейти на страницу регистрации. Авторизованные пользователи могут редактировать свое имя и менять фотографию профиля.

В профиле пользователь может просмотреть свои персональные данные, а также редактировать профиль и выйти из аккаунта.

Страница приветствия показывается всем пользователям при первом входе.

На рисунке 2.3 представлена схема навигации пользователя.

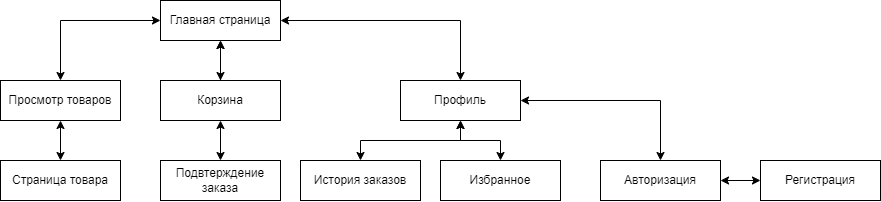


Рисунок 2.3 – Схема навигации для пользователя

На рисунке 2.4 представлена схема навигации администратора.

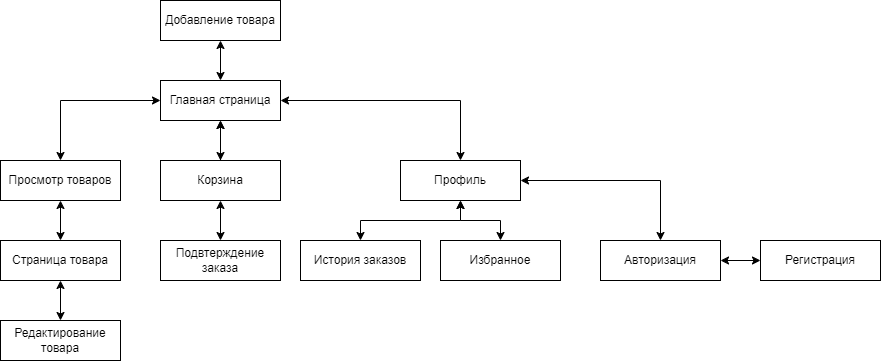


Рисунок 2.4 – Схема навигации администратора

Диаграмма вариантов использований представлена в приложении А.

## 2.4 Выводы

После выполнения проектирования приложения, была получена вся требуемая информация для реализации проекта. Были спроектированы модели базы данных, а также структура мобильного клиента.

# 3 Реализация приложения

Реализация приложения - это процесс преобразования проекта программного обеспечения в работающий продукт. Это включает в себя написание кода, интеграцию различных компонентов системы, тестирование для обнаружения и исправления ошибок, а также оптимизацию для улучшения производительности. Реализация приложения требует глубоких технических знаний и навыков в области программирования, а также понимания бизнес-требований и пользовательских потребностей. Это ключевой этап в жизненном цикле разработки программного обеспечения, который влияет на качество и успешность конечного продукта.

## 3.1 Технические средства разработки

Для реализации курсового проекта будет использован язык программирования Dart вместе с фремворком Flutter.

Dart - это язык программирования, разработанный компанией Google. Он был представлен в 2011 году и позиционируется как замена или альтернатива JavaScript. Dart - это объектно-ориентированный, веб-фреймворк, сценарный, императивный, рефлексивный язык программирования, язык функционального программирования и мультипарадигмальный [1].

Основной сферой применения Dart на сегодняшний день является разработка графических приложений с помощью кроссплатформенного фреймворка Flutter. Dart предлагает современные функции, такие как безопасность от null и сопоставление с образцом. Dart также поддерживает горячую перезагрузку, что позволяет видеть результаты изменений в коде немедленно в работающем приложении.

Flutter - это открытый UI-инструментарий, разработанный Google для создания красивых, компилируемых нативно приложений для мобильных устройств, веб и настольных компьютеров из одной кодовой базы [2]. Flutter работает с существующим кодом и используется разработчиками и организациями по всему миру.

