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:**

**«Приложение для доставки еды «Delivery»»**

Выполнил студент Нечай-Ницевич Денис Павлович

(Ф.И.О.)

Руководитель проекта асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. Кафедрой к.т.н., ст. преп. Блинова Е.А.

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

Консультант: асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Уласевич Н.И. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Содержание

[Введение 5](#_Toc167101281)

[1 Анализ аналогичных решений 6](#_Toc167101282)

[1.1 Glovo 6](#_Toc167101283)

[1.2 Яндекс. Доставка 7](#_Toc167101284)

[1.3 Delivery Club 7](#_Toc167101285)

[2 Проектирование приложения 9](#_Toc167101286)

[2.1 Проектирование базы данных 9](#_Toc167101287)

[2.2 Описание информационных объектов и ограничений целостности 9](#_Toc167101288)

[2.1.1 Коллекция products 9](#_Toc167101289)

[2.1.2 Коллекция vouchers 9](#_Toc167101290)

[2.1.3 Коллекция restaurants 10](#_Toc167101291)

[2.1.4 Коллекция categories 10](#_Toc167101292)

[2.3 Проектирование мобильного приложения 10](#_Toc167101293)

[2.4 Выводы по главе «Проектирование программного продукта» 11](#_Toc167101294)

[3 Программная реализация приложений 12](#_Toc167101295)

[3.1 Технические средства для разработки 12](#_Toc167101296)

[3.2 Разработка глобальной базы данных 12](#_Toc167101297)

[3.3 Разработка мобильного приложения 12](#_Toc167101298)

[3.4 Вывод по главе «Программная реализация приложения» 14](#_Toc167101299)

[4 Тестирование приложения 15](#_Toc167101300)

[5 Руководство пользователя 18](#_Toc167101301)

[5.1 Руководство пользователя 18](#_Toc167101302)

[Заключение 24](#_Toc167101303)

[Список используемых источников 25](#_Toc167101304)

[ПРИЛОЖЕНИЕ А. Листинг удаления города у пользователя 26](#_Toc167101305)

# Введение

В последнее время наблюдается рост популярности мобильных приложений, позволяющих пользователю получить необходимые услуги прямо с телефона, имея доступ к интернету. Это стало возможным благодаря быстрому развитию науки и технологий, а также стремлению людей экономить время и получать важную информацию без необходимости покидать дом. Разработка мобильного приложения для мониторинга погоды дает ряд преимуществ: удобство доступа к актуальным погодным условиям, возможность планировать свои действия в зависимости от погоды, уменьшение рисков, связанных с непредвиденными погодными условиями. Приложение для мониторинга погоды позволяет пользователям всегда быть в курсе последних изменений погоды, повышая их удовлетворенность и лояльность к приложению.

Задачи курсового проекта:

* Изучение существующих мобильных приложений для мониторинга погоды, анализ их достоинств и недостатков для формирования требований к новому приложению;
* Проектирование приложения, позволяющее визуализировать предварительную концепцию и спланировать разработку программного продукта, а также распределить необходимые ресурсы и время на разработку;
* Реализация программного продукта, который будет представлять собой мобильное приложение для мониторинга погоды;
* Разработка руководства пользователя, чтобы помочь пользователям ознакомиться с основными функциями и принципами работы приложения. Это важный элемент разработки, поскольку некоторые пользователи предпочитают заранее изучить функционал перед его использованием.

По завершении изучения и формулирования задач курсового проекта, окончательная цель проекта заключается в предоставлении пользователям удобного и функционального инструмента для мониторинга погодных условий с помощью мобильного приложения.

# Анализ аналогичных решений

В современном мире приложения для доставки еды являются одними из самых востребованных веб-сервисов. Их основная задача – это предоставление удобного способа заказа еды из любимых ресторанов и кафе. При разработке своего приложения для доставки еды я обратил внимание на самые успешные примеры популярных приложений в этой сфере. Аналоги рассмотрены ниже.

## Glovo

Одним из значимых аналогов в сфере мобильных приложений для доставки еды является «Glovo». Это приложение предоставляет пользователю удобный способ заказывать еду и другие товары из различных ресторанов, магазинов и аптек. Особенности приложения включают автоматическое определение местоположения, широкий выбор ресторанов и магазинов, возможность отслеживания заказа в реальном времени, а также доставку не только еды, но и различных товаров, включая лекарства и бытовые предметы.

Интерфейс приложения показан на рисунке 1.1.

