СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

DFD (Data flow diagrams) — диаграмма потоков данных.

HTTPS (HyperText Transfer Protocol Secure) — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности.

IDEF (Integrated Definition) — комплексное описание.

IT (Informational technology) — информационные технологии.

Авторизация — процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе. Источник: https://encyclopedia.kaspersky.ru/glossary/authorization/. Дата обращения: 20 мая 2022.

Аккаунт (account) — учетная запись, в которой хранится различная информация, относящаяся к пользователю, например, его настройки для сайта, данные о потребленных платных услугах и т.п. Источник: https://dic.academic.ru/dic.nsf/business/17471. Дата обращения: 20 мая 2022.

АС — автоматизированная система.

Аутентификация — средство защиты, устанавливающее подлинность лица, получающего доступ к автоматизированной системе, путем сопоставления сообщенного им идентификатора и предъявленного подтверждающего фактора. Источник: https://banks.academic.ru/684/Аутентификация. Дата обращения: 20 мая 2022.

База данных (БД) — совокупность хранимых в памяти компьютера данных, относящихся к определенному объему или кругу деятельности, специально организованных, обновляемых и логически связанных между собой. Источник: https://lopatnikov.pro/slovar/b/data-base/. Дата обращения: 20 мая 2022.

Библиотека — организованная совокупность программ. Обычно библиотека программ хранится во внешней памяти ЭВМ, в рамках той или иной файловой системы, обеспечивающей автоматизированный доступ к

отдельным программам. Источник: https://dic.academic.ru/dic.nsf/enc3p/72150. Дата обращения: 20 мая 2022.

Бизнес-процесс — это совокупность взаимосвязанных мероприятий или задач, направленных на создание определенного продукта или услуги для потребителей. Источник: https://dic.academic.ru/dic.nsf/ruwiki/53238. Дата обращения: 20 мая 2022.

ГОСТ — Государственный общероссийский стандарт - комплекс норм и правил. Источник: https://dic.academic.ru/dic.nsf/es/16523/ГОСТ. Дата обращения: 20 мая 2022.

Информационная система — система сбора, хранения, обработки, преобразования, передачи и обновления информации с использованием компьютерной и другой техники. Источник: https://lopatnikov.pro/slovar/i/informacionnaya-sistema/. Дата обращения: 20 мая 2022.

ИСО/ISO — “International Organization for Standardization”, «Международная Организация по Стандартизации»

Контент — тексты, графика, мультимедиа и иное информационно значимое наполнение информационной системы. Источник: https://dic.academic.ru/dic.nsf/business/19497. Дата обращения: 20 мая 2022.

Модерация — контроль выполнения требований, установленных владельцем сайта. Модерацией также называют проверку выполнения правил веб-сервисов, записанных в пользовательском соглашении. Источник: https://dic.academic.ru/dic.nsf/business/17626. Дата обращения: 20 мая 2022.

МЭК — «Международная электротехническая комиссия»

Приложение — прикладная система или программа, предназначенная для решения задач в конкретной области техники. Источник: https://technical_translator_dictionary.academic.ru/188611/приложение. Дата обращения: 20 мая 2022.

Протокол — правила, регулирующие взаимодействие различных компьютеров или компьютерных периферийных систем. Источник: https://dic.academic.ru/dic.nsf/business/10931. Дата обращения: 20 мая 2022.

СУБД — система управления базами данных

Токен авторизации — нечто, свидетельствующее о наличии полномочий, аутентичности или используется для идентификации его владельца. Источник: https://dic.academic.ru/dic.nsf/ruwiki/1650449. Дата обращения: 20 мая 2022.

Хакатон — мероприятие, во время которого специалисты из разных областей разработки программного обеспечения сообща работают над решением какой-либо проблемы Источник: https://dic.academic.ru/dic.nsf/ruwiki/1886046. Дата обращения: 20 мая 2022.

ВВЕДЕНИЕ

Цель данной курсовой работы — разработка прототипа клиент-серверного мобильного приложения для операционной системы Android, предназначенного для автоматизации заказов и повышения качества обслуживания клиентов пиццерии.

Данная тема актуальна в связи с ростом спроса на услуги доставки еды, усилением конкуренции на рынке и необходимостью предоставления пользователям удобного, интуитивного инструмента для выбора блюд, оформления заказов и отслеживания их статуса в реальном времени. Начинающие предприятия общественного питания часто сталкиваются с проблемой интеграции цифровых решений, что замедляет их адаптацию к современным требованиям клиентов.

Основными задачами являются: проектирование пользовательского интерфейса для выбора и настройки блюд, разработка серверной части для управления заказами и меню, интеграция push-уведомлений для информирования клиентов, а также обеспечение безопасности персональных данных.

Объект исследования — рынок услуг доставки готовой пищи в Российской Федерации. Предмет исследования — функциональные возможности и технологии, используемые в мобильных приложениях для автоматизации заказов. В качестве информационной базы были использованы данные популярных сервисов («Dominos Pizza», «Яндекс Еда»), отзывы пользователей и анализ функционала конкурентов.

В исследовательской части работы проведён анализ существующих аналогов, описаны бизнес-процессы взаимодействия клиента, приложения и пиццерии, а также составлены диаграммы в нотациях IDEF0 (моделирование процессов) и DFD (потоки данных). В проектной части сформулированы требования к прототипу, выбрана клиент-серверная архитектура, определены технологии разработки (Java, Retrofit, REST API). В технологической части

приведено описание реализованного прототипа, включая профиль, меню, корзину и интерфейс заказа.

1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

1.1 Описание предметной области

В предметную область данной курсовой работы входит анализ рынка мобильных приложений для заказа еды, функциональные требования пользователей к подобным сервисам, а также процессы автоматизации заказов, обработки данных (меню, корзина, оплата) и взаимодействия клиента с пиццерией. Эти данные подлежат сбору, структурированию, анализу и интеграции в разрабатываемый прототип системы для обеспечения удобства заказа, персонализации предложений и повышения эффективности обслуживания.

1.2 Анализ существующих аналогов

При проектировании информационной системы для мобильного приложения пиццерии проведен анализ функциональных возможностей ключевых аналогов, включая приложения Domino’s Pizza, «Яндекс.Еда», «Вкусно и точка» и Dodo Pizza. Исследование выявило общие и уникальные характеристики, определяющие эффективность пользовательского взаимодействия.

Приложение Domino’s Pizza демонстрирует высокую степень автоматизации процессов заказа, включая конструктор пиццы с возможностью выбора ингредиентов, интеграцию системы отслеживания статуса доставки с GPS-навигацией, а также поддержку push-уведомлений для информирования о готовности заказа. Однако ограничением является отсутствие гибких фильтров для персонализации меню, что снижает адаптивность интерфейса под индивидуальные предпочтения пользователей.

Сервис «Яндекс.Еда», функционирующий как агрегатор, обеспечивает широкий выбор ресторанов, включая пиццерии, с возможностью фильтрации по кухне, рейтингу и времени доставки. Ключевыми функциональными

преимуществами являются разнообразие методов оплаты (онлайн-платежи, наличные, СБП) и интеграция истории заказов с возможностью повторного оформления. Недостатки связаны с ограниченной кастомизацией блюд: пользователи не могут модифицировать рецептуры в рамках агрегированных предложений, что характерно для специализированных приложений пиццерий.

Приложение «Вкусно и точка» акцентировано на оптимизации скорости обработки заказов, предлагая упрощенный интерфейс с минимальным количеством шагов для оформления. Функционально выделяется системой рекомендаций на основе предыдущих заказов и интеграцией с собственными логистическими решениями, что обеспечивает прогнозируемое время доставки.

