**Министерство образования Республики Беларусь**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кафедра технологий программирования**

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ИНТЕРНЕТ-МАГАЗИНА КОНДИТЕРСКИХ ИЗДЕЛИЙ**

Отчёт по преддипломной практике

|  |  |
| --- | --- |
| Сорока Александра Владимировича  студента 4 курса,  специальность «прикладная  информатика»  Научный руководитель:  доцент кафедры технологий  программирования, кандидат  технических наук  Е.А. Левчук |  |

Минск, 2024

РЕФЕРАТ

Отчёт по преддипломной практике, 37 c., 30 рис.

**Ключевые слова:** ANDROID, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ИНТЕРНЕТ-МАГАЗИН, RETROFIT, КЛИЕНТ, СЕРВЕР.

**Объект исследования –** мобильное приложение для интернет-магазина.

**Цели работы –** рассмотреть технологии разработки клиент-серверных приложений, а также спроектировать приложение для интернет-магазина кондитерских изделий.

**Методы исследования –** а) теоретические: изучение литературы, посвящённой разработке мобильных приложений; б) практические: моделирование, проектирование спецификации и разработка мобильного приложения для интернет-магазина кондитерских изделий для платформы Android.

**Результатами являются –**мобильное приложение для интернет-магазина кондитерских изделий для платформы Android.

**Область применения –** для любого смартфона с операционной системой Android.

РЭФЕРАТ

Справаздача па пераддыпломнай практыцы, 37 с., 30 мал.

**Ключавыя словы**: ANDROID, МАБІЛЬНАЕ ПРЫКЛАДАННЕ, ІНТЭРНЭТ-КРАМА, RETROFIT, КЛІЕНТ, СЭРВЕР.

**Аб'ект даследавання** **–** мабільнае прыкладанне для інтэрнэт-крамы.

**Мэты працы** **–** разгледзець тэхналогіі распрацоўкі кліент-серверных прыкладанняў, а таксама спраектаваць прыкладанне для інтэрнэт-крамы кандытарскіх вырабаў.

**Метады даследавання** **–** а) тэарэтычныя: вывучэнне літаратуры, прысвечанай распрацоўцы мабільных дадаткаў; б) практычныя: мадэляванне, праектаванне спецыфікацыі і распрацоўка мабільнага прыкладанняў для інтэрнэт-крамы кандытарскіх вырабаў для платформы Android.

**Вынікамі з'яўляюцца** **–** мабільнае прыкладанне для інтэрнэт-крамы кандытарскіх вырабаў для платформы Android.

**Вобласць ужывання** **–** для любога смартфона з аперацыйнай сістэмай Android.

ESSAY

Undergraduate Practice Report, 37 p., 30 illustrations.

**Keywords**: ANDROID, MOBILE APPLICATION, ONLINE STORE, RETROFIT, CLIENT, SERVER.

**Object of research –** mobile application for an online store.

**Purpose –** to consider technologies for developing client-server applications, as well as to design an application for an online confectionery store.

**Methods of research** **–** a) theoretical methods: a study of the literature which was devoted to the development of mobile applications; b) practical methods: modeling, designing specifications and developing a mobile application for an online confectionery store for the Android platform.

**The results are** **–** a mobile application for an online confectionery store for the Android platform.

**Scope** **–** for any smartphone with the Android operating system.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 6](#_Toc161876268)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc161876269)

[1.1 Постановка задачи 8](#_Toc161876270)

[1.2 Описание предметной области 9](#_Toc161876271)

[1.3 Анализ существующих мобильных решений 11](#_Toc161876272)

[1.3.1 «Торты.бел» 11](#_Toc161876273)

[1.3.2 The Candy Closet 12](#_Toc161876274)

[1.3.3 CakeFizz 14](#_Toc161876275)

[1.4 Технологии разработки клиент-серверных приложений 15](#_Toc161876276)

[1.4.1 Технологии разработки клиент-серверных приложений на языке Kotlin 16](#_Toc161876277)

[1.4.2 Технологии разработки клиент-серверных приложений на языке Python 17](#_Toc161876278)

[ГЛАВА 2 РАЗРАБОТКА ТРЕБОВАНИЙ И ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ 18](#_Toc161876279)

[2.1 Требования к разработке мобильного приложения 18](#_Toc161876280)

[2.2 Проектирование UML-диаграмм 20](#_Toc161876281)

[2.2.1 Контекстная диаграмма 20](#_Toc161876282)

[2.2.2 Диаграмма вариантов использования 21](#_Toc161876283)

[2.2.3 Диаграмма деятельности 23](#_Toc161876284)

[2.3 Проектирование базы данных 26](#_Toc161876285)

[2.4 Проектирование прототипа мобильного приложения 28](#_Toc161876286)

[ГЛАВА 3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 32](#_Toc161876287)

[3.1 Реализация серверной части 32](#_Toc161876288)

[3.2 Реализация клиентской части 34](#_Toc161876289)

[3.2.1 Экраны приложения 34](#_Toc161876290)

[3.2.2 Реализация пользовательского интерфейса 38](#_Toc161876291)

[3.2.3 Управление навигацией в мобильном приложении 39](#_Toc161876292)

[3.2.2 Взаимодействие с сервером 40](#_Toc161876293)

[ЗАКЛЮЧЕНИЕ 43](#_Toc161876294)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 44](#_Toc161876295)

# **ВВЕДЕНИЕ**

В современном мире электронная коммерция стала неотъемлемой частью нашей повседневной жизни. Каждый день всё больше людей предпочитают делать покупки онлайн вместо походов в магазины. Вдобавок быстрое развитие технологий и всеобщее распространение смартфонов привело к тому, что всё больше людей предпочитают делать покупки именно с помощью своих мобильных телефонов.

Сегодня разработка мобильного приложения для интернет-магазина кажется, как никогда, актуальной, так как это позволит компании расширить аудиторию, упросить пользователям покупку, оперативно информировать их о скидках и акциях с помощью уведомлений на смартфоне, а также собирать и анализировать ценные данные о своей клиентской базе для дальнейшего их анализа и повышения эффективности продаж и конкурентоспособности компании.

Разрабатываемое приложение будет предлагать широкий функционал и возможности, чтобы удовлетворить пользователей интернет-магазина. Помимо функционала также будет уделено внимание дизайну и навигации, чтобы улучшить пользовательский опыт взаимодействия с приложением. Главные функции разрабатываемого мобильного приложения для интернет-магазина: обзор каталога товаров, поиск и фильтрация товаров, корзина и оформление заказа, оплата заказа, личный кабинет и профиль пользователя, а также уведомления и акции.

Приложение будет разрабатываться для платформы Android, так как на данный момент эта операционная система является самой распространённой на мобильных устройствах, что позволит добиться большего числа пользователей разрабатываемого мобильного приложения.

Также для обеспечения работы мобильного приложения и обработки данных, связанных с интернет-магазином кондитерских изделий, будет разработан веб-сервер, который будет взаимодействовать с базой данных.

Методы исследования, которые применялись в работе, включают в себя анализ и моделирование.

В главе 1 будет сформулирована задача курсового проекта, изучена предметная область, в рамках которой будет функционировать мобильное приложение. Будет проведён анализ существующих мобильных решений с целью выявить их недостатки и достоинства и использовать полученную информацию в проектировании мобильного приложения. Также будут подробно рассмотрены основные технологии, которые применяются в разработке клиент-серверных приложений.

В главе 2 будет спроектировано мобильное приложение. Будет разработана спецификация и требования к функционалу приложения. В частности, будут представлены контекстная диаграмма, диаграммы вариантов использования, диаграммы деятельности, схема базы данных и прототипы экранов приложения.

В главе 3 будет показана реализация прототипа мобильного приложения для интернет-магазина кондитерских изделий.

# **ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Постановка задачи**

Главной задачей курсового проекта является проектирование мобильного приложения для интернет-магазина кондитерских изделий. Проектируемое мобильное приложение должно иметь современный привлекательный интерфейс с интуитивно понятной навигацией и простыми элементами управления.

Для достижения главной задачи курсового проекта требуется решить следующие подзадачи:

1. Необходимо спроектировать привлекательный и интуитивно понятный пользовательский интерфейс мобильного приложения, который обеспечит легкую навигацию по каталогу товаров, возможность просмотра подробной информации о кондитерских изделиях и удобный способ оформления заказа.
2. Необходимо спроектировать простой и удобный механизм для аутентификации и регистрации пользователя.
3. Необходимо спроектировать основной функционал мобильного приложения, включая просмотр каталога товаров, добавление товара в корзину, механизм оформления заказа и онлайн-оплаты.
4. Необходимо спроектировать мобильное приложение так, чтобы обеспечить эффективное взаимодействие с сервером, где хранится информация о кондитерских изделиях, заказах и прочем.
5. При проектировании мобильного приложения необходимо учитывать необходимость обеспечения безопасности пользовательских данных, включая личные данные и данные о заказах, при обмене этими данными между мобильным приложением и сервером.
6. Необходимо спроектировать серверную часть приложения и структуру базы данных так, чтобы обеспечить эффективное взаимодействие с базой данных, простоту внесения изменений в базу данных при добавлении новой функциональности в приложение.

## **Описание предметной области**

Интернет-магазин – это электронная платформа, предназначенная для продажи товаров или услуг широкому кругу потребителей через интернет.

Именно интернет-магазины определяют развитие такой сферы мировой экономики, как электронная коммерция. Они помогают компаниям увеличить прибыль, а также предоставляют возможность отказаться от необходимости в офлайн-помещении для реализации товара.