Основные преимущества Flutter включают быструю разработку благодаря возможности горячей перезагрузки, которая позволяет видеть результаты изменений в коде практически мгновенно, не теряя состояния. Кроме того, Flutter позволяет контролировать каждый пиксель для создания настраиваемых, адаптивных дизайнов, которые выглядят и чувствуются отлично на любом экране.

Flutter также поддерживает развертывание на множестве устройств из одной кодовой базы, включая мобильные, веб, настольные и встроенные устройства3. Это делает его мощным инструментом для разработки кросс-платформенных приложений. Flutter поддерживается и используется Google, доверяют ему известные бренды по всему миру и поддерживается сообществом разработчиков из разных стран.

База данных и механизмы аутентификации были разработаны c помощью NodeJS.

Node.js - это кросс-платформенная среда выполнения JavaScript, которая позволяет разработчикам создавать быстрые и масштабируемые серверные приложения. Node.js использует неблокирующую, событийно-ориентированную модель I/O, что делает его легким и эффективным, особенно для приложений, работающих в реальном времени и обрабатывающих большое количество данных.

Express.js - это минималистичный и гибкий фреймворк веб-приложений для Node.js, который предоставляет набор мощных функций для создания одностраничных, многостраничных и гибридных веб-приложений. Express.js упрощает процесс настройки маршрутизации и обработки запросов, предоставляя простой API для создания REST API.

Для аутентификации пользователей в приложении на Node.js и Express.js использовался login.js. Это промежуточное программное обеспечение (middleware), которое предлагает более 500 стратегий аутентификации, включая имя пользователя и пароль, OAuth (Google, Facebook, Twitter) и многие другие.

Для хранения файлов в приложениях на Node.js можно использовать GridFS, который является спецификацией для хранения и извлечения больших файлов, таких как изображения, аудио и видео. Это позволяет легко интегрировать хранение файлов в базу данных и обеспечивает эффективное управление данными.

## 3.2 Разработка мобильного приложения

Структура приложения состоит из нескольких папок: Common, Common\_Widget, Views. Views содержит страницы и bloc-элементы приложения. Common\_Widget содержит основную бизнес-логику приложения.

При реализации приложения использовано асинхронное программирование.

Данные хранятся в удаленном хранилище. После загрузки приложении синхронизируется с БД, что производить проверки правильности введённой информации.

Метод serviceCallLogin выполняет проверку учетных данных пользователя в локальной и удаленной базах данных. Этот метод вызывается после того, как пользователь входит в свой аккаунт. Он начинается с отображения индикатора загрузки, затем отправляет запрос на сервер с использованием ServiceCall.post. Если ответ сервера положительный (статус “1”), метод сохраняет полученные данные пользователя в локальное хранилище и устанавливает флаг успешного входа в систему.

void serviceCallLogin(Map<String, dynamic> parameter) {

Globs.showHUD();

ServiceCall.post(parameter, SVKey.svLogin,

withSuccess: (responseObj) async {

Globs.hideHUD();

if (responseObj[KKey.status] == "1") {

Globs.udSet( responseObj[KKey.payload] as Map? ?? {} , Globs.userPayload);

Globs.udBoolSet(true, Globs.userLogin);

Navigator.pushAndRemoveUntil(context, MaterialPageRoute(

builder: (context) => const OnBoardingView(),

), (route) => false);

} else {

mdShowAlert(Globs.appName,

responseObj[KKey.message] as String? ?? MSG.fail, () {});

}

}, failure: (err) async {

Globs.hideHUD();

mdShowAlert(Globs.appName, err.toString(), () {});

});

Листинг 3.1 – Проверка с БД

После успешной аутентификации, пользователь перенаправляется на экран OnBoardingView, а предыдущие экраны удаляются из стека навигации, что предотвращает возврат к экрану входа при использовании кнопки “Назад”. В случае ошибки или если статус ответа не “1”, отображается сообщение об ошибке.

