# ВВЕДЕНИЕ

В современном мире технологический прогресс непрерывно изменяет наши повседневные привычки и потребности. Одним из самых динамично развивающихся секторов является сфера услуг доставки. В условиях активного обмена товарами и услугами как международного, так и локального характера, важным является развитие эффективных и удобных инструментов управления курьерской доставкой.

В этом контексте мобильные приложения становятся одним из ключевых элементов современного бизнеса в сфере доставки. Их функциональность позволяет как заказчикам, так и поставщикам услуг управлять и отслеживать доставку, обеспечивая высокий уровень сервиса и комфорта.

Данный дипломный проект посвящен разработке мобильного приложения, направленного на управление процесса доставки и управления курьерскими услугами. Целью данного проекта является создание удобного решения, способного удовлетворить потребности как заказчиков, так и поставщиков услуг доставки, обеспечивая максимальную эффективность и удобство взаимодействия.

В работе будут рассмотрены основные аспекты разработки мобильного приложения, начиная от анализа потребностей пользователей и функциональных требований, заканчивая проектированием интерфейса и реализацией программного продукта. Также будет уделено внимание вопросам безопасности, масштабируемости и интеграции с другими информационными системами.

Результаты данного проекта могут быть востребованы как в коммерческой сфере, так и в сфере общественного обслуживания, способствуя развитию современных технологий и повышению качества услуг доставки в целом.

Кроме того, данный проект предоставляет возможность исследования и применения современных методов разработки программного обеспечения, включая принципы гибкой разработки и методологии DevOps. Подходы к тестированию и оптимизации производительности также будут уделены должное внимание с целью обеспечения стабильной и высокопроизводительной работы приложения в условиях реального использования.

В контексте расширения возможностей мобильных устройств и развития интернет-инфраструктуры, данное приложение имеет потенциал стать неотъемлемым инструментом для широкого круга пользователей, сокращая время и усилия, затрачиваемые на организацию и контроль процесса доставки.

Надеемся, что результаты данного дипломного проекта не только помогут в достижении его основных целей, но и привлекут внимание как профессионального сообщества, так и широкой аудитории пользователей, заинтересованных в современных решениях в сфере управления курьерской доставкой

1. **СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ**

**1.1 Анализ предментной области**

Рынок доставки продовольственных товаров демонстрирует стремительный рост в последние годы, в особенности в контексте изменений потребительского поведения и технологического прогресса.

Рост спроса на услуги доставки продовольственных товаров становится заметным явлением в современном мире, особенно в густонаселенных городах и урбанизированных областях. Этот рост спроса связан с изменением образа жизни современного общества. Городские жители все более загружены работой, учебой и другими обязанностями, что уменьшает время на покупки и приготовление еды. Также популярность онлайн-шопинга и удобство использования мобильных приложений для заказа продуктов способствуют росту спроса на услуги доставки продуктов питания. Этот тренд также поддерживается молодыми семьями и активными рабочими людьми, которые предпочитают экономить время и силы, обращаясь к услугам доставки продуктов.

Рынок доставки продовольственных товаров представляет собой конкурентную среду, где действуют различные участники, такие как крупные сети супермаркетов, специализированные службы доставки, стартапы и небольшие местные бизнесы. Каждый из них стремится привлечь и удержать клиентов, предлагая различные услуги, ассортимент продуктов, ценовую политику и уровень сервиса. Крупные сети супермаркетов обычно имеют значительные ресурсы и брендовую известность, что делает их сильными участниками на рынке. Они могут предложить широкий ассортимент продуктов, конкурентные цены и интеграцию услуг доставки с уже существующей инфраструктурой магазинов.

Специализированные службы доставки, такие как такси-доставки еды или сервисы подписки на готовые блюда, обычно специализируются на определенных типах продуктов или услугах. Они могут предложить более гибкие условия доставки и персонализированный сервис, привлекая клиентов своей уникальной концепцией.

Стартапы и небольшие местные бизнесы часто ориентируются на узкую целевую аудиторию или на определенный сегмент рынка. Они могут выделяться инновационными подходами к услугам доставки, использованием местных продуктов или экологически устойчивыми практиками, чтобы привлечь внимание клиентов.