Все интернет-магазины работают по одной схеме. Покупатель выбирает желаемые товары на сайте или в мобильном приложении. Затем оформляет заказ, указывая необходимые данные для доставки и оплаты. После подтверждения операции, компания доставляет заказ покупателю. Некоторые интернет-магазины поддерживают возможность оплаты товара наличными средствами при получении от курьера.

Существует несколько моделей ведения бизнеса, в которых может быть задействован интернет-магазин:

1. Онлайн-бизнес. Только интернет-магазин используется для продажи товара. Просмотр, выбор и заказ товаров осуществляется через сайт или мобильное приложение. Физические магазины у компании отсутствуют.
2. Интернет-магазин как дополнительный канал продаж. В этом случае у компании есть офлайн-бизнес, а интернет-магазин рассматривается как дополнительный источник дохода и средство для привлечения новых клиентов.
3. Аутсорс-услуги. В этой модели владелец интернет-магазина выступает в качестве посредника и обеспечивает платформу для продажи товаров офлайн-магазинов, получая при этом комиссию с каждой сделки, оформленной на его электронной платформе.
4. Дропшиппинг. Это модель ведения бизнеса, при которой интернет-магазин занимается только продажей товара. При этом товар хранится на складе поставщика и доставку покупателю обеспечивает тоже поставщик.

По типу продаж интернет-магазины делятся на оптовые и розничные. По ассортименту их можно разделить на узкоспециализированные и интернет-магазины с широким ассортиментом.

Основными компонентами интернет-магазина являются: главная страница, каталог, карточки товаров, поиск, «корзина», личный кабинет пользователя и служебные страницы.

Главная страница – это первое, что видит пользователь. Поэтому она быть запоминающейся и производить хорошее впечатление. На ней должна быть собрана краткая информация об интернет-магазине: логотип, график работы, акции, ссылки на основные разделы. Однако, если речь про мобильные приложения, то чаще всего в качестве главной страницы выступает страница с каталогом.

Каталог представляет собой список товаров, которые разделены на соответствующие категории и подкатегории, которые обычно упорядочены с помощью иерархической структуры. В этом типе структуры пользователь может переходить от общих категорий к более конкретным.

Карточки товаров содержат полное описание товара. Обычно они включают в себя фотографии, краткое описание, характеристики, стоимость и отзывы покупателей.

Поиск является важнейшим компонентом интернет-магазина, так как обеспечивает для пользователя возможность быстро и легко найти нужный товар вместо того, чтобы искать его в общем каталоге. По ключевым словам поиск выдаёт пользователю набор релевантных результатов и упрощает процесс покупки. Также важную роль в поиске играют фильтры, которые помогут сузить выборку товара и исключить ненужные варианты.

«Корзина» является компонентом интернет-магазина, ответственным за оформление заказа. Обычно «корзина» содержит список всех выбранных товаров и возможность редактировать его (изменить количество, удалить), стоимость каждого товара и суммарную стоимость заказа, способы получения товара (доставка, самовывоз) и необходимая для них информация, а также способ оплаты.

Личный кабинет пользователя позволяет содержит персональную информацию о покупателе. Позволяет частым покупателям сохранять данные для оплаты, адреса доставки, совершённые покупки и откладывать товары для покупки в будущем. Всё это значительно упрощает оформление заказа.

К служебным страницам относятся все разделы, которые носят вспомогательный характер. Например, разделы с информацией об оплате, доставке, адресах пунктов самовывоза, условиях возврата и обмена товаров, часто задаваемые пользователями вопросы и др. Эти разделы призваны обеспечить информационную поддержку пользователя, улучшить пользовательский опыт и способствовать успешному развитию бизнеса.

## **Анализ существующих мобильных решений**

### **«Торты.бел»**

«Торты.бел» - это мобильное приложение кондитерской студии, предлагающее широкий ассортимент кондитерских изделий, включая торты, штрудели и пирожные. Приложение сразу при входе предлагает выбрать способ доставки: доставка или самовывоз, что кажется полезной функцией. Далее у пользователя есть возможность:

* просмотреть меню (рис. 1.1);
* зарегистрироваться или войти в свой профиль;
* изменить способ доставки;
* посмотреть историю заказов;
* посмотреть информацию о кондитерской студии;
* оставить отзыв;
* собрать и оформить заказ.



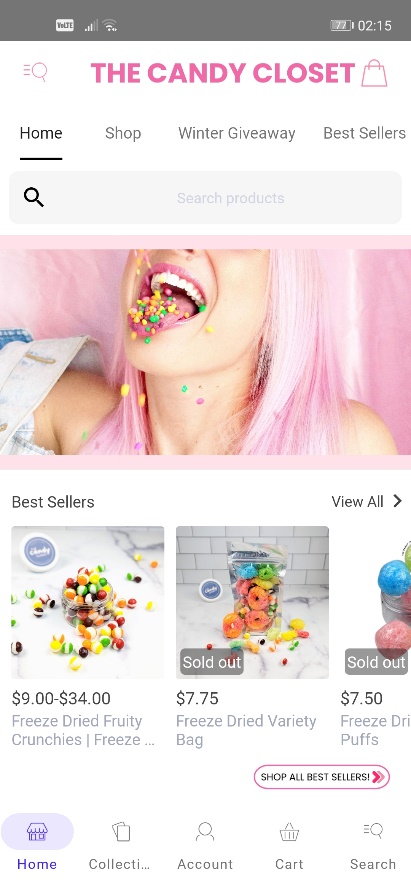
**Рисунок 1.1 Страница меню мобильного приложения "Торты.бел"**

Из плюсов мобильного приложения можно выделить широкий функционал, удовлетворяющий все потребности пользователя. Возможность авторизации через Telegram, сохранение адресов для доставки – положительные моменты с точки зрения пользовательского опыта.

Однако, есть существенные недостатки, такие как отсутствие возможности открыть карточку товара, отсутствие отзывов на товары. Также можно отметить, что иконка кнопки для отображения информации о стоимости доставке не соответствует своей сути, а в конце каждой карточки товара расположена надпись kg, которой не нашлось объяснения.

### **The Candy Closet**

«The Candy Closet» - мобильное приложение международной компании, которое предлагает пользователям помимо широкого ассортимента кондитерских изделий со всего мира эксклюзивные скидки и розыгрыши. При входе в мобильное приложение открывается главная страница (рис. 1.2), которая содержит информацию о наиболее популярных товарах, актуальных скидках, новых поступлениях и ссылки на социальные сети компании и основные разделы сайта.



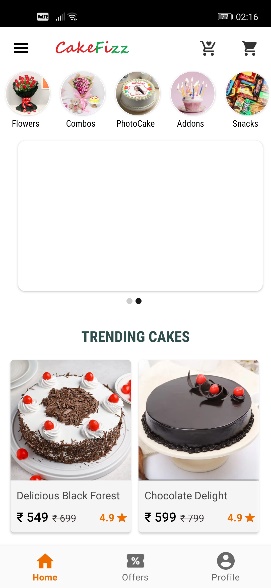
**Рисунок 1.2 Главная страница мобильного приложения "The Candy Closet"**

Мобильные приложение предоставляет пользователю основной функционал интернет-магазина. Однако есть у приложения отличительные элементы, такие как раздел «Интересные факты» и зимний розыгрыш. Это положительно сказывается на привлечении новых клиентов. Также положительно на пользовательском опыте сказывается возможность открыть карточку товара и выбрать параметры товара перед добавлением в корзину и возможность поделится товаром в один клик.

Есть у приложения и недостатки, к которым можно отнести паузы перед отображением информации и несколько запутанное навигационное меню, к которому нужно будет привыкнуть.

### **CakeFizz**

«CakeFizz» - это индийское мобильное приложение, предлагающее широкий ассортимент кондитерских изделий. При входе в мобильное приложение пользователь видит страницу каталога, ссылки на профиль и страницу с купонами (рис. 1.3).



**Рисунок 1.3 Главная страница мобильного приложения "CakeFizz"**

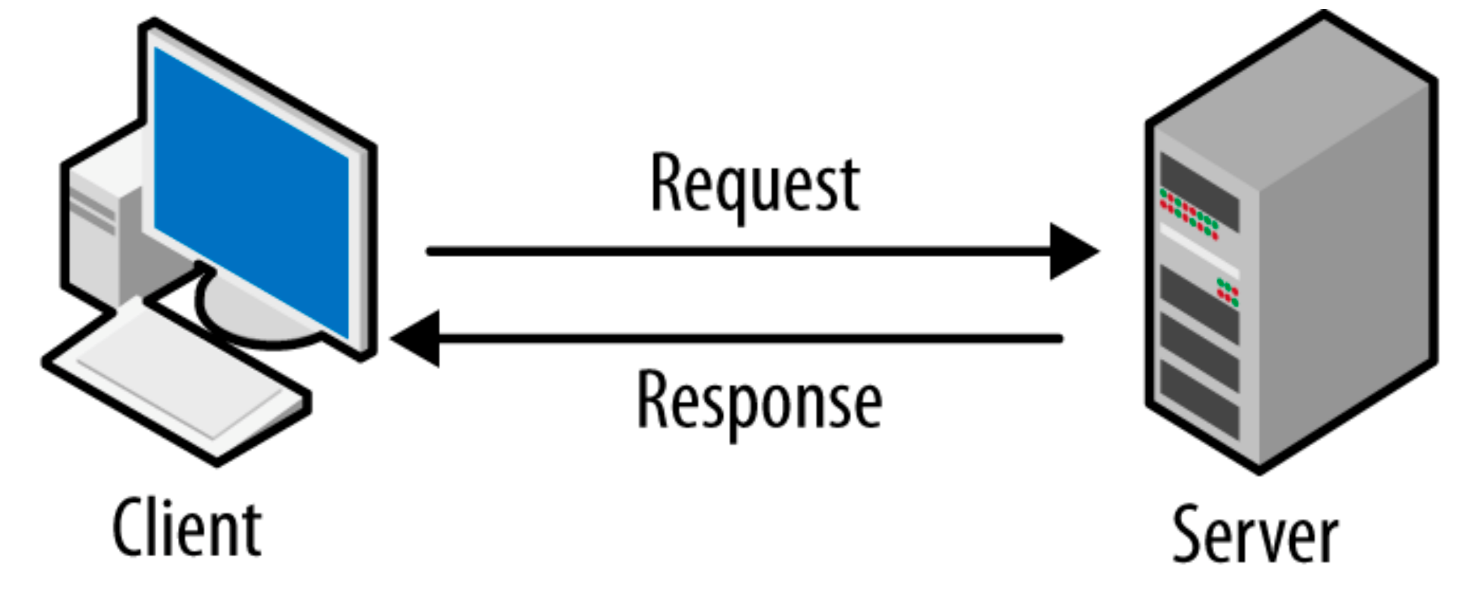
Мобильные приложение предоставляет пользователю основной функционал интернет-магазина. Отличительной чертой этого интернет-магазина можно назвать раздел с купонами, которые предоставляют пользователям скидки на покупки. Наличие раздела «Избранное», рейтинг товаров и отзывы на них, а также предлагаемые похожие товары на странице товара выделяют это приложение среди конкурентов.