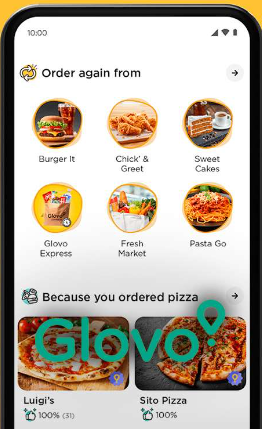


Рисунок 1.1 – Интерфейс приложения «Glovo»

Технологически приложение выделяется своей способностью к интеграции с множеством партнеров, а также удобством и простотой использования. Важной частью функционала является возможность отслеживания курьера в реальном времени, что позволяет пользователю видеть, где находится его заказ и когда он будет доставлен. Glovo также предоставляет пользователю возможность оплаты заказа различными способами, включая банковские карты и электронные кошельки. Несмотря на широкий функционал и удобство использования, пользователи могут столкнуться с недостатками, такими как наличие дополнительных сборов за доставку и ограниченная доступность в некоторых регионах.

## Яндекс. Доставка

Одним из значимых аналогов в сфере мобильных приложений для доставки еды является «Яндекс. Доставка». Это приложение предоставляет пользователю возможность заказать еду и товары из различных ресторанов, магазинов и аптек. Особенности приложения включают автоматическое определение местоположения, широкий выбор ресторанов и магазинов, возможность отслеживания заказа в реальном времени, а также быструю доставку.

Интерфейс приложения показан на рисунке 1.2.

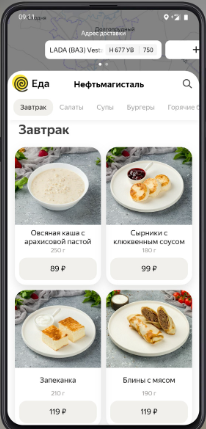


Рисунок 1.2 – Интерфейс приложения «Яндекс. Доставка»

Технологически приложение выделяется своей интеграцией с экосистемой Яндекса, что обеспечивает пользователю удобный и быстрый доступ к различным услугам. Приложение поддерживает различные способы оплаты, включая банковские карты и электронные кошельки. Пользователи могут оставлять отзывы о ресторанах и магазинах, а также ставить оценки курьерам, что повышает качество сервиса. Несмотря на широкий функционал и удобство использования, пользователи могут столкнуться с недостатками, такими как наличие дополнительных сборов за доставку и возможные задержки в часы пик.

## Delivery Club

Мобильное приложение «Delivery Club» предоставляет пользователям удобный и быстрый способ заказывать еду из множества ресторанов и кафе. Оно отличается широким выбором заведений, разнообразием кухонь и возможностью выбирать блюда по вкусу и бюджету. Пользователи могут оформить заказ через приложение и отслеживать его выполнение в реальном времени.

Интерфейс приложения показан на рисунке 1.3.

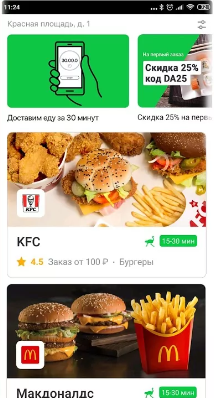


Рисунок 1.3 – Интерфейс приложения «Delivery Club»

Приложение также предлагает различные акции и скидки, что делает заказы еще более выгодными. Оно поддерживает разные способы оплаты, включая банковские карты и электронные кошельки, что обеспечивает удобство для пользователей. Delivery Club имеет функцию отзывов и рейтингов, позволяя пользователям оценивать рестораны и курьеров, что способствует улучшению качества сервиса.

Однако среди недостатков стоит отметить наличие дополнительных сборов за доставку, особенно в часы пик или при заказе из удаленных районов. Также пользователи иногда сообщают о задержках в доставке или неточностях в статусе заказа, что может снижать общее удобство использования приложения.

# Проектирование приложения

Проектирование программного средства — это процесс создания детального плана или концепции программного продукта до его реализации, включающий определение структуры и организации программы, разработку основных компонентов и модулей, их взаимосвязей и функций. На этапе проектирования выбираются архитектурные решения, которые могут быть монолитными, микросервисными, распределенными и т.д., а также подходящие паттерны проектирования для эффективной реализации задач. Проектирование интерфейсов, включающее создание пользовательских интерфейсов (UI) и программных интерфейсов (API), играет ключевую роль в обеспечении взаимодействия между компонентами системы и пользователем. Важно уделить внимание удобству и интуитивной понятности пользовательских интерфейсов, а также четкому и понятному взаимодействию между модулями через API. Определение функциональности системы включает разработку подробного описания всех функций и возможностей, которые должна предоставлять система, определение бизнес-логики, основных сценариев использования и требований к производительности.

## Проектирование базы данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Модель данных играет ключевую роль в проектировании базы данных, так как она определяет структуру хранения данных и способы их обработки.

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

Далее будет описана структура каждой коллекции. В коллекции products хранятся данные о продуктах, vouchers – данные о промокодах, restaurants – данные о ресторанах и categories – данные о категориях.