Когда пользователь добавляет заказ, предполагается, что система сохраняет информацию о заказе в локальной базе данных на устройстве пользователя и в удаленной базе данных. При выходе из системы локальная база данных очищается для обеспечения безопасности и конфиденциальности данных пользователя.

При повторном входе в систему, пользователю отображаются только его заказы, что достигается путем запроса данных о заказах из удаленной базы данных и их загрузки в локальную базу данных на устройстве пользователя. Это обеспечивает актуальность отображаемой информации о заказах.

Листинг класса, который обрабатывает данные о заказах пользователя, представлен в приложении Б.

## 3.3 Выводы

При разработке были применены различные паттерны, характерные для разработки Flutter приложений. Были использованы инструменты для работы с удаленными и локальными базами данных, что способствует высокой производительности и целостности данных.

# 4 Тестирование приложения

Тестирование мобильного программного обеспечения - это процесс проверки функциональности, производительности, безопасности и других ключевых аспектов мобильных приложений перед их выпуском. Это включает в себя проверку приложения на различных устройствах, операционных системах и конфигурациях, чтобы убедиться, что оно работает корректно во всех возможных сценариях. Тестирование может включать в себя автоматические и ручные методы, а также различные типы тестов, такие как функциональное тестирование, тестирование производительности, тестирование безопасности и тестирование пользовательского интерфейса. Цель тестирования мобильного ПО - обеспечить высокое качество приложения и предотвратить проблемы, которые могут негативно повлиять на пользовательский опыт.

Ниже приведен результат тестирования страницы авторизации, рисунок 4.1.

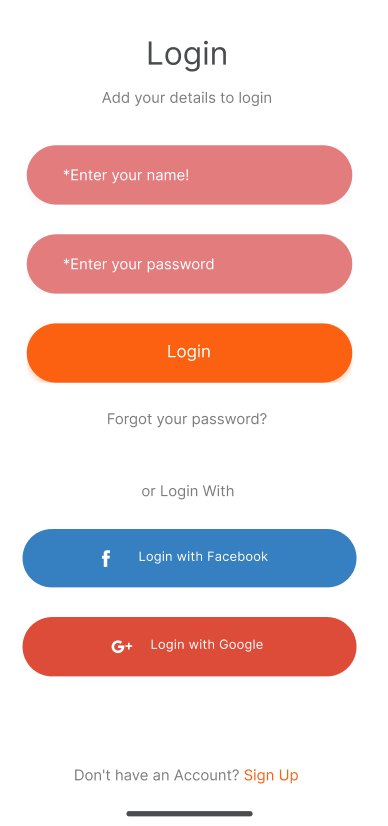


Рисунок 4.1 – Тестирование авторизации

Далее приведен тест страницы регистрации, рисунок 4.2.

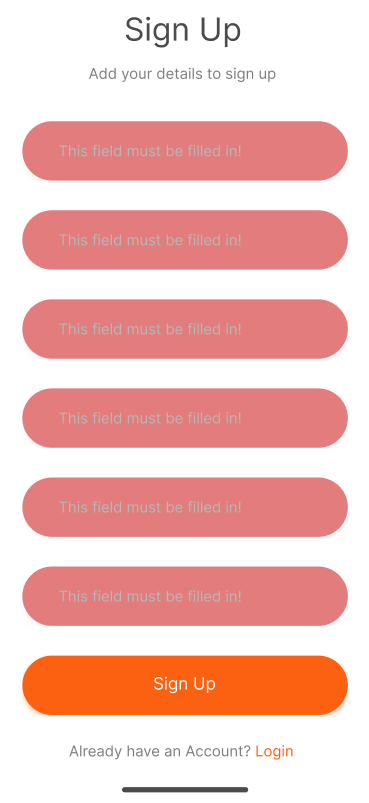


Рисунок 4.2 – Тестирование регистрации

Кроме выше описаных ситуаций, приложение также обрабатывает и другие, например при редактировании или добавлении товара.