К явным недостаткам можно отнести нерабочие ссылки на страницы «О нас» и «Условия использования».

## **Технологии разработки клиент-серверных приложений**

Клиент-серверным называется приложение, которые имеет клиент-серверную архитектуру.

«Клиент-сервер» - это сетевая архитектура, в которой сетевая нагрузка распределена между серверами и клиентами. Клиентом может быть любое устройство, которые запрашивает у сервера данные. Сервер – это устройство или программа, предназначенная для обработки запросов пользователей. Таким образом, можно сказать, что взаимодействие между клиентом и сервером происходит по принципу «запрос - ответ» (рис. 1.4).



**Рисунок 1.4 Клиент-серверная архитектура**

Для взаимодействия клиента с сервером, обычно разрабатываются интерфейсы прикладного программирования (API), которые определяют правила связи клиента с сервером. Существует программная архитектура REST (Representation State Transfer), которая определяет условия работы API. API-интерфейсы называются REST API (RESTful API), если они соответствуют архитектуре REST. При этом веб-службы, которые реализуют эту архитектуру, называют RESTful веб-службами.

RESTful API обычно реализуются с помощью протокола HTTP (протокол передачи гипертекста). Основные методы этого протоколы: GET, POST, PUT, DELETE, - указывают серверу, что необходимо сделать с получаемой информацией.

Основные преимущества RESTful API: возможность масштабирования, гибкость и независимость.

### **Технологии разработки клиент-серверных приложений на языке Kotlin**

В Kotlin есть две основные библиотеки для взаимодействия с сетью: Volley и Retrofit.

Библиотека Retrofit, авторами которой являются разработчики из компании «Square», позволяет создать полноценный RESTful клиент. Retrofit предлагает следующие возможности:

* Простое описание запросов с помощью аннотаций;
* Поддержка приоретизации запросов;
* Поддержка отмены запросов и повторных запросов;
* Поддержка синхронных и асинхронных вызовов REST-методов;
* Возможность работы с RxJava;
* Поддержка параллельных запросов и библиотеки OkHttp;

Volley – HTTP-библиотека, разработанная командой Google, которая упрощает и ускоряет работу в сети для мобильных приложения на Android. Volley предлагает следующие преимущества:

* Автоматическое планирование сетевых запросов;
* Поддержка параллельных запросов и библиотеки OkHttp;
* Поддержка приоретизации запросов;
* Поддержка отмены запросов и повторных запросов;
* Гибкий механизм кэширования;
* Инструменты отладки и отслеживания;

В целом, можно сказать, что библиотеки во многом схожи. Каждая из них используется в зависимости от потребностей и предпочтений разработчика. Retrofit чаще используется, если необходимо работать со стандартной реализацией REST API или асинхронными запросами. Volley же используется в случае, когда необходима важно эффективное кэширование или для простых сетевых запросов.

### **Технологии разработки клиент-серверных приложений на языке Python**

Существует множество популярных библиотек для веб-разработки на языке Python. Среди прочих можно выделить две самые популярные: Flask и Django.

Django – это библиотека для разработки веб-приложений, созданная Адрианом Головати и Саймоном Уиллисоном в 2005 году. Использует объектно-ориентированный подход к разработке. Отлично подходит для работы с крупными проектами. Поддерживает популярные системы управления реляционными БД. Не поддерживает отладку. Присутствует поддержка динамических HTML-страниц.

Flask – это микробиблиотека для разработки веб-приложений, разработанная Армином Ронахером. Использует модульный подход к разработке. Лучше всего применяется в небольших проектах. Использует стороннюю библиотеку SQLAlchemy для взаимодействия с базой данных. Поддерживает виртуальную отладку. Отсутствует поддержка динамических HTML-страниц.

Можно сказать, что Django обладает большим функционалом и больше подходит для разработки крупных веб-приложений с чёткой структурой и максимальной масштабируемостью. Также Django следует использовать, если необходим высокий уровень защиты данных.

В то время, как Flask является более легковесным и для расширения своего функционала использует сторонние библиотеки. Поэтому Flask лучше подходит для небольших проектов, где важна гибкость. При этом от разработчика не требуется предварительное изучение библиотеки.

**Выводы**

1. Изучена предметная область интернет-магазина.
2. Проанализированы аналогичные мобильные решения, выявлены их недостатки и достоинства.
3. Рассмотрены технологии для разработки клиент-серверных приложений.

# **ГЛАВА 2 РАЗРАБОТКА ТРЕБОВАНИЙ И ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

## **Требования к разработке мобильного приложения**

Требования к разработке мобильных приложений делятся на функциональные и нефункциональные.

Функциональные требования описывает основные функциональные возможности разрабатываемой системы, которыми она должна обладать для достижения поставленной цели. Функциональные требования определяют взаимодействие между программным обеспечением и его пользователями, а также поведение системы при различных обстоятельствах. При определении функциональных требований важно ориентироваться на удовлетворение потребностей и ожиданий пользователей. Функциональные требования должна формулироваться конкретно, не оставляя места для двусмысленности. На этапе проектирования мобильного приложения были выявлены следующие функциональные требования:

* Мобильное приложение должно предоставлять пользователю возможность регистрации и аутентификации;
* Мобильное приложение должно отображать каталог всех доступных кондитерских изделий с разделением по категориям;
* Мобильное приложение должно предоставлять возможность посмотреть подробную информацию о товаре, его фотографии и отзывы о нём;
* Мобильное приложение должно предоставлять пользователям возможность оставить отзыв на товар;
* Мобильное приложение должно предоставлять возможность добавлять товары в корзину для дальнейшего оформления заказа;
* Мобильное приложение должно предоставлять возможность оформления заказа с указанием параметров доставки (адреса, времени и др.);
* Мобильное приложение должно поддерживать возможность онлайн-оплаты заказа;
* Мобильное приложение должно предоставлять пользователю возможность управления аккаунтом;
* Мобильное приложение должно отправлять пользователю уведомления о специальных предложениях, акциях и оформлении заказа.

Нефункциональные требования определяют качества и характеристики разрабатываемой системы, описывают ограничения и требования, которые прямо не связаны с функциональностью системы. Нефункциональные требования сосредоточены на таких аспектах, как производительность, надёжность, безопасность и др. На этапе проектирования мобильного приложения были выявлены следующие нефункциональные требования:

* Мобильное приложение должно быть реализовано под платформу Android;
* Мобильное приложение должно поддерживаться устройствами с установленной операционной системой Android, начиная с версии 9.0;
* Мобильное приложение должно работать только в режиме портретной ориентации экрана;
* Мобильное приложение должно иметь адаптивный дизайн для разных размеров экранов;
* Мобильное приложение должно быть простым в использовании для любого пользователя;
* Мобильное приложение должно иметь интуитивно понятный интерфейс;
* Мобильное приложение должно быть адаптировано под тёмную тему на смартфоне;
* Мобильное приложение должно поддерживать высокая скорость при работе с несколькими пользователями;
* Мобильное приложение должно обеспечивать быстрый отклик и минимальное время для загрузки контента;
* Мобильное приложение должно обеспечивать безопасность пользовательских персональных данных;
* Мобильное приложение должно быть масштабируемым.

## **Проектирование UML-диаграмм**

UML (Unified Modeling Language) – это графические язык, который предназначен для визуализации и проектирования программного обеспечения. UML-диаграммы являются важнейшим элементов в разработке программного обеспечения, так как позволяют разработчикам лучше понять требования, функциональность и архитектуру системы.

На этапе проектирования были разработаны следующие виды UML-диаграмм: контекстная диаграмма, диаграмма декомпозиции, диаграмма вариантов использования, диаграмма деятельности.

### **Контекстная диаграмма**

Контекстная диаграмма позволяет описать внешний свойства системы, но не её содержание. Контекстная диаграмма обычно отображает взаимодействие между приложением и внешними элементами (пользователи, другие системы и др.), помогая определить границы системы и перемещение данных между приложением и его внешним окружением.