Dodo Pizza сочетает элементы пользовательского опыта, такие как интерактивный конструктор пиццы с визуализацией этапов приготовления, и систему отслеживания заказов в реальном времени с отображением данных о курьере. Особенностью является интеграция обратной связи: пользователи могут оценить качество каждого компонента заказа (тесто, начинка, доставка).

1.3 Выбор инструментальных средств моделирования

В качестве средства для моделирования была выбрана компьютерная программа “Рамус”. “Рамус” поддерживает три методологии моделирования: функциональное моделирование (IDEF0) — верхнеуровневое; описание бизнес-процессов (IDEF3) — поток работ и диаграммы потоков данных (DFD). Причиной выбора именно данной программы ее доступность и простота в использовании. Данная компьютерная программа предоставляет все необходимые инструменты для создания модели текущей и проектируемой системы во всех необходимых методологиях.

1.4 Описание процесса в нотации IDEF0

Функциональная модель состоит из поименованных процессов, функций, задач, которые должны выполняться в системе и взаимодействия этих процессов, функций, задач между собой и внешним миром.

Функциональный блок представляется в виде прямоугольника и отображает некую функцию. На каждой диаграмме должно быть от трех до шести блоков. Слева в блок поступает входящая информация, справа выходящая (результат работы функции), снизу находится механизм, необходимый для выполнения функции, сверху подаются стрелки управления, которые ограничивают процесс, управляют им.

Вход — объект, используемый и изменяемый, преобразуемый функцией для получения результата на выходе.

Выход — объект, в который преобразуются объекты со входа.

Управление — информация, управляющая и ограничивающая функцией.

Механизм — ресурсу, которые производят выполнение функции.

В данном случае имеется процесс «Использование приложения», представленный на Рисунке 1. Для данного процесса на вход в первую очередь необходимы имя пользователя и пароль для создания учетной записи и входа в нее, использования приложения. Вместе с этим дополнительно могут быть поданы электронная почта и номер телефона.

Также на вход подаются элементы меню пиццерии, содержание корзины и заказы пользователя. Эти данные будут подаваться пользователю в соответствующих панелях интерфейса.

Для пользование приложением на вход также передаются заданные пользователем настройки приложения, указываемые в его профиле. При этом настройки также контролируют работу уведомлений.

При оформлении заказа с доставкой пользователю необходимо указать адрес доставки, поэтому он также передан на вход.

Для управления и ограничения данного процесса необходим ряд документов и правил: ограничения функционала и правила пользования системой, чтобы пользователь не мог сделать что-либо лишнее, соглашение о передаче персональных данных.

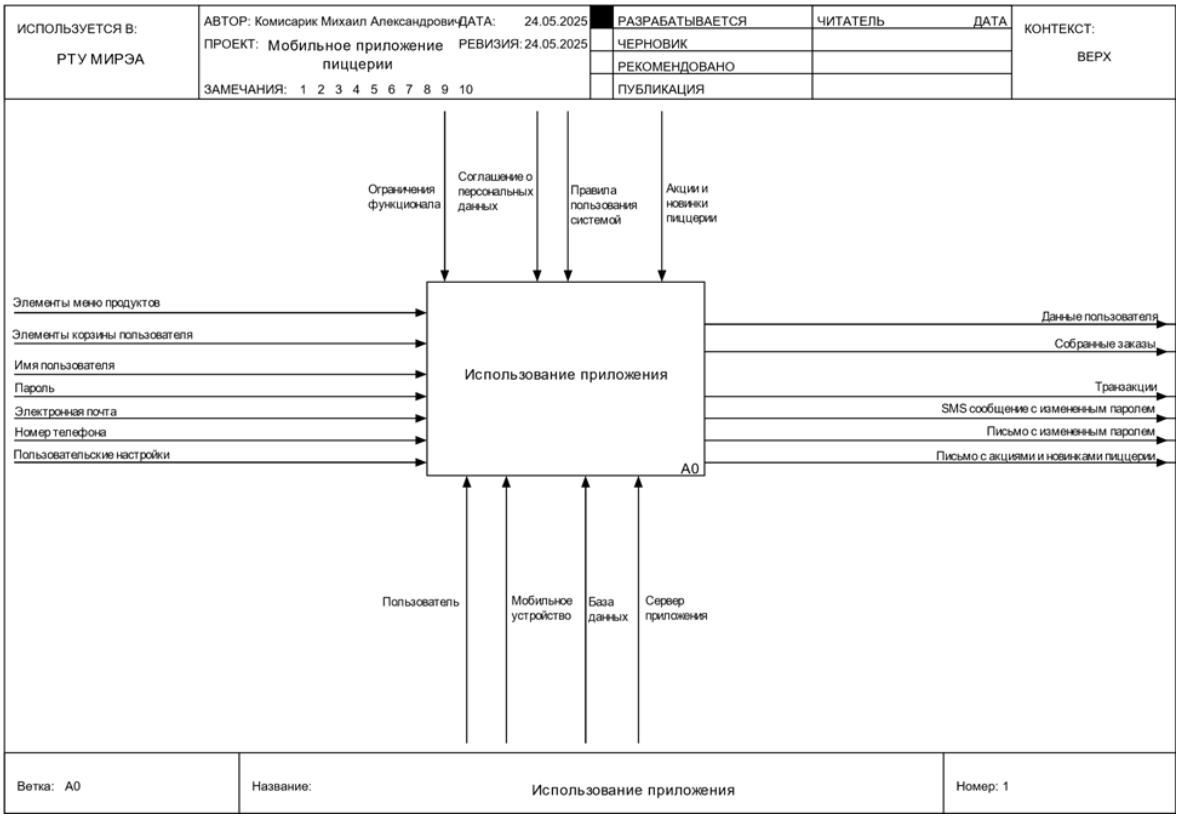

Рисунок 1 – Диаграмма A0 в нотации IDEF0

Декомпозиция процесса использования прототипа приложения представлена на Рисунке 2.

Процесс «Использование приложения», изображенный на Рисунке 1, разделяется на 5 подпроцессов: «Отображение меню блюд», «Авторизация в системе», «Взаимодействие с корзиной», «Сбор заказа» и «Отправка уведомлений». Все процессы изображены на Рисунке 2.