## 4.1 Выводы

Приложение было протестировано на возможные сценарии взаимодействия, которые могли бы привести к ошибкам. Приложение обрабатывает такие сценарии корректно.

# 5 Руководство по использованию

# В данном разделе будет описано руководство по использованию для гостя, пользователя и администратора.

## 5.1 Руководство гостя

После экрана приветствия пользователь попадает на главную страницу, рисунок 5.1.

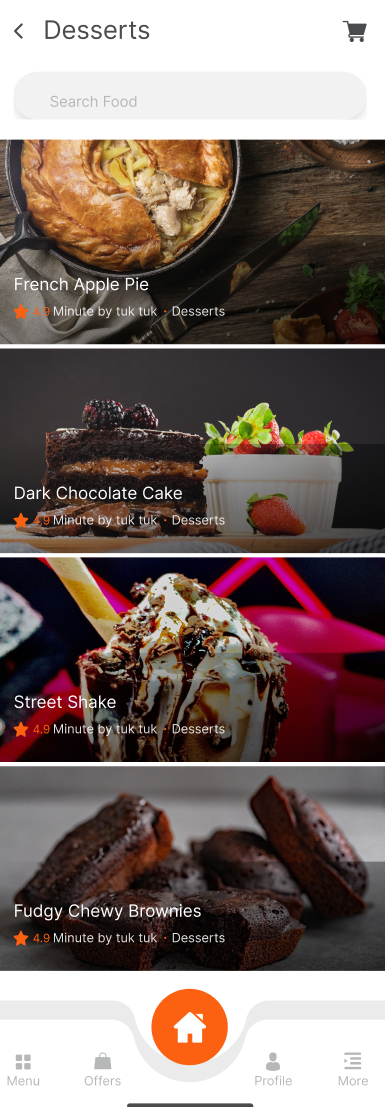


Рисунок 5.1 – Главная страница

Тут пользователь может перейти на страницу каталога, рисунок 5.2.

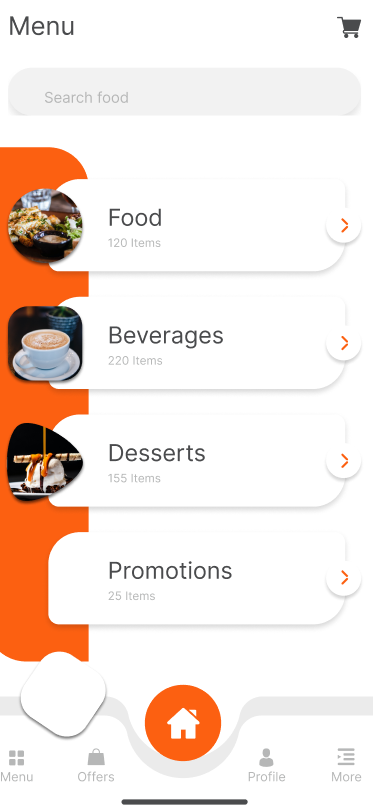


Рисунок 5.2 – Страница каталога

Нажав на товар пользователь может посмотреть информацию о нем открыв страницу товара, где пользователь может добавить его в корзину или в избранное, рисунок 5.3.