Конкурентный рынок доставки продовольственных товаров также характеризуется постоянным появлением новых участников и изменением стратегий существующих игроков. Технологические инновации, изменения в потребительских предпочтениях и внешние факторы, такие как экономические кризисы, могут значительно повлиять на конкуренцию на рынке. В такой среде важно проводить регулярный анализ конкурентов и адаптировать свои стратегии под изменяющиеся условия, чтобы оставаться конкурентоспособным.

Логистика и инфраструктура играют важную роль в сфере доставки продовольственных товаров. Эффективное управление маршрутами доставки, оптимизация складских операций и выбор подходящих транспортных средств являются ключевыми аспектами обеспечения быстрой и надежной доставки. Кроме того, современные информационные системы помогают отслеживать грузы, координировать доставки и повышать прозрачность процессов. Анализ инфраструктуры города также важен для определения оптимальных маршрутов и улучшения эффективности доставки. Все это содействует повышению конкурентоспособности компаний на рынке доставки продовольственных товаров.

**1.2. Постановка задачи**

**Цель проекта:** Создание мобильного приложения, которое обеспечит удобную и эффективную доставку продовольственных товаров для пользователей, а также инструменты управления заказами для администраторов и курьеров.

**Функциональные требования для пользователя:**

1. **Регистрация и аутентификация:** Пользователи могут создать учетную запись, войти в систему с помощью электронной почты или номера телефона.
2. **Просмотр ассортимента:** Пользователи могут просматривать доступные продукты с описанием, изображениями и ценами.
3. **Оформление заказа:** Пользователи могут добавлять продукты в корзину, выбирать параметры доставки (время и адрес), редактировать и подтверждать заказ.
4. **Отслеживание заказа:** Пользователи могут отслеживать статус своего заказа (принят, в пути, доставлен).
5. **Корзина: пользователь может добавлять свои заказы в корзину, там выбирать нужное количество порций и удалять то, что он передумал покупать.**
6. **Изменение данных пользователя:** важно обеспечить данную функцию для удобного пользования при возможном изменении адреса и т.д.

**Функциональные требования для администратора:**

1. **Управление ассортиментом:** Администраторы могут добавлять, редактировать и удалять продукты, устанавливать цены и параметры.
2. **Управление заказами:** Администраторы могут просматривать и управлять заказами, принимать заказы и отслеживать статусы доставок.
3. **Управление пользователями:** Администраторы могут просматривать данные пользователей, управлять учетными записями, решать вопросы поддержки.

**1.3. Анализ существующих решений**

Yandex Food представляет собой популярный сервис доставки еды, предоставляемый компанией Яндекс. Он предлагает широкий выбор ресторанов и кафе для заказа блюд различных кухонь прямо через мобильное приложение или веб-сайт. Этот сервис отличается следующими ключевыми особенностями:

**Широкий выбор ресторанов:** Yandex Food предлагает клиентам доступ к многочисленным ресторанам и кафе, представляющим различные кухни и культуры. Это обеспечивает разнообразие в выборе блюд и удовлетворяет различные вкусы клиентов.

**Удобное приложение:** Мобильное приложение Yandex Food имеет простой и интуитивно понятный интерфейс, что делает процесс заказа еды быстрым и удобным для пользователей.

**Система оценок и отзывов:** Пользователи имеют возможность оценивать рестораны и блюда, а также оставлять отзывы о своем опыте. Это помогает другим клиентам принимать информированные решения при выборе ресторана и блюд.

Shef — это уникальный сервис доставки еды, который соединяет клиентов с местными поварами, готовящими домашние блюда. В отличие от многих других сервисов доставки еды, Shef делает акцент на свежеприготовленных, аутентичных блюдах от местных шефов. Вот подробное описание работы Shef и его отличительных особенностей:

**Местные и аутентичные блюда**: Shef позволяет талантливым домашним поварам делиться своими кулинарными навыками с сообществом. Эти повара, называемые «шефами», готовят еду небольшими партиями, используя свежие и качественные ингредиенты. Такой подход гарантирует, что пища не производится массово, не транспортируется на большие расстояния и не замораживается, что позволяет сохранять высокое качество и вкус еды​.