### **Коллекция products**

Коллекция products состоит из следующих полей:

* category. Хранит категорию продукта;
* description. Хранит описание продукта;
* id. Хранит уникальный идентификатор продукта;
* name. Хранит имя продукта;
* price. Хранит цену продукта;
* restaurantId. Хранит уникальный идентификатор ресторана.

### **Коллекция vouchers**

Коллекция vouchers состоит из следующих полей:

* code. Хранит код промокода;
* value. Хранит значение промокода.

### **Коллекция restaurants**

В состав коллекции restaurants входят следующие поля:

* address. Хранит адрес ресторана;
* categories. Хранит категорию ресторана;
* description. Хранит описание ресторана;
* name. Хранит название ресторана;
* openingHours. Хранит время работы;
* tags. Хранит теги.

### **Коллекция categories**

В состав коллекции categories входят следующие поля:

* id. Хранит уникальный идентификатор категории;
* imageUrl. Хранит ссылку на изображение;
* index. Хранит индекс категории;
* name. Хранит название категории.

## Проектирование мобильного приложения

Клиентская часть приложения не менее важна, чем серверная. Это то, с чем будет контактировать пользователь, то, от чего в первую очередь зависят хорошие или плохие впечатления при использовании приложения. Здесь стоит понимать, что исходя из специфики проекта, в первую очередь разрабатывается мобильное приложение и самые строгие требования со стороны пользователя будут предъявляться именно к нему.

Точно так же, как и серверная часть приложения, как и практически любой другой проект, который имеет риск сильно разрастись в будущем, лучше всего разбить логику клиентской части приложения на части таким образом, чтобы на этой основе можно было каждую часть реализовывать последовательно и независимо от других частей.

Навигация – очень важный для мобильного приложения элемент, при её проектировании необходимо поддерживать логическую связь между экранами.

При первом запуске приложения пользователь попадает на Activity авторизации, с которой он может попасть в Activity регистрации. После успешной авторизации/регистрации пользователь попадет в Activity текущей погоды, где может посмотреть текущую погоду в городе по часам.

В Activity настроек пользователь может просмотреть свои персональные данные и поменять их, изменить пароль.

Activity поиска позволяет найти и выбрать главный город, для которого будет отображаться погода.

Общая схема взаимодействия между страницами Android-клиента представлена на рисунке 2.2.



Рисунок 2.2 – Схема навигации для пользователя

Данная диаграмма показывает связь между страницами приложения.

## Выводы по главе «Проектирование программного продукта»

В данной главе было подробно рассмотрено проектирование каждого отдельного компонента приложения. Процесс проектирования начался с анализа требований и определения ключевых функций, которые должны быть реализованы в мобильном приложении. Были проведены исследования аналогичных решений и изучены лучшие практики разработки мобильных приложений.

По части мобильного приложения был определен основной функционал, который включает регистрацию и авторизацию пользователей, возможность просматривать доступные блюда и рестораны, добавлять еду в корзину и оформлять заказы. Также была разработана диаграмма вариантов использования приложения, которая наглядно демонстрирует все возможные сценарии взаимодействия пользователя с приложением. Эта диаграмма помогла уточнить и структурировать все требования к функциональности приложения.

Была также спроектирована база данных, где были определены необходимые таблицы, такие как таблицы пользователей, ресторанов, блюд и заказов. Каждая таблица была тщательно продумана и охарактеризована, включая поля каждой из них. Были установлены связи между таблицами, которые заключаются в создании первичных и внешних ключей для обеспечения целостности и связности данных. Например, была создана связь между таблицей пользователей и таблицей заказов, что позволяет точно отслеживать, какой пользователь сделал какой заказ.

# **Программная реализация приложений**

## Технические средства для разработки

Для разработки глобальной базы данных были использованы следующие средства:

* SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционных БД;
* SQLite– система управления реляционными БД;

Средства для разработки Web API:

* Visual Studio Code – среда для разработки;
* Node.js – это среда выполнения JavaScript, основанная на движке V8 от Google.
* Express – это минималистичный и гибкий фреймворк для создания веб-приложений с использованием Node.js.

Средства для разработки Android-клиента:

* Android Studio – среда для разработки Android приложений, с хорошим статическим анализатором кода;
* Flutter – это UI фреймворк, разработанный компанией Google, который позволяет создавать кроссплатформенные мобильные приложения.
* Dart – это язык программирования, разработанный компанией Google, который используется для создания мобильных, веб- и настольных приложений.

Почти все вышеперечисленные средства для разработки являются современными и популярными, поэтому выбор и пал на них.

## Разработка глобальной базы данных

Для хранения данных будет использоваться Firebase.

Основные обращения к базе данных будут осуществляется через хранимые процедуры, которые позволяют инкапсулировать сложную логику внутри процедуры и предоставить удобный интерфейс для взаимодействия с БД

## Разработка мобильного приложения

Мобильное приложение построено с использованием нескольких папок, каждая из которых выполняет определенную функцию и обеспечивает структурированное и логичное разделение кода.