**Рисунок 2.1 Контекстная диаграмма**

### **Диаграмма вариантов использования**

Диаграмма вариантов использования – это визуальное представление функциональных требований к системе. Диаграмма вариантов использования отображает взаимодействие между акторами и системой, а также визуально представить основные сценарии использования системы.

В рамках разрабатываемой системы предусмотрен два актора: Пользователь, который будет взаимодействовать с системой, и Платежная система, которая будет отвечать за обработку платежей. В ходе проектирования мобильного приложения была составлена общая диаграмма вариантов использования (рис. 2.2).

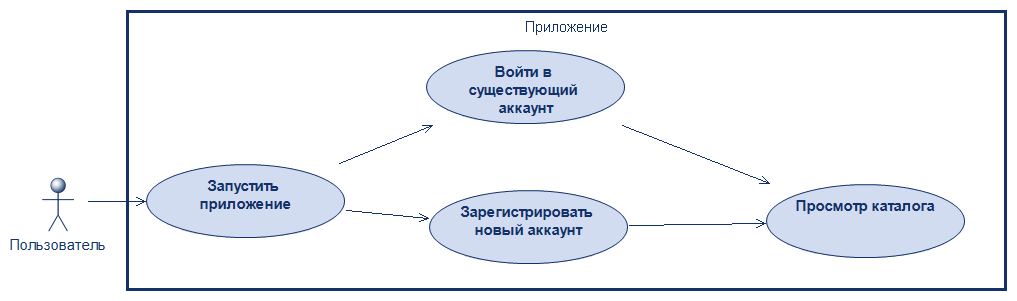


**Рисунок 2.2 Общая диаграмма вариантов использования**

Пользователь входит в приложение, после чего он может просмотреть каталог и добавить нужный товар в корзину либо сохранить его в «Избранное». После добавления нужных товаров в корзину пользователь может оформить заказ, после чего приступить к оплате заказа. Обработку платежа производит платёжная система.

В ходе подготовки спецификации, используемой для разработки приложения, также были составлены несколько основных диаграмм вариантов использования:

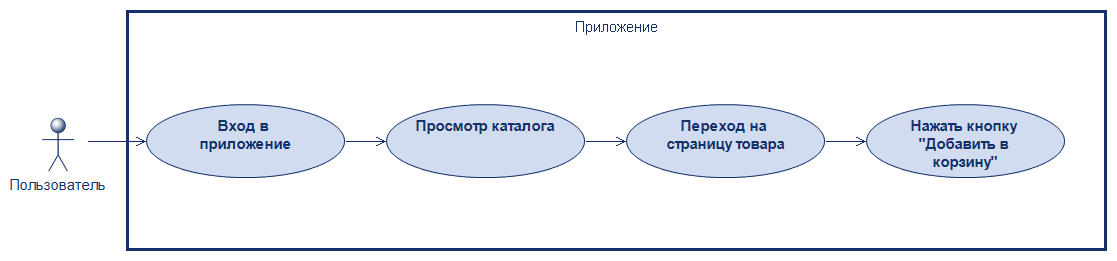
1. Вариант использования «Вход в приложение»



**Рисунок 2.3 Диаграмма варианта использования "Вход в приложение"**

Пользователь запускает приложение и далее может продолжить работу в нём один из двух возможных способов: войти в существующий аккаунт или зарегистрировать новый аккаунт. После регистрации или входа пользователь переходит к просмотру каталога.

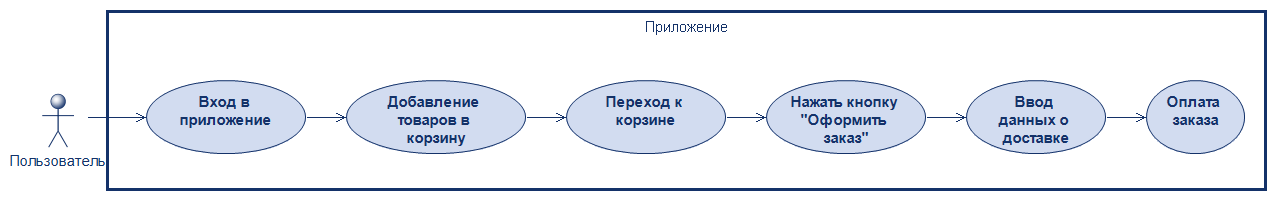
1. Вариант использования «Добавление товара в корзину»



**Рисунок 2.4 Диаграмма варианта использования «Добавление товара в корзину»**

Пользователь входит в приложение, изучает ассортимент товаров в каталоге, выбирает нужный товар и переходит на его страницу, где для добавления в корзину нажимает на соответствующую кнопку.

1. Вариант использования «Оформление заказа»



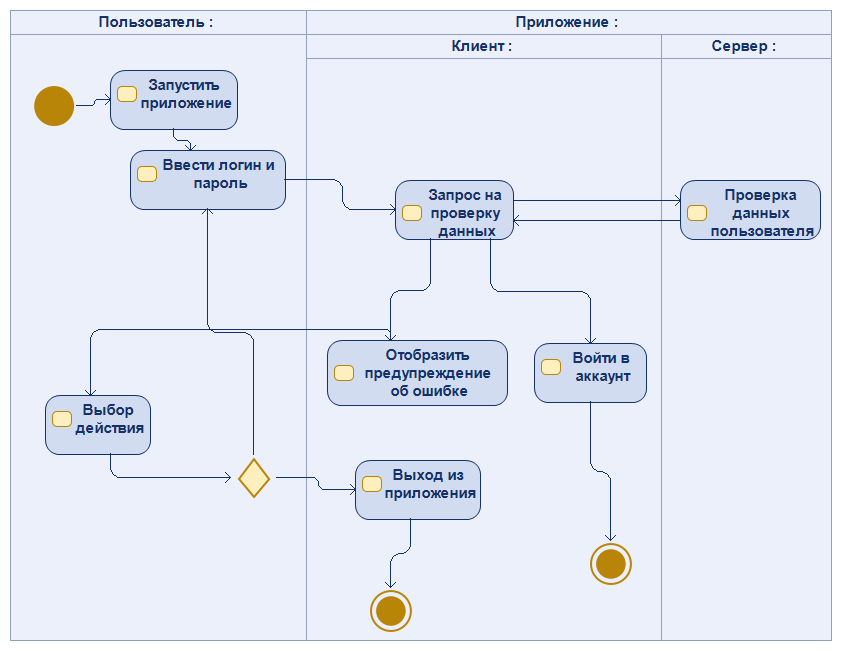
**Рисунок 2.5 Диаграмма варианта использования "Оформить заказ"**

Пользователь входит в приложение, добавляет необходимые товары в корзину. Затем переходит в корзину и нажимает кнопку «Оформить заказ». После этого заполняет данные, необходимые для осуществления доставки товара и переходит к оплате заказ.

### **Диаграмма деятельности**

Диаграмма деятельности – это UML-диаграмма, которая отображает этапы в рамках определенного процесса, который может происходит в системе. Диаграмма деятельности позволяет лучше понять логику и этапы выполнения определённого процесса в системе, а также выявить возможные проблемы в процессе.

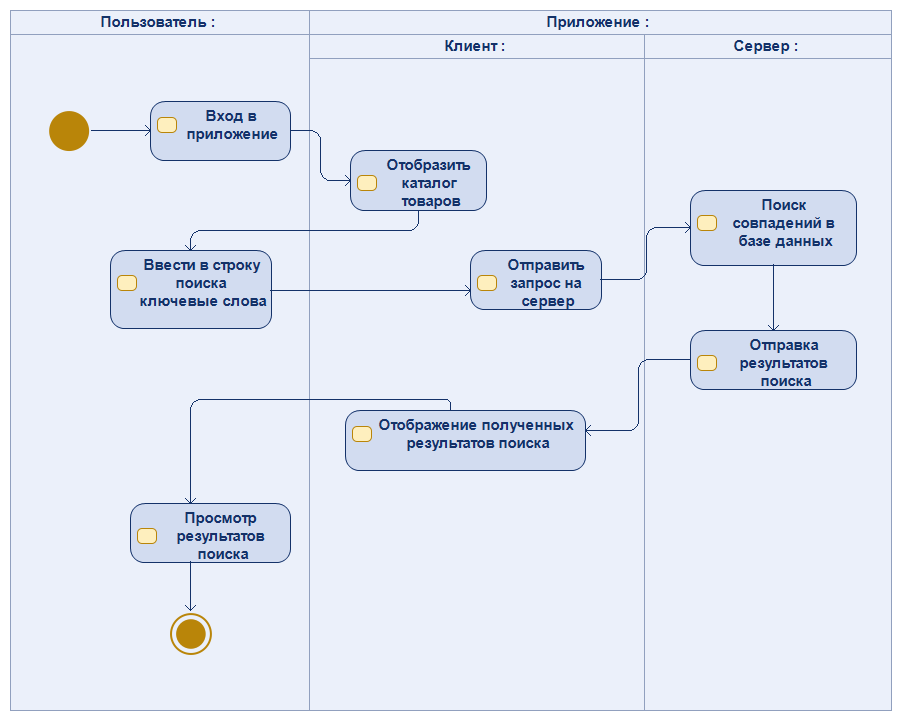
Далее будут приведены диаграммы деятельности, которые были составлены в ходе подготовки спецификации, используемой для разработки мобильного приложения.

****

**Рисунок 2.6 Диаграмма деятельности "Аутентификация"**

Представленная на рисунке 2.6 диаграмма подробно описывает процесс аутентификации пользователя.