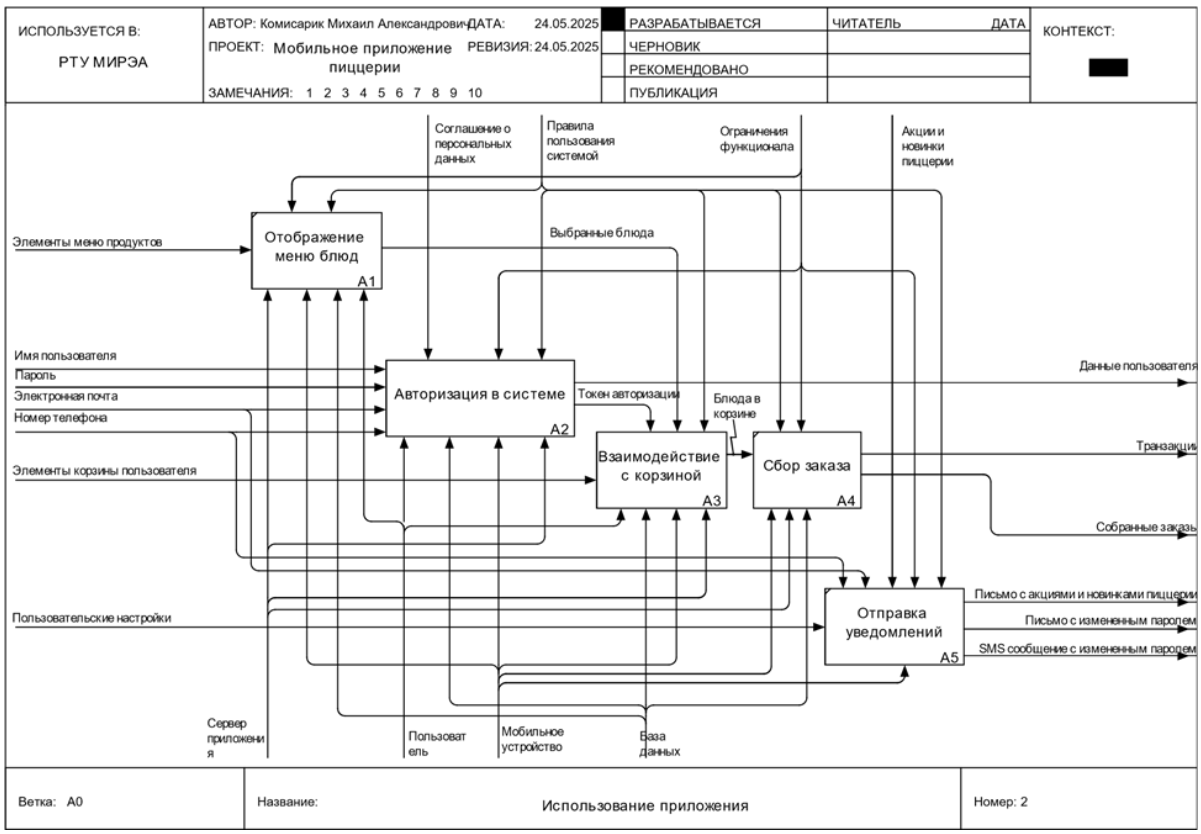

Рисунок 2 – Декомпозиция процесса «Использование приложения» в нотации IDEF0

Отображение меню продуктов осуществляется на основе элементов меню продуктов, поступающих из базы данных приложения. В самом процессе участвуют база данных, откуда читается информация и сервер приложения. На выход подаются выбранные пользователем блюда.

Для авторизации в системе (Рисунок 3) на вход необходимы электронная почта, пароль, имя пользователя, номер телефона, так как необходимо создать учетную запись и войти в нее. Электронная почта необходима только для регистрации и восстановления пароля, тогда как пароль и имя пользователя и для аутентификации. Номер телефона необходим для взаимодействия с пользователем, а также для восстановления пароля. На выход с авторизации поступает токен пользователя, который дает возможность пользователю осуществлять все действия, поэтому он поступает на управление процессами взаимодействия с корзиной и оплатой заказа. Данные пользователя подаются на выход и используются при последующей авторизации.

Подпроцесс авторизации выполняют пользователь, заполняющий данные, база данных для проверки существования профиля с такими данными и

система, мобильное устройство с установленным приложением предоставляющее все формы регистрации и аутентификации. Управляют подпроцессом соглашение о персональных данных, и ограничения функционала и правила пользования системой, которые должен изучить и принять пользователь, прежде чем зарегистрироваться.

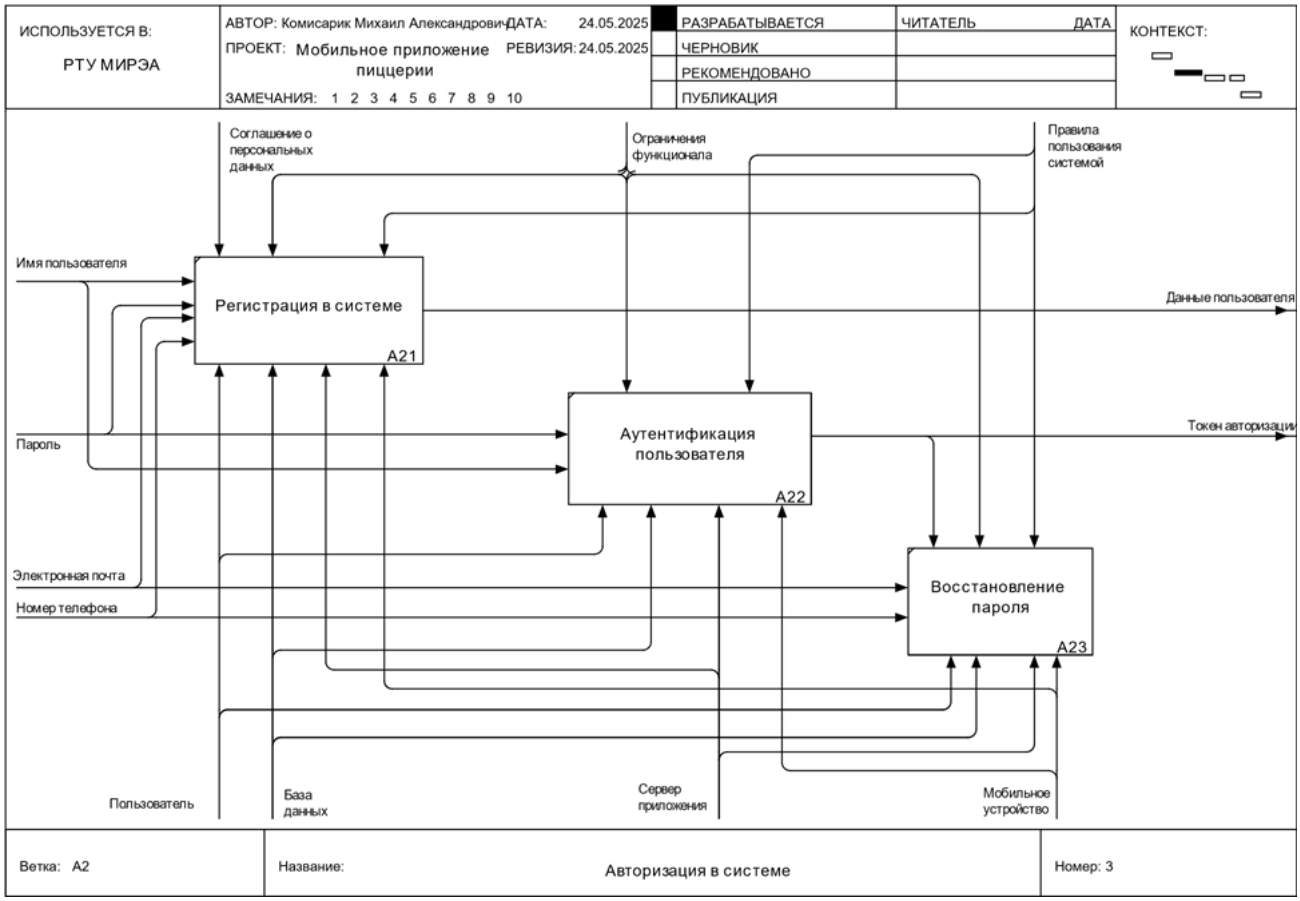

Рисунок 3 – Декомпозиция процесс «Авторизация в системе» в нотации IDEF0

Подпроцесс взаимодействия с корзиной (Рисунок 4) принимает на вход выбранные в процессе отображения меню блюда и сохраненные с прошлой сессии данные корзины а также токен авторизации, без которого невозможны никакие пользовательские операции. На выход подается список элементов корзины после взаимодействия с пользователем.

Модификацию элементов корзины производит пользователь на мобильном устройстве. Данные передаются с сервера из базы данных. Управляют данным подпроцессом правила пользования системой, ограничения функционала, токен авторизации.