* blocs включает классы, которые отвечают за управление состоянием приложения. Здесь находятся блоки управления состоянием, которые обеспечивают логику работы с данными и взаимодействие с интерфейсом пользователя.
* config содержит конфигурационные файлы и настройки, используемые во всем приложении. Это могут быть настройки API, ключи доступа и другие параметры, необходимые для работы приложения.
* models включает классы, которые описывают данные и бизнес-логику приложения. Здесь находятся классы моделей данных, отображающие структуру данных приложения и логика обработки этих данных.
* repositories содержит компоненты, которые обеспечивают работу с сервером и глобальной базой данных. Реализованы возможности отправки HTTP-запросов и обработки полученных ответов. Например, асинхронная функция placeOrder выполняет запрос к серверу для создания нового заказа и возвращает объект, содержащий информацию о выполнении функции.
* screens отвечает за взаимодействие с пользовательским интерфейсом и состоит из классов, которые определяют страницы приложения и их функциональность. Это могут быть страницы авторизации, регистрации, меню ресторанов и оформления заказа.
* widgets представляет отдельные элементы интерфейса, такие как кнопки, текстовые поля и другие виджеты, используемые на различных страницах. Эти виджеты обеспечивают консистентность и удобство при создании интерфейса пользователя.

Реализация асихронной функции, которая выполняет запрос к серверу для создания нового заказа, представлена в листинге 3.5.

Future<void> placeOrder(int userId, int restaurantId, List<int> productIds) async {

final response = await http.post(

Uri.parse('${Config.url}/placeOrder'),

headers: <String, String>{'Content-Type': 'application/json'},

body: json.encode({'userId': userId, 'restaurantId': restaurantId, 'productIds': productIds}),

);

if (response.statusCode == 200) {

print('Order placed successfully');

var orderDetails = json.decode(response.body);

setState(() {

\_orderDetails = orderDetails;

});

} else {

throw Exception('Failed to place order');

}

Листинг 3.5 – Создание нового заказа

Пример реализации функции удаления заказа у пользователя представлен в листинге, который находится в приложении А.

Функция отправляет HTTP DELETE-запрос к серверу, передавая идентификаторы пользователя и заказа. Если сервер успешно выполняет удаление (ответ с кодом состояния 200), функция обновляет список заказов пользователя, уменьшая счетчик заказов и показывая уведомление об успешном удалении заказа. В случае ошибки функция показывает уведомление с сообщением об ошибке.

## Вывод по главе «Программная реализация приложения»

В этой главе подробно рассмотрен процесс разработки всех компонентов приложения. Сначала было проведено исследование и отбор необходимых технологий для реализации курсового проекта. После выбора технологий была создана структура базы данных, определив ключевые таблицы и связи между ними для обеспечения целостности и эффективности работы с данными. Были разработаны запросы для взаимодействия с базой данных, включая получение, добавление, обновление и удаление данных.

Затем было разработано серверное приложение, которое играет ключевую роль в обработке данных и взаимодействии между клиентской частью и базой данных. В этом разделе была представлена графическая структура серверного приложения, описаны используемые классы и методы, а также их функциональность. Описание включало логику обработки запросов, маршрутизацию, аутентификацию пользователей и другие важные аспекты работы сервера.

После завершения разработки серверной части проекта началась работа над мобильным приложением. В этом разделе также была представлена графическая структура приложения, рассмотрены все классы и методы, их роль и взаимодействие. Особое внимание уделено интерфейсу пользователя, взаимодействию с сервером, обработке данных и обеспечению удобства использования приложения. Были детально рассмотрены компоненты приложения, такие как модели данных, сервисы, утилиты и представления, что позволило получить полное представление о работе мобильного приложения.

После этого началась разработка веб-приложения. Здесь была представлена его графическая структура и основные файлы. Обсуждение включало описание архитектуры веб-приложения, взаимодействие с сервером и базой данных, а также реализацию пользовательского интерфейса. Были рассмотрены классы и методы, используемые для обеспечения функциональности веб-приложения, включая обработку пользовательских запросов, отображение данных и другие важные аспекты.

Таким образом, в этой главе детально описан весь процесс разработки приложения, начиная с выбора технологий и заканчивая реализацией всех компонентов – от базы данных до мобильного и веб-приложения. Это позволило получить полное представление о создании комплексного программного обеспечения, включающего все необходимые этапы и технологии для успешной реализации проекта.

# Тестирование приложения

Тестирование мобильных приложений важно и необходимо для обеспечения их качества и производительности. Оно помогает уменьшить риск сбоев приложения, гарантирует отсутствие ошибок и улучшает восприятие пользователей, что в свою очередь способствует увеличению количества загрузок и успешному запуску приложения на рынке.