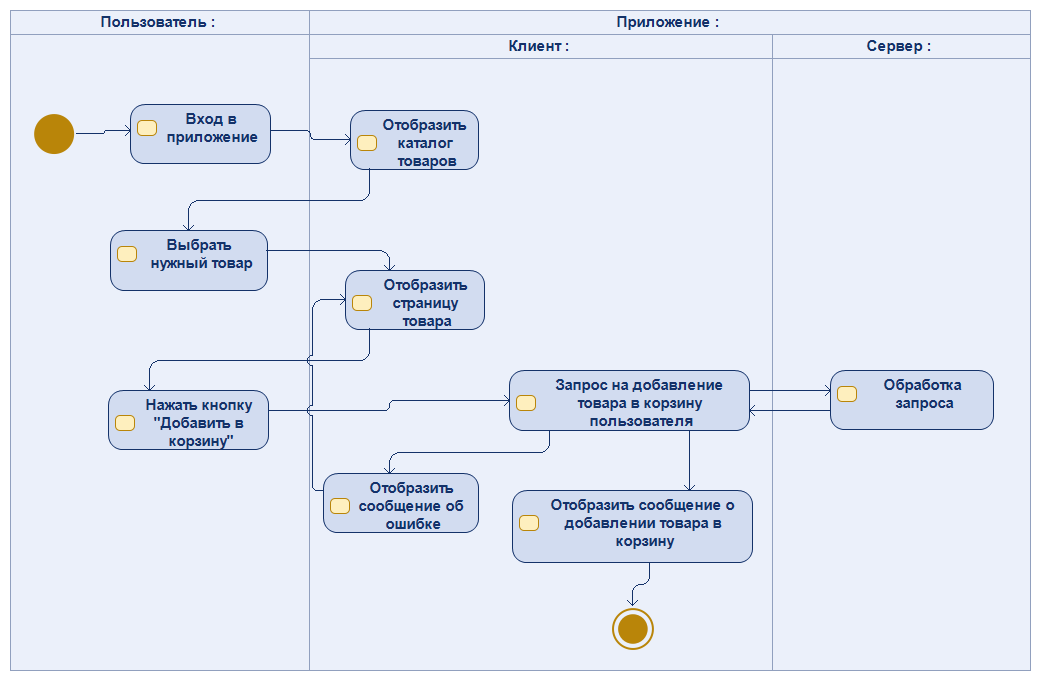
Для использования приложения пользователь должен пройти процедуру аутентификации. Для этого пользователь вводит логин и пароль, указанные при регистрации. Затем мобильное приложение отправляет запрос на сервер для подтверждения наличия такого пользователя в базе данных. При успешной аутентификации пользователь входи в свой аккаунт и продолжает работу с приложением. В противном случае пользователь видит получает предупреждение о неправильном вводе и может либо выйти из приложения, либо ещё раз попробовать ввести правильные данные.



**Рисунок 2.7 Диаграмма деятельности "Поиск товара"**

Представленная на рисунке 2.7 диаграмма подробно описывает процесс поиска пользователем нужно товара.

Войдя в приложение, пользователю наблюдает каталог товаров и строку поиска. В неё он вводит ключевые слова для поиска необходимого товара. Затем клиентская часть приложения отправляет запрос на сервер, который по ключевым словам ищет совпадения в базе данных и отправляет клиентской части ответ в виде результатов поиска, которые отображаются на экране.



**Рисунок 2.8 Диаграмма деятельности "Добавление товара в корзину"**

Представленная на рисунке 2.8 диаграмма подробно описывает процесс добавления пользователем нужно товара в корзину для дальнейшего оформления заказа.

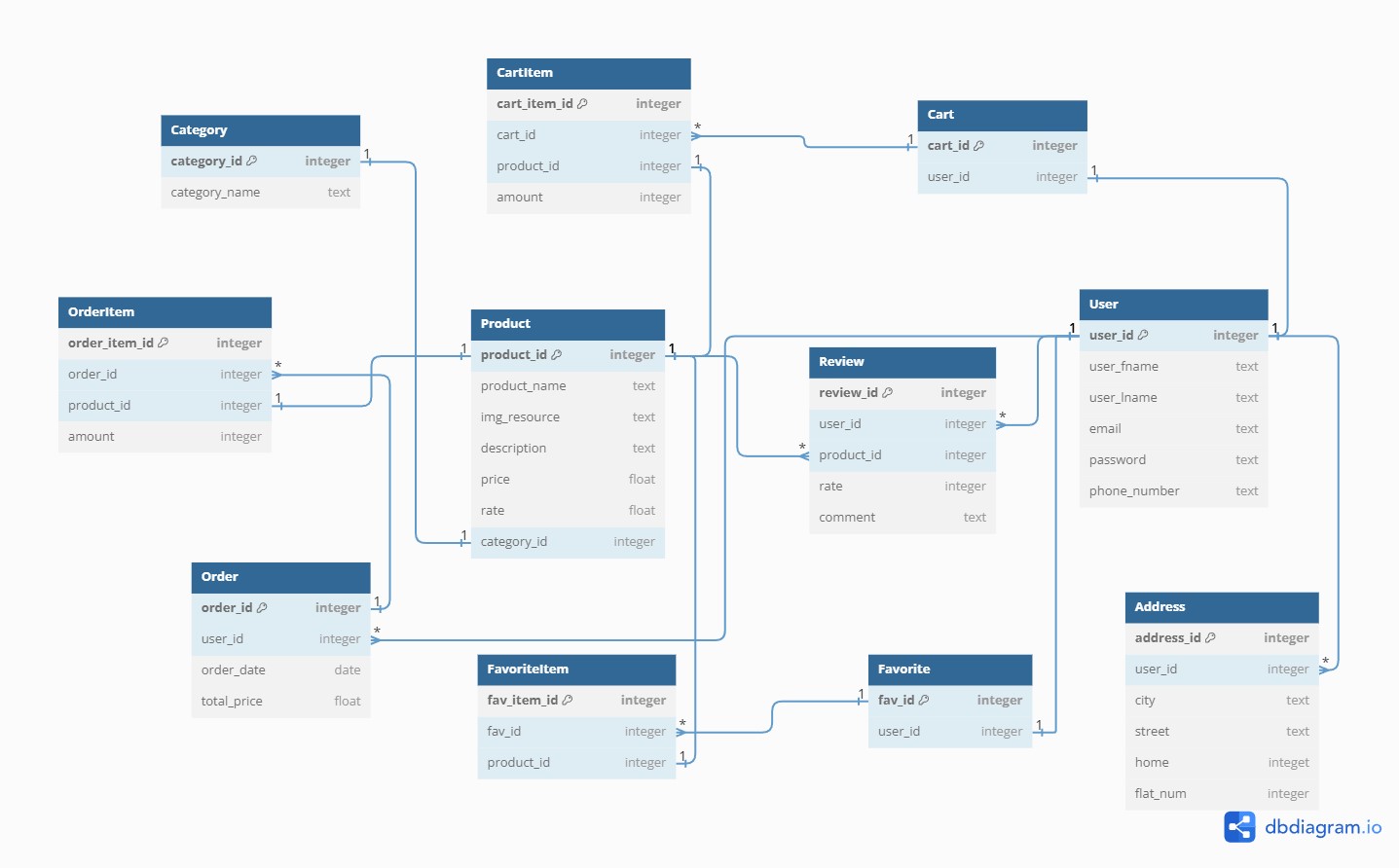
Войдя в приложение, пользователю наблюдает каталог товаров и строку поиска. Выбирая нужные товар, пользователь попадает на страницу этого товара. После нажатия на кнопку «Добавить в корзину» клиентская часть приложения отправляет запрос на сервер для внесения соответствующих изменения в базу данных. В случае успешного внесения изменений пользователь уведомляется об успешном добавлении товара в корзину. В противном случае пользователь видит сообщение об ошибке, оставаясь на странице товара.

## **Проектирование базы данных**

Функционал проектируемого приложения подразумевает под собой необходимость хранения следующих данных:

* регистрационные данные пользователей;
* история заказов пользователей;
* адресах доставки, платёжные данные пользователей;
* данные об «избранных» товарах пользователя;
* данных о товарах (описание, фото, цена, наличие на складе и др.);

Для хранения перечисленных выше данных и обеспечения быстрого доступа к ним будет использоваться база данных. Для её проектирования были выделены необходимые сущности, их атрибуты, а также определены связи между таблицами, представляющими эти сущности. Полученная схема базы данных представлена на рисунке 2.9.



**Рисунок 2.9 Схема базы данных**

База данных будет состоять из 11 таблиц. Далее будут подробно описаны эти таблицы и связи между ними.

Таблица User будет сохранять данные о пользователе: имя, фамилию, email, пароль и телефонный номер. В качестве первичного ключа выступает атрибут user\_id, а поле address\_id – это внешний ключ, который связывает эту таблицу с таблицей Address, которая представляет собой данные об адресе для доставки (город, улицу и номер квартиры). Между этими таблицами установлена связь «один ко многим», т.к. у пользователя есть возможность добавить несколько адресов для доставки.

Таблица Product содержит информацию о товаре: название товара, его описание, ссылку на изображение товара, цену и среднюю оценку товара, а также количество доступных к покупке экземпляров. Первичным ключом в этой таблице является product\_id, а атрибут category\_id – это внешний ключ, который связывает эту таблицу с таблицей Category. Таблица Category будет сохранять категории товаров для удобного представления в каталоге и имеет 2 атрибута: первичный ключ category\_id и название категории category\_name.

Таблица Order будет содержать информацию о заказе: дату оформления заказа, сумму покупок и id пользователя, который оформил заказ. Первичный ключ в этой таблице – order\_id. Таблица Order связана с таблицей OrderItem связью «один ко многим», т.к. таблица OrderItem представляет собой данные о элементе заказа: id товара и его количество.