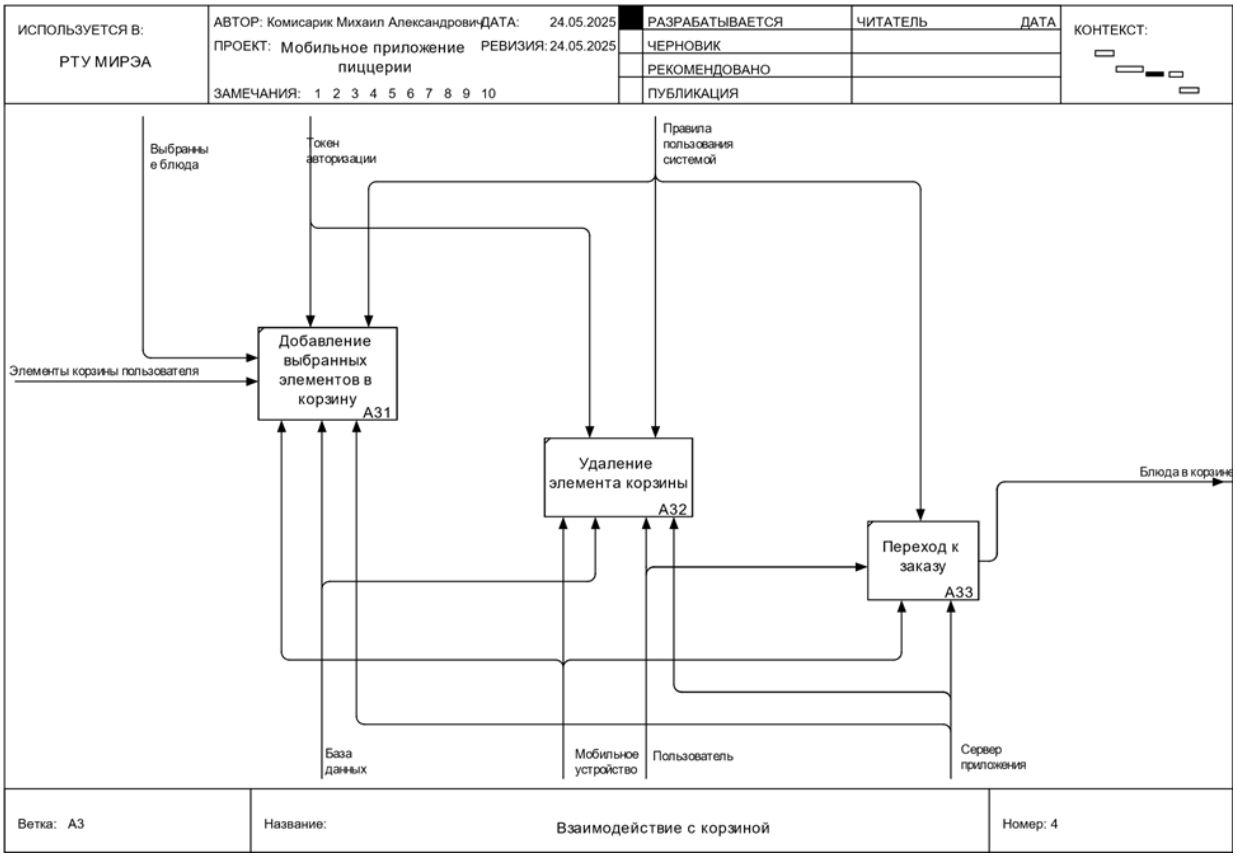

Рисунок 4 – Декомпозиция процесса «Взаимодействие с корзиной» в нотации IDEF0

Подпроцесс отправка уведомлений запускается только в случае, если это разрешено настройками, на вход идет электронная почта пользователя, если пользователь разрешил посылать туда уведомления, номер телефона, если пользователь разрешил посылать sms сообщения, и акции и новинки пиццерии, о которых требуется уведомить пользователя.

Уведомления приходят на мобильное устройство пользователя, для чего необходимо предоставить разрешение на отправку уведомлений. Управляют данным процессом ограничения функционала, правила пользования системой.

1.5 Описание процесса в нотации DFD

Диаграмма в нотации DFD отображает течение информации в пределах процесса. Для отображения входных и выходных данных используются стрелки. Процессы отображаются прямоугольниками с скругленными углами. Хранилища — прямоугольники без правой стороны, внешние объекты — прямоугольники с широкими границами.

На Рисунке 5 представлена диаграмма в нотации DFD, отображающая процесс использования приложения.

Изначально происходит получение данных меню по соответствующему запросу.

Далее необходимо рассмотреть процесс авторизации в системе. Он включает в себя как регистрацию, так и аутентификацию с восстановлением пароля. На вход подаются электронная почта, номер телефона, имя пользователя, пароль, данные сохраненного пользователя. На выход токен авторизации, электронная почта, номер телефона, пароль, сохраненный пользователь и данные авторизации.

Добавление элементов в корзину принимает выборку из ранее полученных товаров меню, которую определяет пользователь. Удаление элементов из корзины принимает выборку из элементов корзины, которую определяет пользователь, эта выборка будет удалена из корзины. На выход поступает обновленное содержимое корзины.

Для оформления заказа от пользователя поступает адрес доставки, из корзины поступает содержимое корзины, на выход идут данные об оформленном заказе.

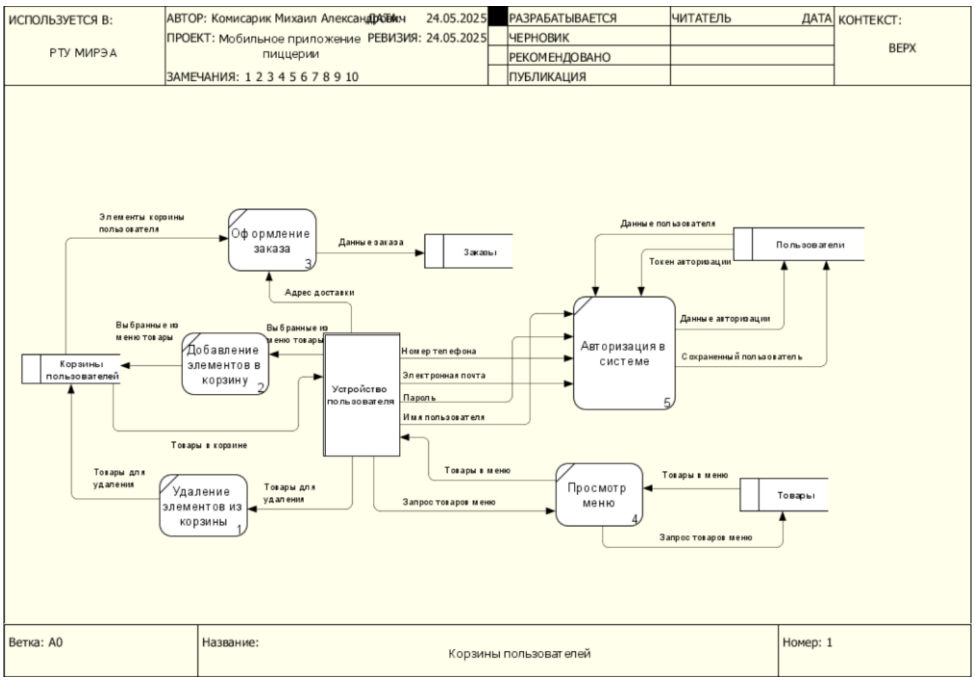

Рисунок 5 – Декомпозиция процесса «Использование приложения» в нотации DFD

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Определение пользовательских требований

Для реализации системы необходимо рассмотреть роли пользователей в разрабатываемом прототипе.

Пользователь — человек, который может просматривать контент и создавать списки.

Администратор информационной системы — создатель информационной системы, отвечающий за управление всей системы в целом.

После определения ролей были составлены пользовательские требования для каждого типа пользователя:

1) Пользователь (от 1 до ∞):
   - просматривать меню;
   - выходить из аккаунта;
   - просматривать корзину;
   - удалять элементы корзины;
   - добавлять элементы в корзину;
   - оформлять заказ из элементов корзины;
   - регистрироваться в системе;
   - авторизовываться в системе;
   - изменять настройки приложения;
2) Администратор информационной системы (от 1 до 3):
   - пополнять базу товаров меню.