Для начала протестируем экран загрузки. После запуска приложения, пользователь видит заставку, которая затем плавно переходит на следующую страницу. Это обеспечивает плавный и приятный пользовательский опыт. На рисунке 4.1 показано тестирование экрана загрузки, чтобы убедиться, что он отображается корректно и переходит на следующую страницу без задержек или сбоев.

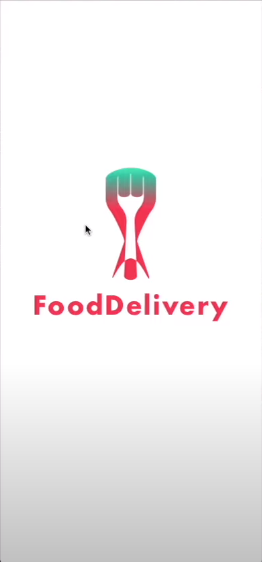


Рисунок 4.1 – Тестирование экрана загрузки

Далее протестируем страницу с картой для выбора местоположения. Пользователь может выбрать свое текущее местоположение или указать его на карте. При выборе и сохранении местоположения данные должны корректно сохраняться в системе. Тестирование этой функции продемонстрировано на рисунке 4.2, где проверяется корректность отображения карты и сохранения выбранного местоположения.

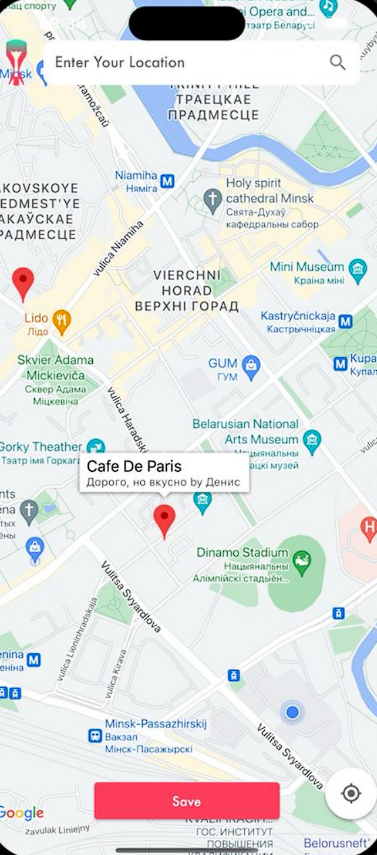


Рисунок 4.2 – Тестирование выбора местоположения

Следующий этап – тестирование домашнего экрана. На этой странице пользователь может выбирать различные виды еды, а также просматривать лучшие рейтинговые рестораны и акции. Особое внимание уделяется функции поиска ресторана. При вводе названия ресторана в строку поиска и нажатии на кнопку поиска должно отображаться корректное количество результатов. Тестирование показано на рисунке 4.3.

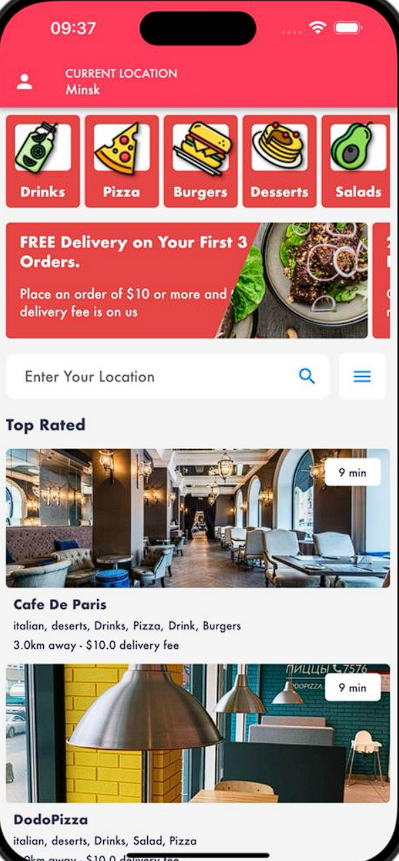


Рисунок 4.3 – Тестирование поиска ресторана

Теперь протестируем функцию добавления еды в корзину из страницы ресторана. Пользователь выбирает ресторан, просматривает меню и добавляет выбранные блюда в корзину. При добавлении еды должны корректно отображаться название блюда, его цена и общее количество. На рисунке 4.4 продемонстрировано тестирование добавления еды в корзину, включая проверку правильности отображения выбранных позиций и общей стоимости.

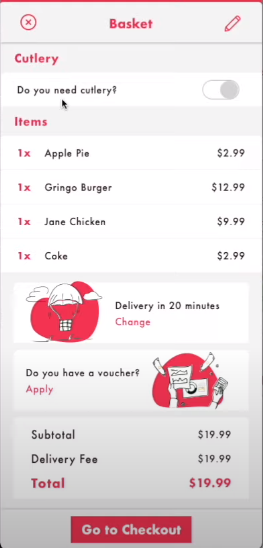


Рисунок 4.4 – Тестирование добавления еды в корзину

Наконец, протестируем страницу оформления заказа. Пользователь может просмотреть выбранные блюда, изменить время доставки, применить купон и выбрать способ оплаты. На рисунке 4.5 показано тестирование функции изменения времени доставки и применения купона. Это тестирование включает проверку корректности расчета общей стоимости заказа с учетом скидок и корректного отображения информации о доставке.