**Разнообразие и настройка под клиента**: Платформа предлагает широкий ассортимент кухонь, отражающий культурное разнообразие поваров. Клиенты могут выбирать из различных блюд, включая варианты для различных диетических предпочтений, таких как веганские, безглютеновые, низкоуглеводные и другие. Такое обширное меню позволяет персонализировать план питания, учитывая конкретные потребности и кулинарные предпочтения​​.

**Пользовательский опыт**: Веб-сайт и приложение Shef разработаны для обеспечения удобного пользовательского опыта. Клиенты могут просматривать меню, знакомиться с поварами и легко размещать заказы. Сервис также акцентирует внимание на прозрачности, предоставляя подробные описания ингредиентов и методов приготовления для каждого блюда​​.

**Поддержка местных поваров**: Одним из ключевых преимуществ Shef является поддержка местных предпринимателей. Заказывая через Shef, клиенты напрямую поддерживают домашних поваров и владельцев малого бизнеса, помогая им зарабатывать на жизнь через их кулинарное увлечение. Эта модель приносит пользу не только поварам, но и способствует укреплению сообщества​.

**Удобство и качество**: Блюда, заказанные через Shef, доставляются свежими и готовыми к употреблению, что обеспечивает удобную альтернативу домашней готовке без компромиссов в качестве. Сервис особенно привлекателен для занятых людей и семей, которые хотят наслаждаться домашними блюдами без необходимости тратить время и усилия на их приготовление​.

# Проектирование СТРУКТУРЫ ПРИЛОЖЕНИЯ

## 2.1 Выбор средств реализации

Перед началом разработки мобильного приложения для доставки еды было проведено тщательное исследование и анализ различных технологий, чтобы выбрать наиболее подходящие инструменты и платформы. В итоге было принято решение использовать Kotlin в среде разработки Android Studio с интеграцией Firebase. Этот выбор основан на следующих обоснованных причинах.

**Kotlin:**

**Современный и безопасный язык программирования**: Kotlin, разработанный компанией JetBrains, был выбран в качестве основного языка программирования для разработки Android-приложения. Это современный язык, поддерживающий как объектно-ориентированное, так и функциональное программирование, что позволяет писать более выразительный и лаконичный код. Kotlin обеспечивает повышенную безопасность благодаря встроенной системе обработки null-значений, что значительно снижает вероятность возникновения ошибок, связанных с null-pointer exceptions (NPE). Эти свойства делают Kotlin предпочтительным выбором для создания надежного и поддерживаемого кода.

**Полная совместимость с Java**: Одним из ключевых факторов выбора Kotlin стала его полная совместимость с Java. Это позволяет использовать существующие библиотеки и фреймворки на Java без необходимости полного переписывания кода, что ускоряет процесс разработки и облегчает интеграцию с уже существующими системами. Кроме того, разработчики, знакомые с Java, могут быстро освоить Kotlin благодаря схожести синтаксиса и концепций.

**Поддержка Google и сообщество разработчиков**: Kotlin является официально поддерживаемым языком программирования для разработки приложений под Android, что гарантирует его долгосрочную поддержку и регулярные обновления от Google. Широкое сообщество разработчиков также предоставляет доступ к обширной документации, форумам и учебным материалам, что упрощает процесс обучения и решения возникающих проблем.

**Android Studio:**

**Мощная интегрированная среда разработки (IDE)**: Android Studio, официальная IDE для разработки Android-приложений, была выбрана из-за своей высокой функциональности и интеграции с инструментами разработки. Среда предлагает все необходимые инструменты для написания, тестирования и отладки кода, что позволяет ускорить цикл разработки и повысить качество конечного продукта. Инструменты, такие как визуальный редактор макетов, анализатор APK и менеджер виртуальных устройств, значительно облегчают процесс разработки и тестирования.

**Интеграция с Kotlin**: Android Studio предоставляет полную поддержку Kotlin, включая автодополнение, рефакторинг кода и инструменты анализа. Это обеспечивает более продуктивную и удобную работу с кодом, что особенно важно при разработке сложных приложений.