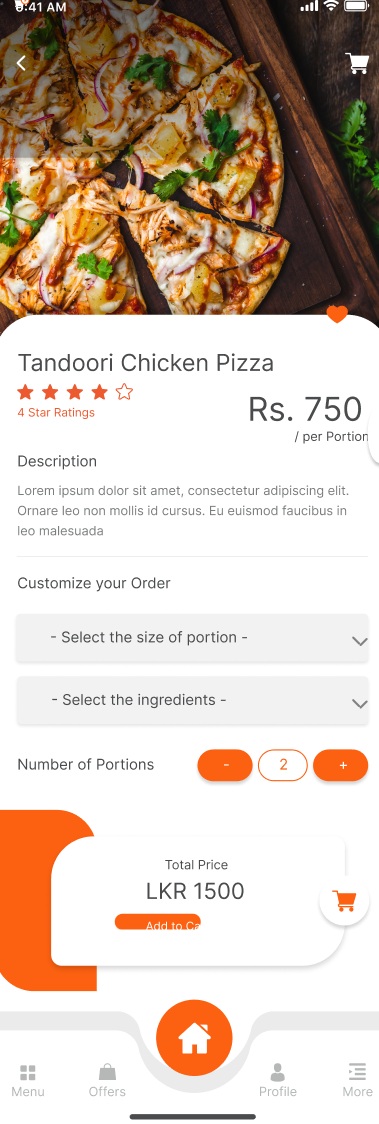


Рисунок 5.3 – Страница товара

Из главной страницы можно перейти к странице профиля, где можно будет выбрать опцию входа. Страница авторизации представлена на рисунке 5.4.

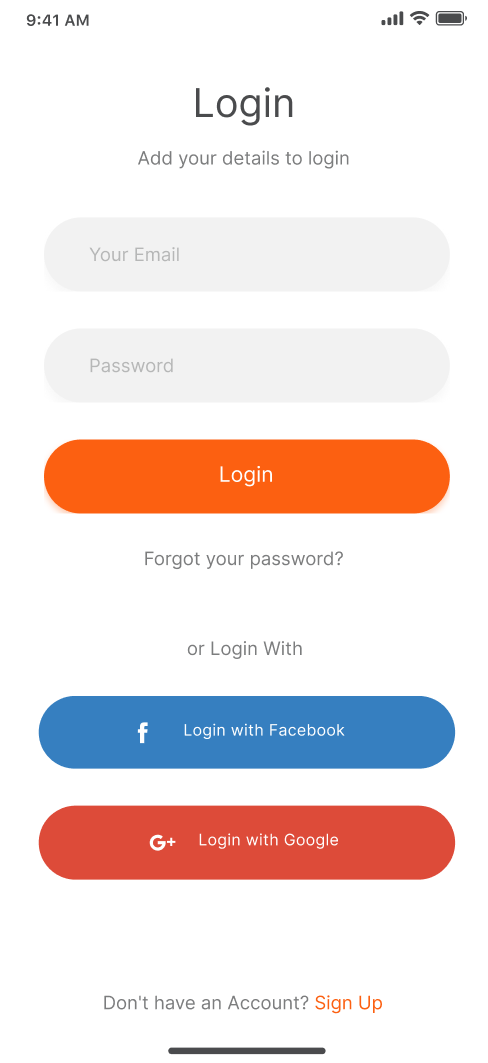


Рисунок 5.4 – Страница авторизации

Нажав на опцию снизу можно перейти к регистрации. Страница регистрации представлена на рисунке 5.5.