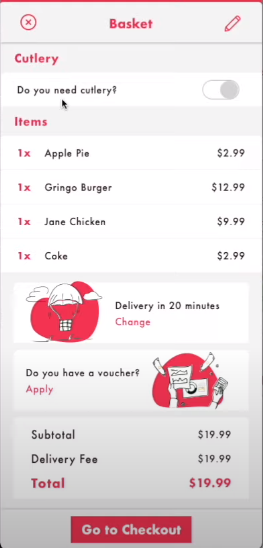


Рисунок 4.4 – Тестирование оформления заказа

Тестирование приложения является неотъемлемой частью процесса разработки, позволяющей обеспечить высокое качество продукта и удовлетворенность пользователей. В рамках данного проекта, благодаря тщательной валидации и обработке ошибок, было создано надежное и удобное в использовании мобильное приложение для доставки еды, которое успешно выполняет свои функции и отвечает потребностям целевой аудитории.

# Руководство пользователя

В данном разделе будет описано руководство по использованию для пользователя.

## Руководство пользователя

При первом запуске приложения пользователь видит экран загрузки. Это заставка, которая отображается в течение нескольких секунд, прежде чем пользователь попадет на следующую страницу. Это важно для загрузки всех необходимых данных и создания приятного первого впечатления. Представлено на рисунке 5.1.

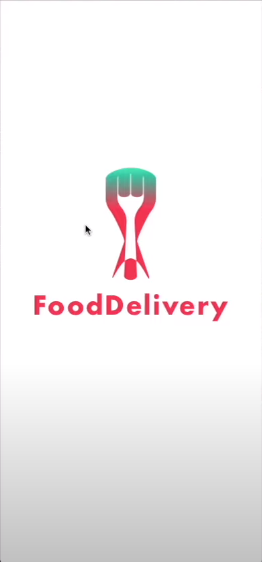


Рисунок 5.1 – Страница загрузки

После экрана загрузки пользователь попадает на страницу с картой для выбора местоположения. Здесь можно выбрать текущее местоположение или указать его вручную на карте. После выбора местоположения пользователь нажимает кнопку сохранения, чтобы зафиксировать выбор. Это местоположение будет использоваться для поиска ближайших ресторанов и расчета времени доставки. Представлено на рисунке 5.2.

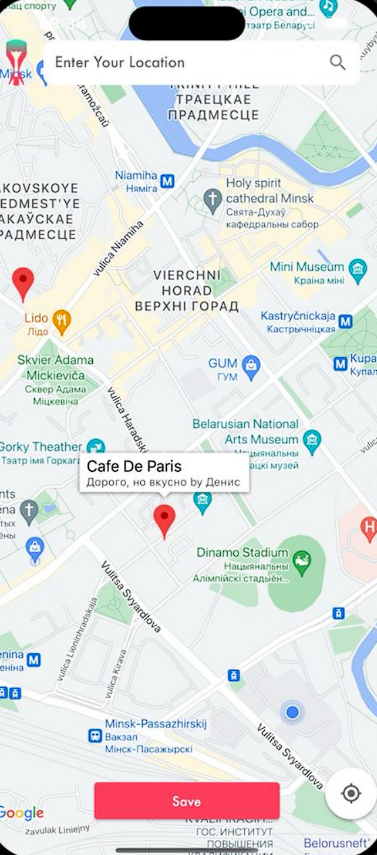


Рисунок 5.2 – Страница с картой для выбора местоположения

После сохранения местоположения пользователь переходит на главную страницу. На этой странице доступны следующие функции:

* Выбор различных видов еды: Пользователь может выбирать из множества категорий еды, таких как пицца, суши, бургеры и т.д.
* Отображение лучших рейтинговых ресторанов и акций: На главной странице также отображаются рестораны с высоким рейтингом и текущие акции.
* Поиск ресторана: В верхней части экрана находится строка поиска, где можно ввести название ресторана и найти его среди доступных.

Представлено на рисунке 5.3.