**Firebase:**

**Мощная облачная платформа**: Firebase была выбрана в качестве бэкенд-платформы из-за своего широкого набора функций и инструментов, которые идеально подходят для разработки и поддержки мобильных приложений. Firebase предлагает такие сервисы, как аутентификация пользователей, база данных в реальном времени, хранение файлов и аналитика, что позволяет сосредоточиться на разработке функционала приложения, не беспокоясь о инфраструктуре.

**Аутентификация и безопасность**: Firebase Authentication позволяет легко интегрировать различные методы аутентификации, включая email, Google и социальные сети. Это обеспечивает высокий уровень безопасности и удобства для пользователей, что особенно важно для приложений, работающих с персональными данными.

**База данных в реальном времени и облачные функции**: Firebase Realtime Database и Firestore предоставляют возможности для хранения и синхронизации данных в реальном времени, что критически важно для приложения доставки еды, где необходимо быстро обновлять статусы заказов и данные пользователей. Firebase Cloud Functions позволяет выполнять серверные операции, такие как обработка платежей и отправка уведомлений, без необходимости управления собственным сервером, что упрощает масштабирование и администрирование приложения.

**Аналитика и мониторинг**: Firebase Analytics предоставляет детальную информацию о поведении пользователей, что помогает оптимизировать приложение и улучшать пользовательский опыт. Firebase Crashlytics помогает отслеживать и исправлять ошибки, повышая стабильность приложения.

Выбор Kotlin в качестве языка программирования, Android Studio как среды разработки и Firebase как облачной платформы был обусловлен их современными возможностями, высокой функциональностью и поддержкой. Эти инструменты обеспечивают надежную, безопасную и эффективную разработку мобильного приложения, что особенно важно для создания высококачественного сервиса доставки еды.

## 2.2 Анализ и проектирование архитектуры приложения

Для обеспечения модульности, повторного использования компонентов и легкости тестирования, было решено использовать архитектурный стиль MVVM (Model-View-ViewModel) при разработке мобильного приложения для доставки еды.

#### MVVM (Model-View-ViewModel)

**Model**:

* **Описание**: Слой данных, который отвечает за управление бизнес-логикой и взаимодействие с источниками данных, такими как базы данных и сетевые API. В нашем случае, это будут Firebase Realtime Database и Firestore.
* **Реализация**:
  + **Firebase Realtime Database**: Используется для хранения данных, которые требуют мгновенного обновления, например, статус заказов и текущие позиции меню.
  + **Firestore**: Используется для хранения структурированных данных и выполнения сложных запросов. Это может включать профили пользователей, историю заказов и данные об инвентаре.
  + **Репозитории**: Абстрактный слой, который инкапсулирует логику доступа к данным. Репозитории взаимодействуют с Firebase, предоставляя методы для получения и обновления данных.

**View**:

* **Описание**: UI-компоненты, которые отвечают за отображение данных пользователю и обработку пользовательского интерфейса. В Android это Activity и Fragment.
* **Реализация**:
  + **XML файлы для верстки интерфейса**: Определяют макеты экранов приложения, такие как экраны регистрации, меню и статус заказа.
  + **Activity и Fragment**: Компоненты, которые управляют жизненным циклом и взаимодействуют с ViewModel для получения данных и обновления интерфейса.

**ViewModel**:

* **Описание**: Управляет логикой интерфейса пользователя и взаимодействует с Model для получения данных. ViewModel также сохраняет состояние UI и переживает изменение конфигураций, таких как поворот экрана.
* **Реализация**:
  + **LiveData**: Используется для хранения и управления данными, которые могут изменяться. Это позволяет View подписываться на изменения данных и автоматически обновляться.
  + **Классы ViewModel**: Реализуют логику для получения данных из репозиториев, обработки пользовательских событий и управления состоянием UI.