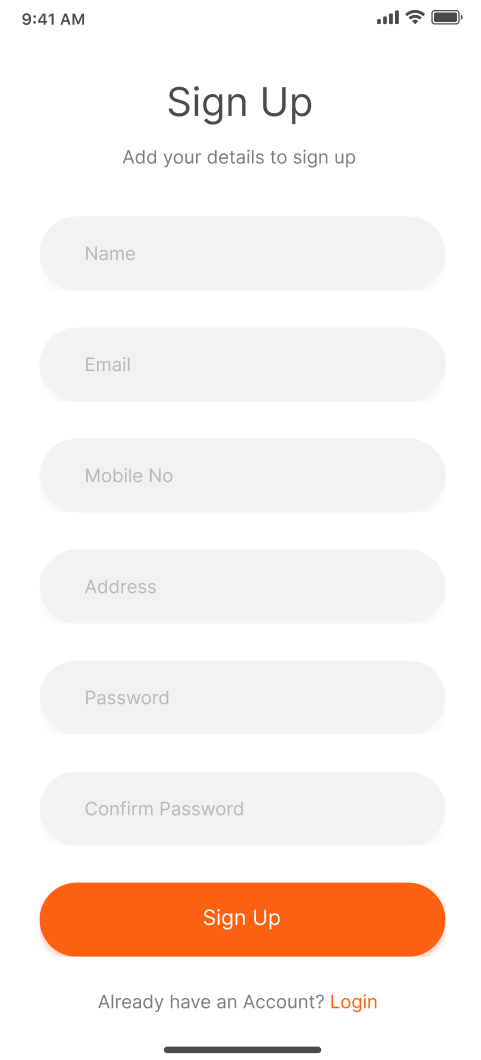


Рисунок 5.5 – Страница регистрации

После успешного входа пользователь приобретает роль пользователя или администратора.

## 5.2 Руководство пользователя

После входа пользователь может изменить свою фотографию, имя, а также посмотреть свои избранные товары и историю заказов, рисунок 5.6.

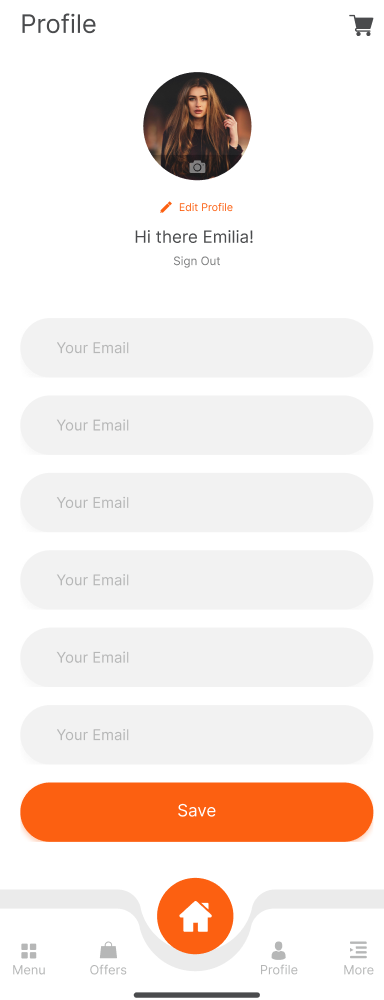


Рисунок 5.6 – Страница профиля

Используя таб-бар снизу, пользователь может перейти к странице своей корзины, рисунок 5.7.

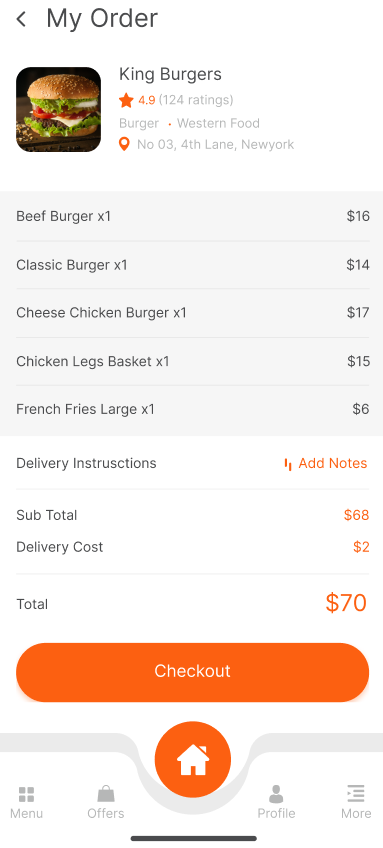


Рисунок 5.7 – Страница корзины

После этого пользователь может перейти к странице подтверждения заказа, рисунок 5.8.