Аналогичным образом спроектированы таблица Cart и CartItem, Favorite и FavoriteItem для хранения данных о корзине пользователя и «избранных» товарах.

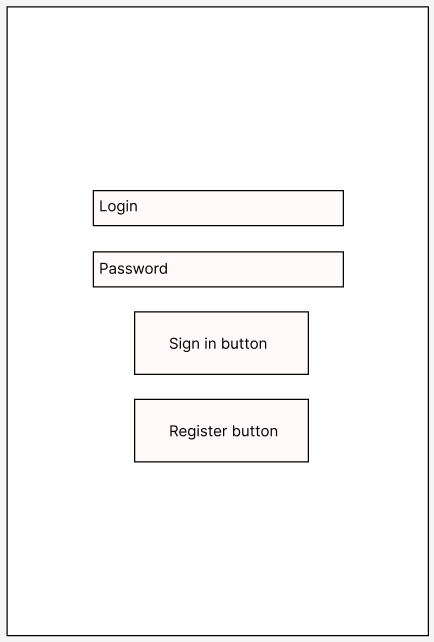
Таблица Review будет сохранять данные об отзывах на товары и содержит следующие атрибуты: первичный ключ review\_id, id продукта, на который оставляют отзыв, id пользователя, который оставляет отзыв, отзыв и оценка товару. Таблица Review связана с таблицами Product и User связью «многие к одному».

В описанной базе данных все таблицы находятся в третьей нормальной форме, т.е. все их атрибуты атомарные и каждая колонка содержит уникальные данные, все не ключевые атрибуты полностью и не транзитивно зависят от первичного ключа. Это значит, что спроектированная база данных соответствует требованиям нормализации и обеспечивает эффективное хранение данных.

## **Проектирование прототипа мобильного приложения**

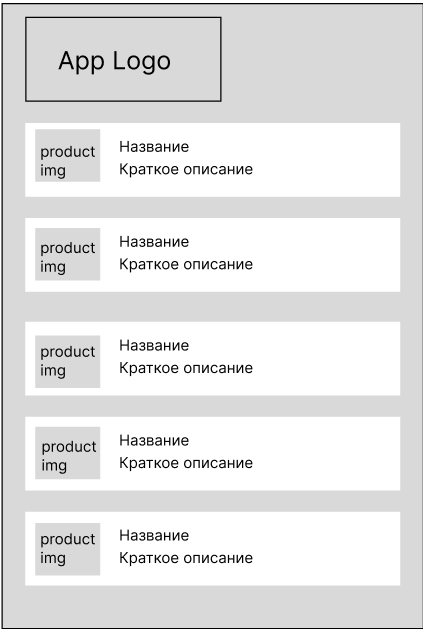
Приложение будет иметь 4 основных экрана: экран регистрации/авторизации, экран каталога товаров, экран с информацией о товаре и экран оформления заказа. Навигация между экранами будет реализована с помощью «гамбургер-меню» - бокового меню, которое выплывает из-за левой границы экрана только по нажатию на кнопку, а в остальное время невидимо для пользователя.

На экране регистрации/авторизации (рис. 2.10) расположены поля для ввода пароль, логина и кнопки регистрации/авторизации.



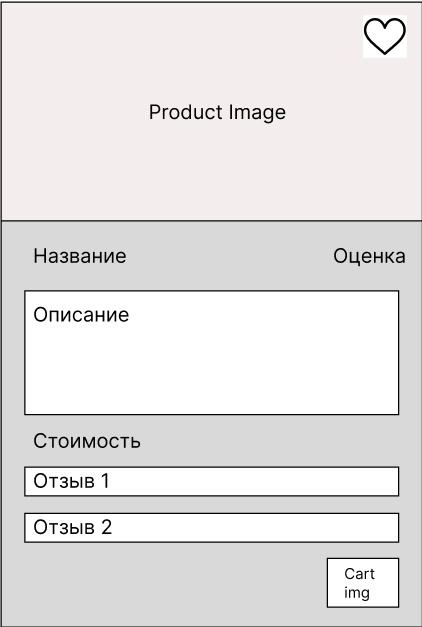
**Рисунок 2.10 Прототип экрана регистрации/авторизации**

На экране каталога товаров (рис. 2.11) будет расположен логотип приложения в верхнем левом углу, а также список товаров. Каждый элемент списка будет содержать фотографию, название и краткое описание товара.



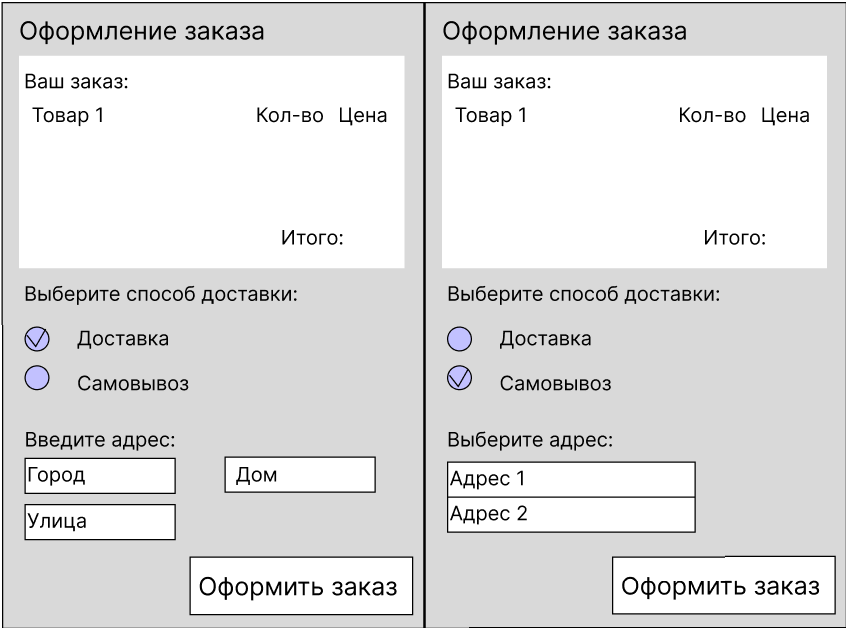
**Рисунок 2.11 Прототип экрана каталога товаров**

На экране с информацией о товаре (рис. 2.12) будет размещена фотография товара, его название, подробное описание, стоимость, отзывы пользователей на этот товар и его среднюю оценку. Также в правом нижнем углу будет расположена кнопка для добавления товара в корзину, а в верхнем правом углу будет кнопка для добавления товара в «Избранное».



**Рисунок 2.12 Прототип экрана с информацией о товаре**

На экране оформления заказа будет расположена информация о заказе: перечень выбранных товаров, их количество, стоимость и итоговая сумма заказа. Также можно будет выбрать способ доставки заказа. В зависимости от выбора представлены 2 варианта этого экрана (рис. 2.13). В случае выбора доставки пользователю будет предоставлена возможность ввести адрес для доставки. Если будет выбран вариант «самовывоз», то пользователю будет предложено выбрать адрес пункта выдачи заказов из списка. В нижнем правом углу экрана расположена кнопка «Оформить заказ».



**Рисунок 2.13 Прототипы экрана оформления заказа**

**Выводы**

1. Разработаны функциональные и нефункциональные требования к приложению.
2. Разработаны диаграммы, показывающие поведение пользователя в системе, а также схема базы данных для хранения информации о пользователях и товарах интернет-магазина.
3. Спроектированы прототипы экранов приложения.

# **ГЛАВА 3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

## **3.1 Реализация серверной части**

Серверная часть представляет собой веб-приложение, написанное на языке программирования Python с использованием библиотеки Flask. Взаимодействие с клиентом осуществляется с помощью HTTP-протокола. Сервер принимает HTTP-запросы, которые передают данные в формате JSON в теле запроса или через строку запроса в формате URL в зависимости от метода, обрабатывает и отсылает ответ, включающий статусный код HTTP и необходимые данные также в формате JSON.



**Рисунок 3.1 Метод getCatalog()**

На рис. 3.1 представлен пример метода, который отвечает за обработку запроса на получение каталога товаров. Декоратор @app.route используется в библиотеке Flask для функции с конкретным URL-адресом. Таким образом, когда веб-приложение получит запрос по адресу “/api/v1/catalog”, будет вызвана функция getCatalog() для обработки этого запроса и генерации ответа. В функции getCatalog() создаётся сессия с базой данных, выполняется запрос на получение данных о всех продуктах, затем данные о каждом продукте добавляются в словарь, которые в конце сериализуется в JSON-строку и метод возвращает эту строку в качестве ответа на HTTP-запрос. Причем параметр “ensure\_ascii=False” при сериализации словаря указывает на то, что все символы, которые не являются частью стандартного набора ASCII-символов, сохраняются в исходной форме. Причиной для включения этого параметра стала необходимость передавать символы русского алфавита в JSON-строке.

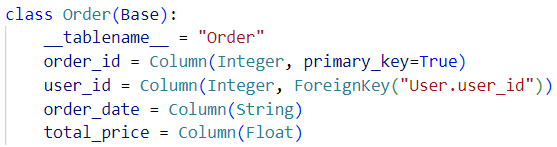
Помимо представленного примера были также определены конченые точки и написаны обработчики для следующих запросов:

* Запрос на получение всех категорий товаров, доступных в интернет-магазине;
* Запрос на получение информации о всех товарах определённой категории;
* Запрос на регистрацию;
* Запрос на вход;
* Запрос на получение данных о корзине пользователя;
* Запрос на добавление товара в корзину;
* Запрос на удаление товара из корзины;
* Запрос на добавление товара в “Избранные”;
* Запрос на удаление товара из раздела “Избранные”;
* Запрос на получение всех товаров пользователя из раздела “Избранные”;
* Запрос на добавление товара к заказу;
* Запрос на удаление товара из заказа;
* Запрос на получение подробной информации о товаре;
* Запрос на получение подробной информации о пользователе.