2.2 Определение функциональных требований

На основе пользовательских требований были определены функциональные требования для выбранной системы.

1) Гость должен иметь возможность зарегистрироваться, заполнив форму с именем пользователя, электронной почтой, номером телефона и паролем.

2) Гость может войти в систему, указав имя пользователя и пароль.

3) При корректном вводе имени пользователя и пароля гость входит в систему, ему выводится уведомление о входе. После этого роль соответствует роли «Пользователь».

4) При некорректном вводе логина и/или пароля отображается информация об ошибке и необходимости повторного заполнения формы.

5) Гость и пользователь должны иметь возможность просматривать меню товаров.

6) Пользователь может добавлять товары в корзину.

7) Пользователь может убирать товары из корзины.

8) Пользователь может оформить заказ на товары из корзины.

9) Пользователь должен иметь возможность поменять настройки профиля.

10) Пользователь должен иметь возможность выйти из системы.

11) Пользователь должен иметь возможность изменить пароль. Для выполнения этого пользователь должен указать текущий пароль и новый пароль.

12) Пользователь должен иметь возможность производить поиск по названию элемента меню.

2.3 Описание используемых средств разработки

Для разработки были использованы две интегрированные среды разработки. Для разработки клиентской части была использована программа “Android Studio”, разработанная компанией “Google”. Для разработки серверной части использовалась “IntelliJ IDEA”, которая разрабатывается и поддерживается компанией “JetBrains”. На данный момент, “Android Studio” является основной средой разработки Android приложений, которая

поддерживается создателями операционной системы “Android”. “IntelliJ IDEA” была выбрана по той причине, что имеет похожий пользовательский интерфейс, поэтому не происходит трудозатрат на знакомство с данной средой разработки.

2.4 Архитектура программной системы

Архитектура разрабатываемой системы представляет из себя трехуровневую клиент-серверную архитектуру (Рисунок 6). В клиентском приложении располагается пользовательский интерфейс, с помощью которого происходит взаимодействие с сервером приложения, где реализована бизнес-логика. Сервер баз данных хранит данные и управляет ими.

Архитектура серверной части частично использует подход Domain-Driven-Development, который предлагает разделять разрабатываемую систему на уровень предметной области, уровень приложения и уровень инфраструктуры. Система разделена на 3 уровня, ограничение контекста с помощью создания модулей происходит при помощи инструмента контейнеризации Docker, каждый сервер хранится в своем контейнере. Так же архитектура организуется за счет того, что при разработке серверной части использовался фреймворк Spring Boot, который подразумевает деление в соответствии с тем, как указано на Рисунке 7.

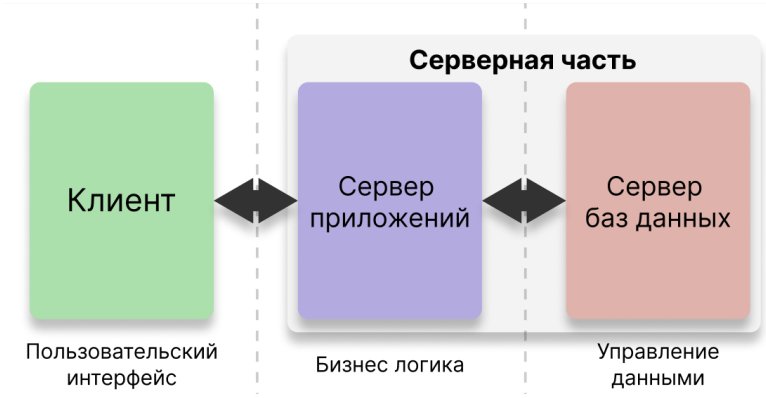

Рисунок 6 – Трехуровневая клиент-серверная архитектура

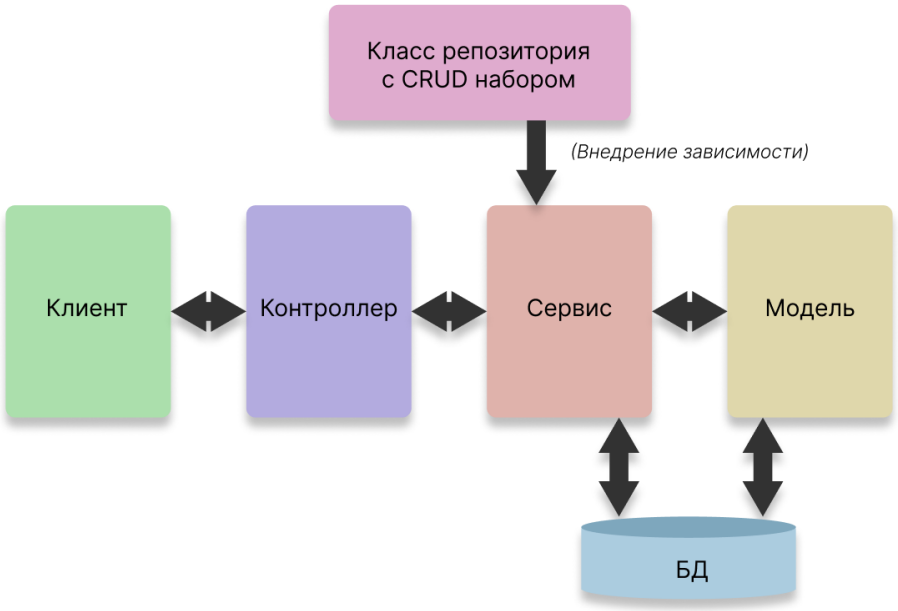

Рисунок 7 – Архитектура серверной части приложения

Что касается архитектуры клиентского приложения, то тут использовался подход «Чистой архитектуры».

Чистая архитектура направлена на разделение слоев. Уровень с бизнес-логикой (Domain), уровень данных (Data) и уровень представления (Presentation) будут разделены. На уровне представления существовал выбор между такими паттернами как MVC, MVP, MVVM, MVI. Уровень представления будет взаимодействовать с Domain-уровнем, хранящим модели данных. Выбор паттерна пал на MVVM (Рисунок 8), так как MVI еще слабо структурирован и является новым подходом, а MVC и MVP имеет ряд недостатков.

Недостатки MVC:

— сложность в Unit-тестировании;

— низкий уровень разделения.

Недостатки MVP:

— необходимость в создании дополнительного интерфейса;

— наличие повторяющегося шаблонного кода (Boilerplate code).

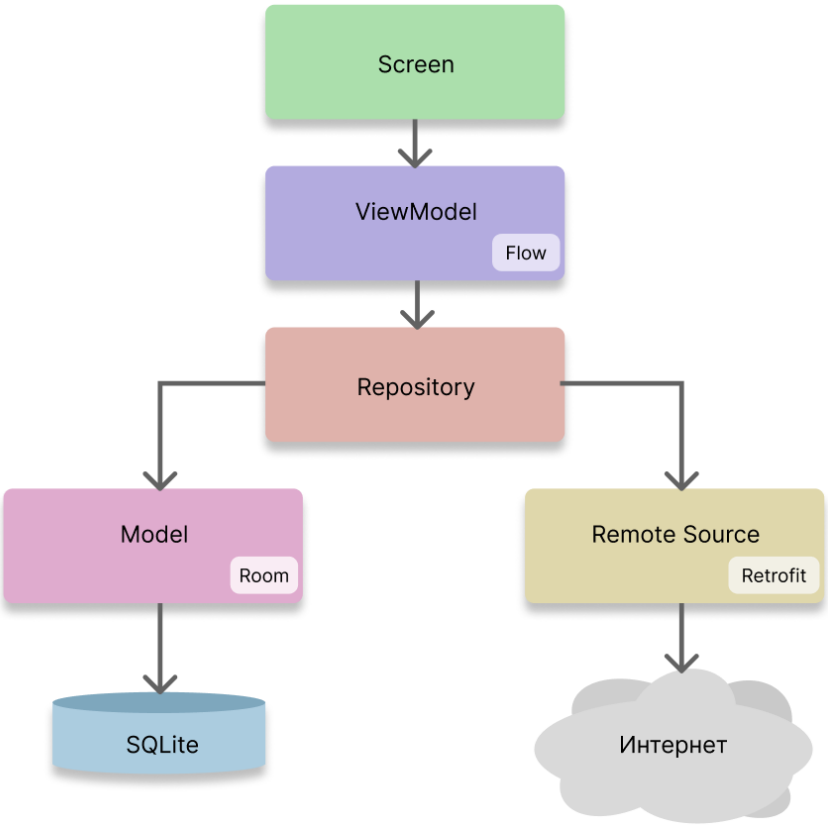

Рисунок 8 – Архитектура клиентской части приложения

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Описание моделей и структур данных