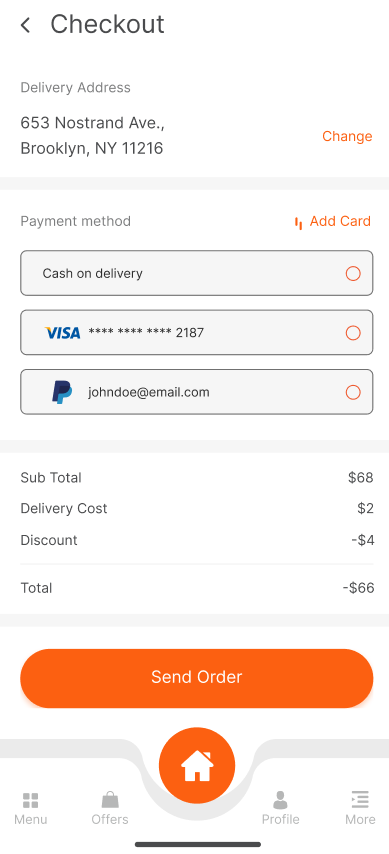


Рисунок 5.8 – Страница подтверждения заказа

После этого пользователю будет представлено подтверждение оплаты, рисунок 5.9.

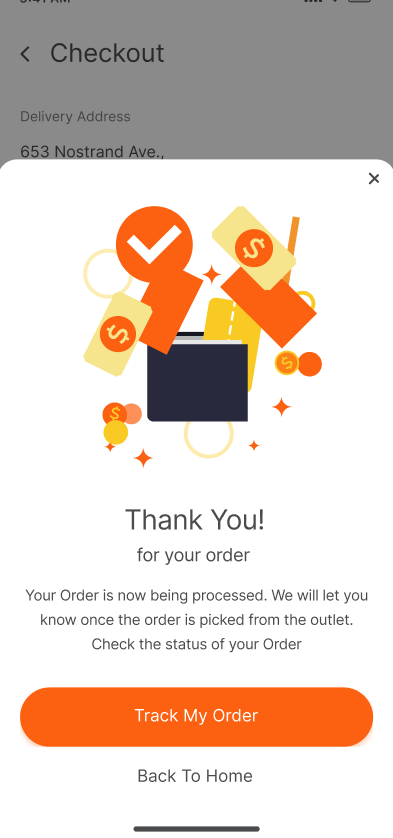


Рисунок 5.9 – Подтверждение оплаты

Нажав на кнопку снизу заказ будет зарегистрирован и появится в истории заказов.

## 5.3 Выводы

Исходя из приведенного выше руководства пользователя ясно, что приложение имеет удобный и понятный интерфейс, что важно для конечных пользователей приложения.

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение пиццерии

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* авторизация;
* просмотр и поиск товаров;
* добавление товаров в корзину, оформление заказа;
* добавление товаров в избанное;
* просмотр своих заказов;
* изменение профиля;

Администратору предоставляет следующие функциональные возможности:

* авторизация;
* просмотр и поиск товаров;
* добавление товаров в корзину, оформление заказа;
* добавление товаров в избанное;
* просмотр своих заказов;
* изменение профиля;
* добавление, удаление и изменение товаров;

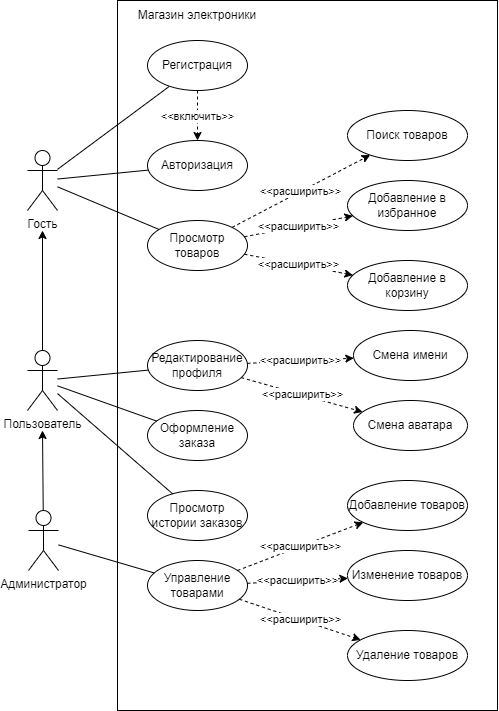
Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованных источников