Кроме описанных выше обработчиков запросов были разработаны вспомогательные функции, например, для проверки введённых регистрационных данных на наличие пользователя с такими же данными в базе данных.

Также серверная часть приложения отвечает за хранение данных. Для этого была выбрана легковесная база данных SQLite. Была создана и заполнена база данных, полностью соответствующая разработанной спецификации. Для взаимодействия с ней используется библиотека SQLAlchemy. Эта библиотека использует объектно-ориентированный подход, а именно объектно-реляционное отображение для упрощения и ускорения работы с базой данных. Также, это оставляет возможности на изменение базы данных, при условии, что она есть в списке тех, с которыми поддерживает работу библиотека SQLAlchemy. Эта библиотека предоставляет средства для отображения таблиц базы данных на классы и строк базы данных на объекты соответственно.

Были определены модели таблиц созданной базы данных с помощью соответствующих классов языка Python. На рис. 3.2 представлен пример класса-модели таблицы Order. Выполнение запросов к базе данных осуществлялось с помощью методов объекта Session.

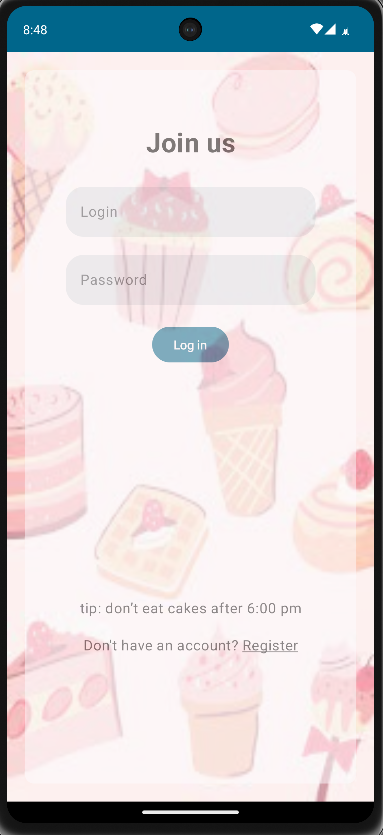


**Рисунок 3.2 Класс Order**

## **3.2 Реализация клиентской части**

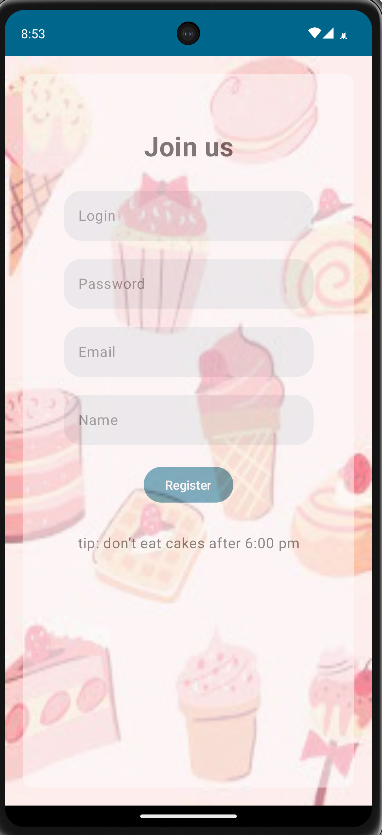
### **3.2.1 Экраны приложения**

Клиентская часть представляет собой мобильное приложение, написанное на языке Kotlin. Первое, что видит пользователь при запуске приложения - экран входа (рис. 3.3). Он содержит два поля TextField для ввода логина и пароля (в целях безопасности поле для ввода пароля не отображает вводимые символы), кнопку “Log In” для выполнения входа, а в нижней части экрана расположена кнопка “Register” для перехода на экран регистрации для незарегистрированных пользователей.



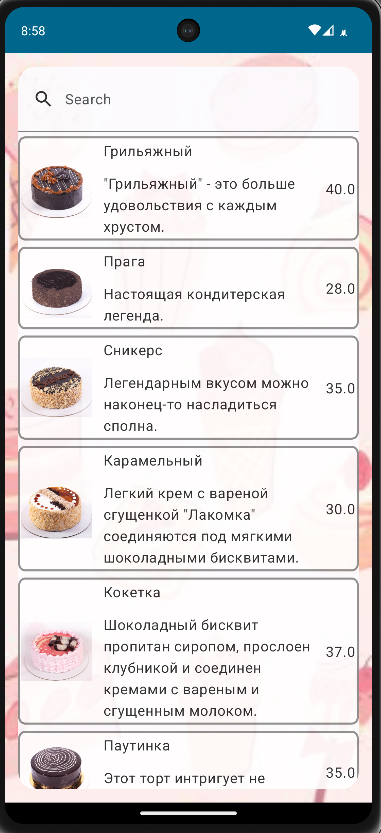
**Рисунок 3.3 Экран входа**

Экран регистрации (рис. 3.4) выглядит аналогично экрану входа, за исключением того, что содержит больше полей для ввода личных данных пользователя (имя и электронная почта). Однако эти поля не обязательны для заполнения и можно зарегистрироваться без них.



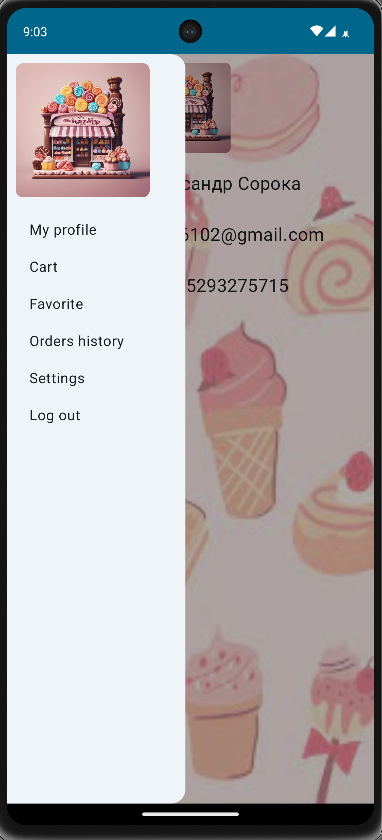
**Рисунок 3.4 Экран регистрации**

Экран каталога товаров (рис. 3.5) содержит строку поиска для поиска товаров по названию и прокручиваемый список товаров.



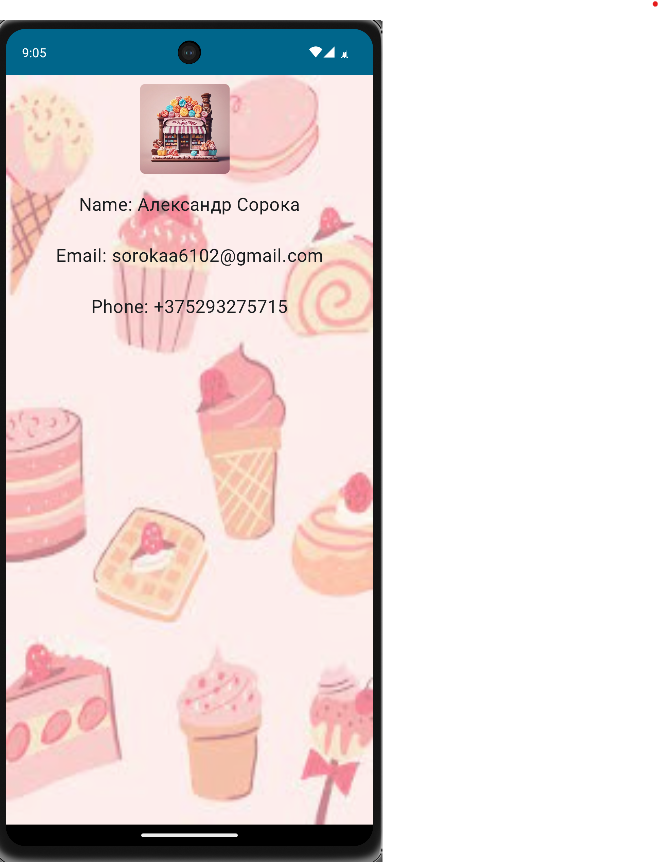
**Рисунок 3.5 Экран каталога товаров**

Боковое меню приложения (рис. 3.6) появляется с левой стороны при проведении пальцем по экрану слева направо. Оно содержит элементы для навигации между экранами приложения, а также логотип приложения.



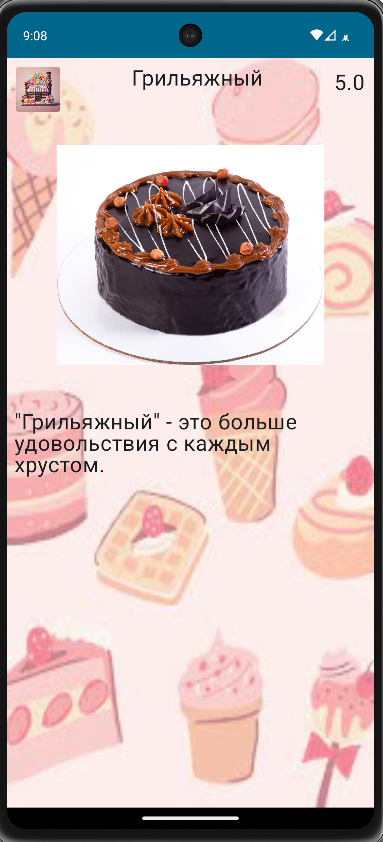
**Рисунок 3.6 Боковое меню**

Экран личного кабинета пользователя (рис. 3.7) содержит логотип приложения и информацию о пользователе (имя, электронная почта и номер телефона), которую он может при желании изменить.