В приложении реализованы следующие модели данных: блюдо из меню (Product), избранный продукт (FavouriteProduct), пользователь (User), корзина пользователя (Cart), элемент корзины (CartItem), заказ пользователя (Order), адрес пользователя (Address), пользовательский отзыв на продукт (ProductRating), способ оплаты.

Product обладает следующими параметрами:

— уникальный номер. Тип: целочисленный;

— название. Тип: строка;

— описание. Тип: строка;

— стоимость. Тип: двойная точность;

— ссылка на картинку. Тип: строка.

FavouriteProduct обладает следующими параметрами:

— номер продукта. Тип: целочисленный;

— номер пользователя. Тип: UUID.

CartItem обладает следующими параметрами:

— номер продукта. Тип: целочисленный;

— количество продукта. Тип: целочисленный.

Cart обладает следующими параметрами:

— уникальный номер. Тип: целочисленный;

— номер пользователя. Тип: UUID;

— список продуктов. Тип: массив элементов CartItem.

User обладает следующими параметрами:

— уникальный номер. Тип: UUID;

— имя пользователя. Тип: строка;

— электронная почта. Тип: строка;

— номер телефона. Тип: строка;

— хеш пароля. Тип: строка;

— номер адреса. Тип: UUID;

— ссылка на картинку. Тип: строка.

Address обладает следующими параметрами:

— уникальный номер. Тип: UUID;

— название города. Тип: строка;

— название улицы. Тип: строка;

— номер дома. Тип: целочисленный;

— номер квартиры\оффиса. Тип: целочисленный.

Order обладает следующими параметрами:

— уникальный номер. Тип: UUID;

— способ оплаты. Тип: строка;

— цена к оплате. Тип: двойная точность;

— дата оформления. Тип: дата;

— содержимое заказа. Тип: список Product.

3.2 Описание серверной части

3.2.1 Реализация моделей данных в серверном приложении

При реализации моделей данных, которые будут храниться в базе данных была использована спецификация Java Persistence API (JPA). Она позволяет избежать написания SQL запросов, взамен этого представить сущности в виде Kotlin/Java классов, используя при этом аннотации. “@Entity” указывает на то, что данный класс является сущностью, “@Id” показывает, что поле класса является первичным ключом, с помощью “@Column” отмечаются колонки таблицы. Также JPA предоставляет простое использование отношений между таблицами. Для этого используются аннотации “@OneToOne”, “@OneToMany”,

“@ManyToMany”. Поэтому были созданы классы в соответствии с моделями, указанными ранее. Часть из созданных классов представлена в Листингах 1-2.

Листинг 1 – Класс User, хранящийся в базе данных сервера

```
@Data
@Entity
@Table(name = "users")
@AllArgsConstructor
@NoArgsConstructor
public class User
{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private UUID id;

    @Column(nullable = false, unique = true, length = 50)
    private String name;
    @Column(nullable = false, name = "password_hash")
    private String passwordHash;
    @Column(unique = true, length = 50)
    private String email;
    @Column(unique = true, name = "phone_number", length = 20)
    private String phoneNumber;
    @Column(name = "profile_image_path")
    private String profileImagePath;
}
```

Листинг 2 – Класс Product, хранящийся в базе данных сервера

```
@Data
@Entity
@Table(name = "products")
@NoArgsConstructor
public class Product
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String name;
    @Column(nullable = false)
    private String description;
    @Column(nullable = false)
    private double price;
    @Column(nullable = false, name = "image_path")
    private String imagePath;
}
```

3.2.2 Реализация хранения данных в серверном приложении

Хранение данных в Spring осуществляется с помощью интерфейса класса репозитория, наследуемого от JpaRepository, который предоставляет функции создания, чтения, обновления, удаления. Также интерфейс можно расширять с помощью написания абстрактных функций, которые будет реализовывать фреймворк. Примеры репозиториев представлен в Листинге 3.

Листинг 3 – Репозиторий, предоставляющий доступ к сущности User в базе данных

```
public interface UserRepository extends JpaRepository<User, Long>
{
    Optional<User> findByName(String name);
    boolean existsByName(String name);

    Optional<User> findByEmail(String email);
    boolean existsByEmail(String email);

    Optional<User> findByPhoneNumber(String phoneNumber);
    boolean existsByPhoneNumber(String phoneNumber);
}
```

3.2.3 Реализация контроллеров для взаимодействия с клиентом

Spring Boot также предоставляет способ описания классов, которые будут обрабатывать входящие REST API запросы. Для этого класс помечают аннотацией “@Controller”. Также существуют отдельные аннотации для методов, чтобы они обрабатывали конкретный тип запроса (POST, GET, PUT, DELETE). Пример обработчика запросов, который позволяет создавать, удалять, обновлять и читать данные типа User представлен в Листинге 4.

Листинг 4 – Класс-обработчик HTTP запросов, предоставляющий доступ к User

```
@RestController
@RequestMapping("/api/users")
public class UserController
{
    private final UserService userService;

    public UserController(UserService userService)
    {
        this.userService = userService;
    }

    @GetMapping
    public ResponseEntity<List<User>> getAllUsers()
    {
        return ResponseEntity.ok(userService.getAllUsers());
    }

    @GetMapping("/me")
    public ResponseEntity<UserResponseDto> getUserByToken()
    {
        Long id =
Long.valueOf(SecurityContextHolder.getContext().getAuthentication().getName());
        User user = userService.findUserById(id);
        if (user == null)
        {
            throw new UserNotFoundException("User not found");
        }
        UserResponseDto response = UserService.convertToResponseDto(user);
        return ResponseEntity.ok(response);
    }

    @PostMapping("/me")
```

```
    public ResponseEntity<UserResponseDto> saveUserByToken(@RequestBody
UserRequestDto newUser)
    {
        Long id =
Long.valueOf(SecurityContextHolder.getContext().getAuthentication().getName());
        User user = userService.findUserById(id);
        checkUserRequestValid(newUser.getPassword(), user);
        user.setName(newUser.getName());
        user.setEmail(newUser.getEmail());
        user.setPhoneNumber(newUser.getPhoneNumber());
        return
ResponseEntity.ok(UserService.convertToResponseDto(userService.saveUser(user)));
    }

    @DeleteMapping("/me")
    public ResponseEntity<Void> deleteUserByToken()
    {
        Long id =
Long.valueOf(SecurityContextHolder.getContext().getAuthentication().getName());
        userService.deleteUserById(id);
        return ResponseEntity.ok().build();
    }

    @PostMapping("/me/password")
    public ResponseEntity<Void> confirmPasswordByToken(@RequestBody PasswordDto
request)
    {
        Long id =
Long.valueOf(SecurityContextHolder.getContext().getAuthentication().getName());
        User user = userService.findUserById(id);
        checkUserRequestValid(request.getPassword(), user);
        return ResponseEntity.ok().build();
    }

    private static void checkUserRequestValid(String password, User user)
    {
        if (user == null)
        {
            throw new UserNotFoundException("User not found");
        }
        if (!AuthService.matchPassword(password, user.getPasswordHash()))
        {
            throw new InvalidCredentialsException("Invalid password");
        }
    }
}
```

3.3 Описание клиентской части

3.3.1 Модели и структуры данных клиентской части

При разработке мобильного приложения хранить большое количество данных на устройстве пользователя не требуется, поэтому все хранение данных обошлось использованием встроенной библиотеки Android SharedPreferences. Библиотека позволяет удобно хранить пары ключ-значение, записывать и

удалять их в любой момент. Для хранения данных с сервера используются аналогичные типы данных, такие как UserResponseDto или MenuItem (Листинги 5-6).