1. Dart Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dart.dev/guides/ – Дата доступа: 22.03.2024;
2. Flutter Documentation [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Дата доступа: 14.03.2024
3. Firebase Documentation [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs/ – Дата доступа: 20.04.2024;
4. Hive Documentation [Электронный ресурс]. – Режим доступа: https://docs.hivedb.dev/#/ – Дата доступа: 26.04.2024.

# ПРИЛОЖЕНИЕ А. UML диаграмма использований



# ПРИЛОЖЕНИЕ Б. Класс Globs

import 'dart:convert';

import 'package:flutter/services.dart';

import 'package:flutter\_easyloading/flutter\_easyloading.dart';

import 'package:flutter\_timezone/flutter\_timezone.dart';

import 'package:food\_delivery/main.dart';

class Globs {

static const appName = "Food Delivery";

static const userPayload = "user\_payload";

static const userLogin = "user\_login";

static void showHUD({String status = "loading ....."}) async {

await Future.delayed(const Duration(milliseconds: 1));

EasyLoading.show(status: status);

}

static void hideHUD() {

EasyLoading.dismiss();

}

static void udSet(dynamic data, String key){

var jsonStr = json.encode(data);

prefs?.setString(key, jsonStr);

}

static void udStringSet(String data, String key){

prefs?.setString(key, data);

}

static void udBoolSet(bool data, String key) {

prefs?.setBool(key, data);

}

static void udIntSet(int data, String key) {

prefs?.setInt(key, data);

}

static void udDoubleSet(double data, String key) {

prefs?.setDouble(key, data);

}

static dynamic udValue(String key) {

return json.decode(prefs?.get(key) as String? ?? "{}");

}

static String udValueString(String key) {

return prefs?.get(key) as String? ?? "";

}

static bool udValueBool(String key) {

return prefs?.get(key) as bool? ?? false;

}

static bool udValueTrueBool(String key) {

return prefs?.get(key) as bool? ?? true;

}

static int udValueInt(String key) {

return prefs?.get(key) as int? ?? 0;

}

static double udValueDouble(String key) {

return prefs?.get(key) as double? ?? 0.0;

}

static void udRemove(String key) {

prefs?.remove(key);

}

static Future<String> timeZone() async {

try {

return await FlutterTimezone.getLocalTimezone();

} on PlatformException {

return "";

}

}

}

class SVKey {

static const mainUrl = "http://192.168.1.2:3001";

static const baseUrl = '$mainUrl/api/';

static const nodeUrl = mainUrl;

static const svLogin = '${baseUrl}login';

static const svSignUp = '${baseUrl}sign\_up';

static const svForgotPasswordRequest = '${baseUrl}forgot\_password\_request';

static const svForgotPasswordVerify = '${baseUrl}forgot\_password\_verify';

static const svForgotPasswordSetNew = '${baseUrl}forgot\_password\_set\_new';

}

class KKey {

static const payload = "payload";

static const status = "status";

static const message = "message";

static const authToken = "auth\_token";

static const name = "name";

static const email = "email";

static const mobile = "mobile";

static const address = "address";

static const userId = "user\_id";

static const resetCode = "reset\_code";

}

class MSG {

static const enterEmail = "Please enter your valid email address.";

static const enterName = "Please enter your name.";

static const enterCode = "Please enter valid reset code.";

static const enterMobile = "Please enter your valid mobile number.";

static const enterAddress = "Please enter your address.";

static const enterPassword =

"Please enter password minimum 6 characters at least.";

static const enterPasswordNotMatch =

"Please enter password not match.";

static const success = "success";

static const fail = "fail";

}