### 4. Компоненты архитектуры

#### Клиентская часть

**View**:

* **XML файлы для верстки интерфейса**:
  + Определяют макеты для различных экранов приложения, таких как экран регистрации, экран выбора меню, экран оплаты и экран отслеживания заказа.
  + Пример: файл activity\_main.xml может содержать элементы интерфейса для отображения списка блюд, кнопок добавления в корзину и кнопки оформления заказа.
* **Activity и Fragment**:
  + **Activity**: Основной компонент UI, который управляет одним экраном приложения. Пример: MainActivity, которая содержит фрагменты для отображения меню и корзины.
  + **Fragment**: Компонент, который может быть использован в нескольких активити, обеспечивая переиспользование UI-компонентов. Пример: MenuFragment, который отображает список доступных блюд.

**ViewModel**:

* **LiveData**: Компонент, который обеспечивает реактивное программирование в Android. Используется для отслеживания изменений данных и уведомления View об этих изменениях.
  + Пример: MutableLiveData<List<MenuItem>> в MenuViewModel для хранения списка блюд.
* **Классы ViewModel**: Управляют логикой приложения, взаимодействуют с репозиториями и обеспечивают данные для View.
  + Пример: MenuViewModel, который содержит методы для получения списка блюд из репозитория и обработки добавления блюд в корзину.

**Model**:

* **Репозитории**: Абстрактный слой, который взаимодействует с Firebase и предоставляет методы для получения и обновления данных.
  + Пример: MenuRepository, который содержит методы для получения списка блюд из Firestore и обновления данных о заказах в Realtime Database.
* **Firebase**:
  + **Realtime Database**: Обеспечивает хранение и синхронизацию данных в реальном времени. Используется для хранения данных, которые требуют мгновенного обновления, таких как статусы заказов и актуальные позиции меню.
  + **Firestore**: Обеспечивает гибкое и масштабируемое хранение данных с возможностью выполнения сложных запросов. Используется для хранения структурированных данных, таких как профили пользователей и история заказов.

#### Бэкенд

**Firebase Authentication**:

* Обеспечивает безопасную аутентификацию пользователей через различные методы, включая email, Google и социальные сети. Это позволяет легко интегрировать безопасные методы входа в приложение.

**Firebase Realtime Database и Firestore**:

* **Realtime Database**: Используется для хранения данных, которые требуют мгновенного обновления и синхронизации в реальном времени. Пример использования: отслеживание текущих заказов и обновление их статусов в реальном времени.
* **Firestore**: Используется для хранения структурированных данных и выполнения сложных запросов. Пример использования: хранение данных о меню, профилей пользователей и истории заказов.

**Firebase Cloud Functions**:

* Позволяет выполнять серверные операции, такие как обработка платежей, отправка уведомлений и выполнение сложной логики. Это уменьшает нагрузку на клиентское приложение и улучшает производительность.
* Пример использования: автоматическая отправка уведомлений пользователям при изменении статуса заказа или обработка транзакций при оплате заказа.

**Интеграционные компоненты**

**API**:

* Взаимодействие с внешними сервисами, такими как платежные системы (например, Stripe, PayPal) и службы доставки. API обеспечивают безопасную и надежную интеграцию с внешними системами.
* Пример использования: вызов API платежного провайдера для обработки транзакций.

**Notification System**:

* Использование Firebase Cloud Messaging для отправки пуш-уведомлений пользователям. Это позволяет информировать пользователей о статусе их заказов, специальных предложениях и других важных событиях.
* Пример использования: отправка уведомления пользователю о том, что его заказ доставлен.

Использование архитектурного стиля MVVM в сочетании с Kotlin, Android Studio и Firebase обеспечивает высокую модульность, повторное использование компонентов и легкость тестирования. Клиентская часть приложения построена на основе четкого разделения между View, ViewModel и Model, что упрощает поддержку и расширение функционала. Firebase предоставляет мощные инструменты для управления аутентификацией, хранением данных и выполнения серверных функций, что делает разработку и масштабирование приложения более эффективными.

## 2.3 Проектирование базы данных

# РАЗРАБОТКА АЛГОРИТМОВ

# Тестирование системы

# Заключение

Начало формы

# Список исользованных источников

1. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.

2. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.

3. ГОСТ 19.102-77. ЕСПД. Стадии разработки.

4. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.

5. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.

6. ГОСТ 19.402-78. ЕСПД. Описание программы.

7. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

8. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

9. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.

10. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.