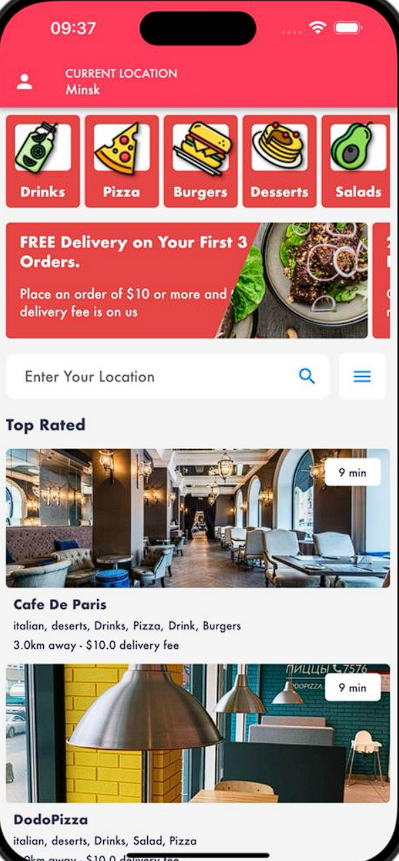


Рисунок 5.3 – Главная страница

При нажатии на определенный тип еды, например, пицца, открывается страница с ресторанами, специализирующимися на этом типе еды. Здесь пользователь видит список всех ресторанов, предлагающих выбранный тип еды. На этой странице можно фильтровать рестораны по цене, выбирая от дешевых к дорогим, а также просматривать специальные предложения и скидки, которые могут быть доступны в некоторых ресторанах. Представлено на рисунке 5.4.

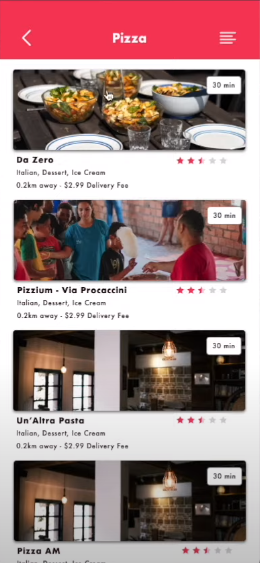


Рисунок 5.4 – Страница с ресторанами

При выборе ресторана из списка пользователь переходит на страницу этого ресторана. Здесь отображается название выбранного ресторана, краткая информация о нем, включая рейтинг, время работы и контактные данные. Также представлен меню ресторана, где указаны все доступные блюда и их цены. Пользователь может добавлять блюда в корзину, нажав на соответствующие иконки, и перейти в корзину, используя специальную кнопку. Представлено на рисунке 5.5.

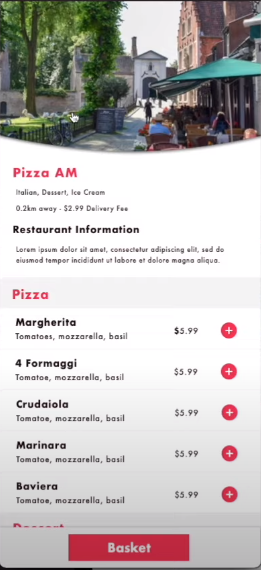


Рисунок 5.5 – Страница ресторана

На странице корзины пользователь видит список всех добавленных блюд с указанием их количества и стоимости. Здесь можно выбрать или изменить время доставки. Если у пользователя есть купон, он может применить его для получения скидки. Отображается общая стоимость заказа с учетом всех примененных скидок. Пользователь нажимает кнопку оформления заказа, чтобы перейти к следующему этапу. Представлено на рисунке 5.6.

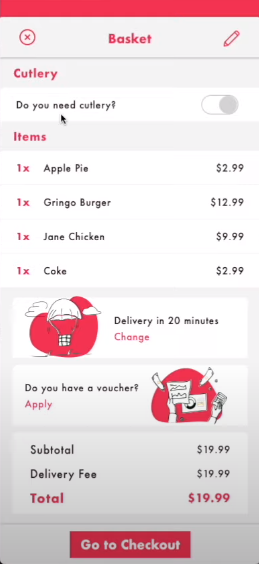


Рисунок 5.6 – Страница корзины

При нажатии на кнопку изменения времени доставки открывается страница выбора времени или дня доставки. Пользователь может выбрать удобное для него время из предложенных вариантов. Продемонстрировано на рисунке 5.7.

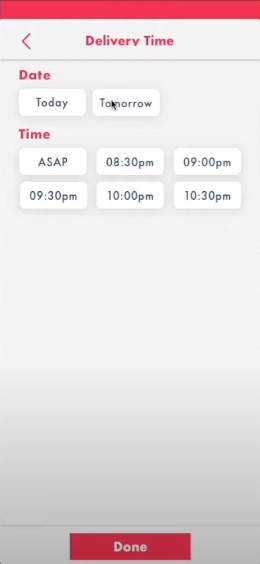


Рисунок 5.7 – Страница выбора времени доставки

При нажатии на кнопку купона происходит переход на страницу, где пользователь может просмотреть список всех ранее использованных купонов. Также доступно поле для ввода нового купона, чтобы применить его к текущему заказу. Представлено на рисунке 5.8.