**Рисунок 3.7 Личный кабинет пользователя**

Экран товара (рис. 3.8) содержит информацию о товаре, а именно: название, фотографию, описание и пользовательский рейтинг.

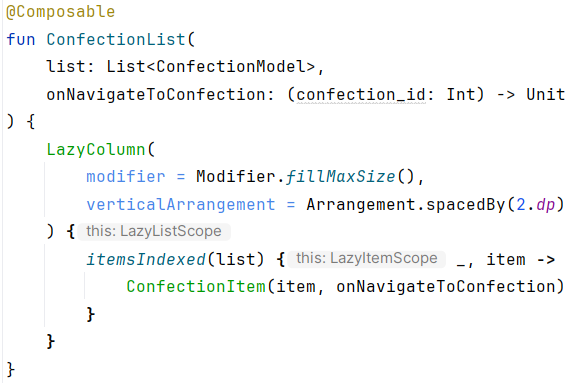


**Рисунок 3.8 Экран товара**

### **3.2.2 Реализация пользовательского интерфейса**

Пользовательский интерфейс в мобильном приложении реализован с помощью библиотеки Jetpack Compose. Jetpack Compose – это современный инструментарий для разработки пользовательского интерфейса для Android приложений на языке Kotlin. В отличие от традиционной XML-разметки библиотека Jetpack Compose использует декларативный подход, т.е. интерфейс описывается в коде. Основным элементом библиотеки являются функции, помеченный аннотацией @Composable, так называемые composable-функции. Именно такие функции отвечают за отрисовку компонентов пользовательского интерфейса, а их композиции за привлекательный дизайн приложения. С помощью атрибутов, таких как, modifier, colors, shape и т.д. можно управлять цветом, формой и поведением элементов интерфейса, которые определяются composable-функциями.

Для реализации таких элементов, как боковое меню или прокручиваемый список используются специальные функции, такие как, DrawerBody, DrawerHead и LazyColumn. На рисунке 3.9 представлен фрагмент кода, который отвечает за отрисовку списка товаров. Элемент LazyColumn реализует прокручиваемый список, элементы которого отрисовываются в соответствии с функцией ConfectionItem. В данную функцию в качестве аргумента передана функция onNavigateToConfection, которая используется для навигации к конкретному кондитерскому изделию.



**Рисунок 3.9 Функция ConfectionList**

Однако большая часть интерфейса представляет собой композицию простых composable-функций, таких как, Text, Column, Row и др. Например, на рисунке 3.10 представлен фрагмент функции UserScreen, которая отвечает за отрисовку экрана личного кабинета пользователя и состоит преимущественно из простых функций.



**Рисунок 3.10 Функция UserScreen**

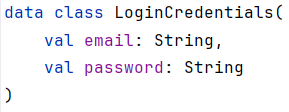
### **3.2.3 Управление навигацией в мобильном приложении**

Для управления навигацией между экранами мобильного приложения используется библиотека Navigation Compose. Основные компоненты этой библиотеки включают в себя:

* NavHost – это контейнер для отображения экранов приложения в порядке, согласно которому организована навигация в приложении.
* NavGraph – это навигационная структура приложения, которая содержит все возможные экраны и связи между ними.
* NavController – это класс, предназначенный для управления навигацией внутри контейнера NavHost. Он предоставляет методы для перехода между экранами, передачи аргументов между экранами и обратной навигации.

### **3.2.4 Взаимодействие с сервером**

Взаимодействие с сервером реализовано с помощью библиотеки Retrofit. Для работы с Retrofit были определены классы данных, которые соответствуют структуре запросов и ответов в формате JSON, необходимых для взаимодействия с сервером. На рисунке 3.11 описан класс данных LoginCredentilas для передачи регистрационных данных пользователя на сервер при выполнении запроса на аутентификацию в приложении.



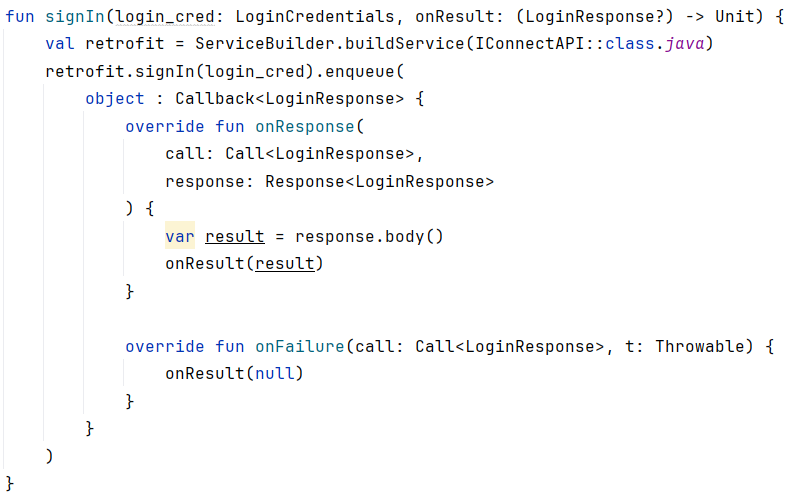
**Рисунок 3.11 Класс данных LoginCredentials**

Был определён интерфейс IConnectAPI, который описывает методы для работы с удаленным сервером с использованием Retrofit. В интерфейсе определены методы для выполнения HTTP-запросов к API. Каждый метод в интерфейсе аннотирован специальной аннотацией, которая указывает на тип HTTP-запроса и URL-адрес, на который будет отправляться запрос (например, @POST(“/api/v1/catalog”)). Кроме того, каждый метод определяет параметры запроса в виде объекта, передаваемого через аннотацию @Body, если параметр передаётся в теле запроса, или @Query, если этот параметр передаётся в строке запроса в формате URL. На рисунке 3.11 представлен запрос на регистрацию пользователя.



**Рисунок 3.12 Метод signIn интерфейса IConnectAPI**

Был определен объект ServiceBuilder, который используется для создания экземпляра Retrofit и его конфигурации определёнными параметрами и базовым URL-адресом, и функция buildService, которая создаёт и возвращает экземпляр сервиса API на основе предоставленного класса интерфейса. Класс ApiManager инкапсулирует логику взаимодействия с сервером. Он использует интерфейс IConnectAPI для создания экземпляра сервиса API и выполнения HTTP-запросов посредством вызова методов этого интерфейса. Результаты запроса обрабатываются асинхронно с помощью метода enqueue. Затем он ожидает ответа от сервера и вызывает функцию обратного вызова onResult, которая обрабатывает результат выполнения запроса в приложении. На рисунке 3.12 представлен метод signIn класса ApiManager, который реализует запрос на аутентификации в приложении.



**Рисунок 3.13 Метод signIn класса ApiManager**

**Выводы**

1. Разработан главный экран приложения.
2. Разработан экран личного кабинета пользователя.
3. Разработан экран с информацией о товаре.
4. Разработано боковое меню для навигации в приложении.
5. Создана и заполнена база данных.
6. Реализована основная часть запросов к серверу в мобильном приложении, а также обработчики этих запросов на серверной части.

# **ЗАКЛЮЧЕНИЕ**

Целью работы было проектирование и разработка мобильного приложения для интернет-магазина кондитерских изделий.

В ходе выполнения работы были рассмотрены и решены следующие задачи:

1. Изучена предметная область интернет-магазинов.
2. Были проанализированы аналогичные мобильные решения, выявлены их недостатки и достоинства.
3. Были рассмотрены библиотеки, используемые для разработки клиент-серверных приложений.
4. Разработаны функциональные и нефункциональные требования для приложения.
5. Разработаны следующие UML диаграммы: контекстная, вариантов использования и деятельности.
6. Разработана схема базы данных, используемой в приложении.
7. Разработаны прототипы экранов приложения.
8. Создана и заполнена база данных по разработанной схеме.
9. Реализован частичный функционал мобильного приложения для интернет-магазина кондитерских изделий с простым и понятным интерфейсом.

Для проектирования приложения использовалась среда Modelio Open Source 5.3 и онлайн-редактор Figma.

Для реализации рабочей версии прототипа приложения использовались среды разработки Visual Studio Code и Android Studio, языки программирования Python и Kotlin, библиотеки Flask, SQLAlchemy, Retrofit, Jetpack Compose, Coil.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Kotlin в действии. / пер. с англ. Киселев А. Н. - М.: ДМК Пресс, 2018. - 402 с.: ил.
2. Android Developers [Электронный ресурс] / Documentation for Android Operation System – Режим доступа: <http://developer.android.com/index.html>
3. Retrofit [Электронный ресурс] / Retrofit’s documentation – Режим доступа: https://square.github.io/retrofit/
4. Pallets Projects [Электронный ресурс] / Flask’s documentation – Режим доступа: https://flask.palletsprojects.com/en/3.0.x
5. QaRocks [Электронный ресурс] / Что такое клиент-серверная архитектура? – Режим доступа: <https://qarocks.ru/client-server-architecture/>
6. Unisender [Электронный ресурс] / Интернет-магазин – Режим доступа: <https://www.unisender.com/ru/glossary/chto-takoe-internet-magazin-vidy-struktura-celi/>