Листинг 5 – Класс UserResponseDto, хранящийся в оперативной памяти клиента

```
public class UserResponseDto
{
    private final String name;
    private final String email;
    private final String phoneNumber;

    public UserResponseDto(String name, String email, String phoneNumber)
    {
        this.name = name;
        this.email = email;
        this.phoneNumber = phoneNumber;
    }

    public String getName()
    {
        return name;
    }

    public String getEmail()
    {
        return email;
    }

    public String getPhoneNumber()
    {
        return phoneNumber;
    }
}
```

Листинг 6 – Класс MenuItem, хранящийся в оперативной памяти клиента

```
public class MenuItem
{
    private final String name;
    private final String description;
    private final double price;
    private final String imagePath;

    public MenuItem(String name, String description, double price, String
imagePath)
    {
        this.name = name;
        this.description = description;
        this.price = price;
        this.imagePath = imagePath;
    }

    public String getName()
    {
        return name;
    }

    public String getDescription()
    {
        return description;
    }
```

```
    public String getPrice()
    {
        return "от " + price + " руб.";
    }

    public String getImageUrl()
    {
        return RetrofitClient.BASE_URL + "/api/public/images/" + imagePath;
    }
}
```

3.3.2 Классы взаимодействия с данными на сервере

Для взаимодействия с сервером и отправки REST запросов была использована библиотека Retrofit, которая позволяет отправлять GET, POST, PUT, DELETE запросы. Для сериализации данных в JSON и десериализации из него использована библиотека GSON.

Так как пользователю необходимо авторизироваться для использования приложения, а также отправлять JWT токен с каждым запросом, то при получении токена, он сохраняется в SharedPreferences приложения в зашифрованном виде и передается с каждым запросом с помощью аннотации @Header("Authorization"). Запросы с авторизацией делаются из класса ProfileViewModel.

3.3.3 Классы предоставления данных

Классы предоставления данных, располагаемые в пакете presentation представляют из себя главную Activity, которая отображает корневой экран с навигацией. В зависимости от того, хранится ли токен в локальной базе данных или нет, экран будет отображать страницу входа или домашнюю страницу приложения.

Для каждого экрана был создан класс, наследуемый от ViewModel, который хранит данные экрана, чтобы данные не удалялись при его обновлении. ViewModel реализует также взаимодействие пользователя с данными. Она запрашивает данные из репозитория, осуществляет удаление, получение, обновление данных, как в локальном, так и удаленном хранилище.

Пример части класса ViewModel, который собирает данные пользователя и сохраняет их в собственные поля, представлен в Листинге 7.

Листинг 7 – Класс ProfileViewModel, хранящий данные об авторизации пользователя

```
public class ProfileViewModel extends ViewModel
{
    private final MutableLiveData<AuthState> authState = new
MutableLiveData<AuthState>(AuthState.REFRESHING);
    private final SecureStorageHelper storageHelper;
    private final ApiService api;

    private final MutableLiveData<String> username = new MutableLiveData<>();
    private final MutableLiveData<String> email = new MutableLiveData<>();
    private final MutableLiveData<String> phoneNumber = new MutableLiveData<>();

    public ProfileViewModel(SecureStorageHelper storageHelper, ApiService api)
    {
        this.storageHelper = storageHelper;
        this.api = api;
        checkInitialAuthState();
    }

    private void checkInitialAuthState()
    {
        new Thread(() ->
        {
            try
            {
                authState.postValue(AuthState.REFRESHING);
                Gson gson = new Gson();
                TokenPair tokens =
gson.fromJson(storageHelper.getDecryptedData(), TokenPair.class);
                String refreshToken = tokens.getRefreshToken();
                String accessToken = tokens.getAccessToken();
                if (refreshToken == null)
                {
                    authState.postValue(AuthState.UNAUTHENTICATED);
                    return;
                }
                if (accessToken != null)
                {
```

3.4 Тестирование и верификация прототипа

Для тестирования приложения была вручную проверена работа основного функционала:

— Отображение меню (Рисунок 9),

— Фильтрация меню по запросу (Рисунок 10),

— Регистрация пользователя (Рисунок 13),

— Авторизация пользователя (Рисунки 11-12),

— Выход из аккаунта (Рисунок 11),

— Сохранение пользовательских данных (Рисунки 14-15),

— Удаление аккаунта (Рисунки 16-17),

— Просмотр корзины и подсчитанных данных к оплате (Рисунки 19-20),

— Добавление продуктов в корзину из меню (Рисунок 18),

— Изменение числа каждого элемента корзины (Рисунок 21),

— Удаление элементов корзины (Рисунок 22).