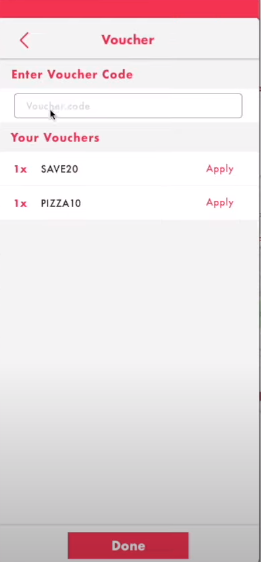


Рисунок 5.8 – Страница купонов

При нажатии на кнопку оформления заказа пользователь попадает на страницу с выбором адреса и способа оплаты. Здесь можно выбрать ранее сохраненный адрес или ввести новый адрес для доставки. Пользователь выбирает подходящий способ оплаты, например, кредитная карта, электронные кошельки или наличные при доставке. Отображается подтверждение общей стоимости заказа с учетом доставки и примененных купонов. Пользователь завершает процесс, нажав на кнопку оплаты заказа. Представлено на рисунке 5.9.

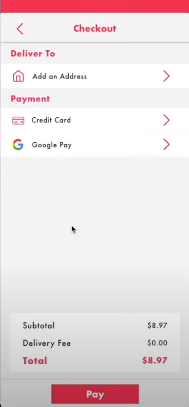


Рисунок 5.9 – Страница оформления заказа

После успешной оплаты пользователь переходит на страницу с информацией о заказах. Здесь представлен список всех ранее оплаченных заказов с подробной информацией о каждом из них. Также отображается информация о текущем заказе, который находится в процессе доставки, с указанием времени и статуса доставки. Представлено на рисунке 5.10.

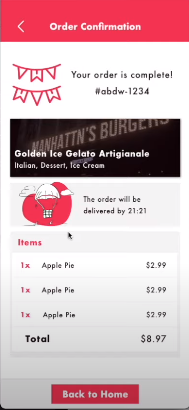


Рисунок 5.10 – Страница с информацией о заказах

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение, которое предоставляет собой приложение для доставки еды.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* Регистрация пользователей
* Авторизация пользователей
* Просмотр различных видов еды и лучших рейтинговых ресторанов
* Отображение акций
* Поиск ресторанов или выбор из списка
* Просмотр списка ресторанов по типу еды (например, пицца)
* Фильтрация ресторанов по цене
* Просмотр специальных предложений ресторанов
* Просмотр детальной информации о ресторане (название, информация, меню, цены)
* Добавление еды в корзину
* Переход в корзину с информацией о выбранной еде, времени доставки, наличии купона и общей стоимости заказа
* Изменение времени доставки
* Ввод и использование купонов
* Оформление заказа
* Выбор адреса доставки
* Выбор способа оплаты
* Подтверждение общей стоимости заказа
* Оплата заказа
* Просмотр информации о оплаченных заказах и текущих доставках

Разработанное программное средство реагирует на ошибочный ввод данных, выводя соответствующее сообщение об ошибке. Программа включает реализацию корзины и фильтрации продуктов, а также заказ доставки на указанный адрес с использованием геолокации.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанное приложение для доставки еды работает верно и требования технического задания выполнены в полном объеме.

# Список используемых источников

1. Flutter Community [Электронный ресурс]. – Режим доступа: https://fluttercommunity.dev/ – Дата доступа: 22.04.2024.
2. Dart Language Tour [Электронный ресурс]. – Режим доступа: https://dart.dev/guides/language/language-tour – Дата доступа: 10.03.2024.
3. Firebase Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://firebase.google.com/docs – Дата доступа: 15.04.2024.
4. Pro Dart and Flutter [Электронный ресурс]. – Режим доступа: https://pro-dart-flutter.dev/ – Дата доступа: 11.04.2024.
5. Google Developer Tutorials [Электронный ресурс]. – Режим доступа: https://developers.google.com/ – Дата доступа: 18.04.2024.

# ПРИЛОЖЕНИЕ А. Листинг удаления города у пользователя

Future<void> \_deleteOrder(int userId, int orderId) async {

final response = await http.delete(

Uri.parse('${Config.url}/deleteOrder'),

headers: <String, String>{'Content-Type': 'application/json'},

body: json.encode({'userId': userId, 'orderId': orderId}),

);

if (response.statusCode == 200) {

final newOrders = await getUserOrders(userId);

setState(() {

orderCount -= 1;

orders = newOrders;

});

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text('Success'),

content: Text('Order was removed'),

actions: <Widget>[

TextButton(

child: Text('OK'),

onPressed: () => Navigator.of(context).pop(),

),

],

);

},

);

} else {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text('Error'),

content: Text('Failed to remove the order.'),

actions: <Widget>[

TextButton(

child: Text('OK'),

onPressed: () => Navigator.of(context).pop(),

),

],

);

},

);

}