Рисунок 9 – Панель меню приложения

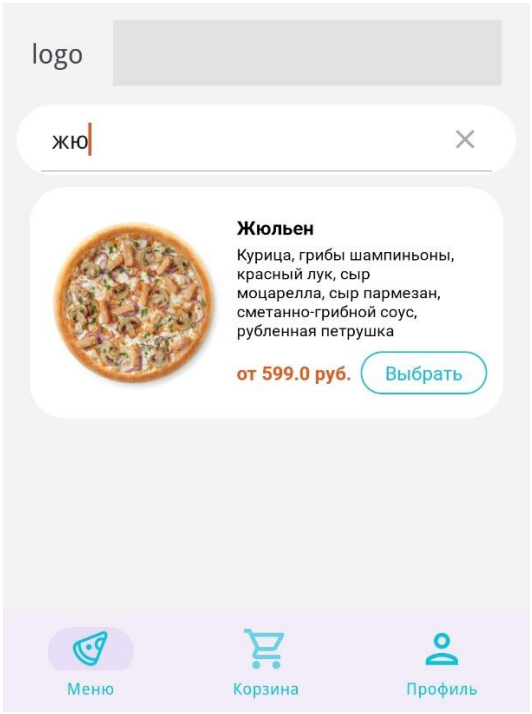

Рисунок 10 – Фильтрация меню по запросу

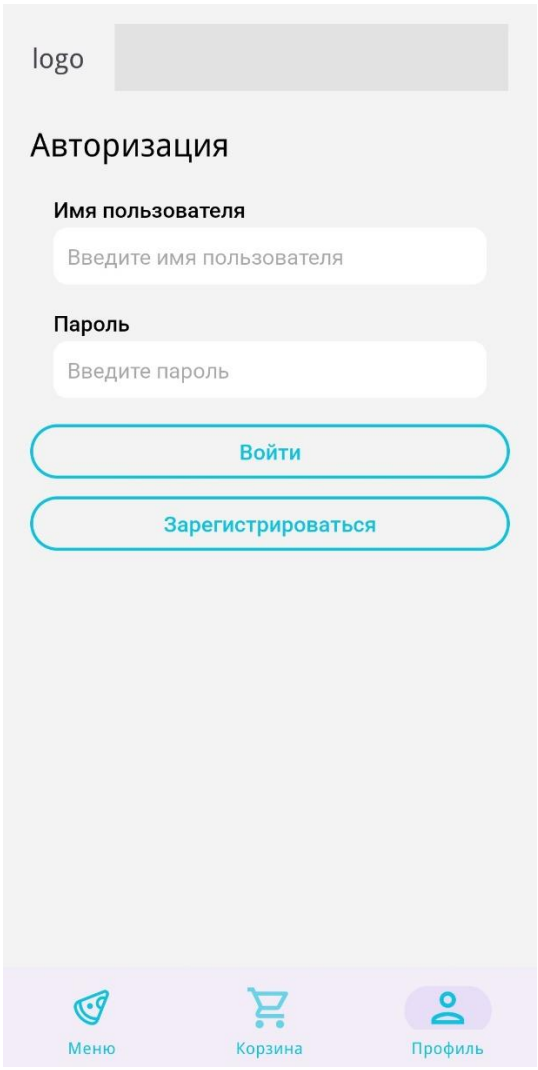

Рисунок 11 – Панель авторизации гостя

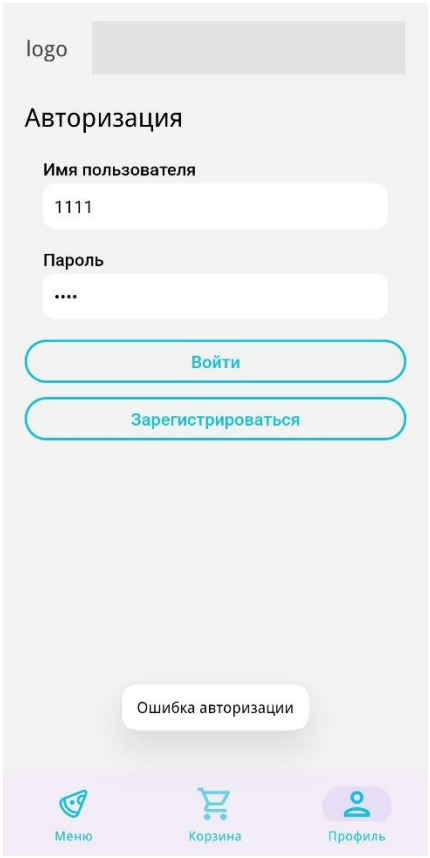

Рисунок 12 – Вход с ошибочными данными

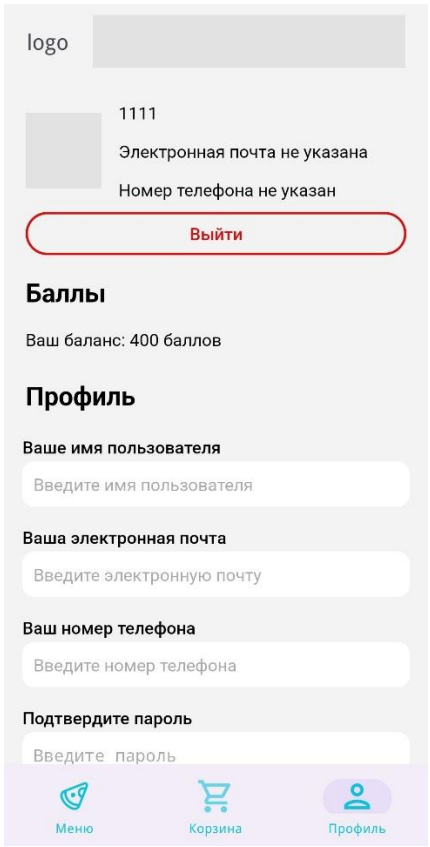

Рисунок 13 – Успешная регистрация пользователя

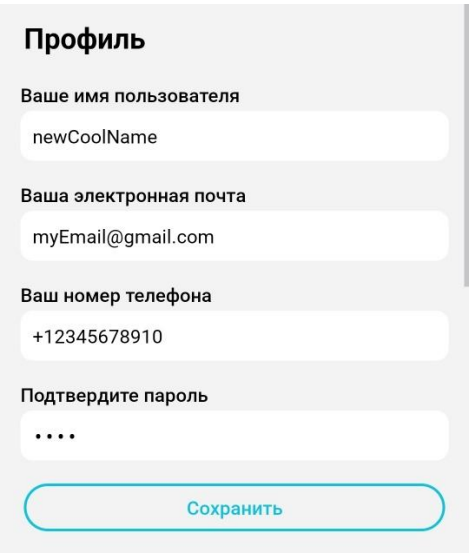

Рисунок 14 – Введенные пользовательские данные

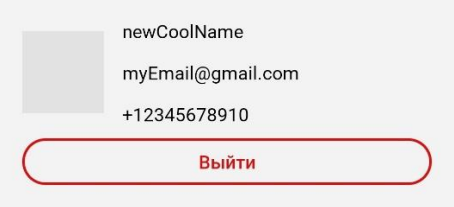

Рисунок 15 – Данные успешно заданы

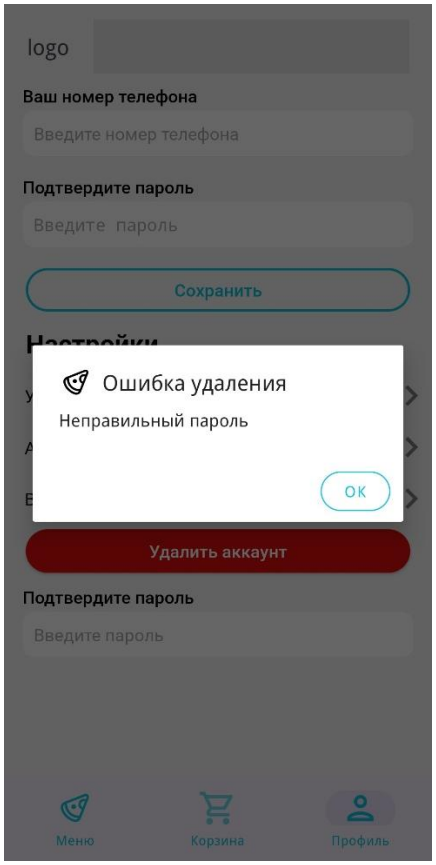

Рисунок 16 – Попытка удаления аккаунта с неверно введенным паролем

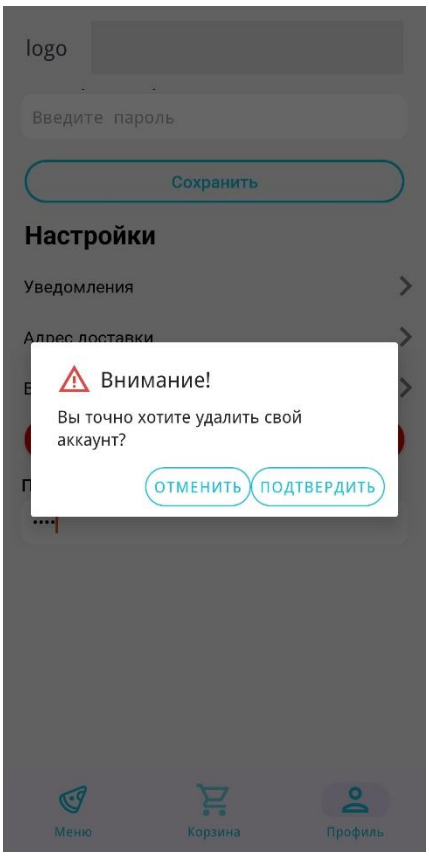

Рисунок 17 – Диалоговое окно подтверждения при попытке удаления аккаунта

Рисунок 18 – Добавление продуктов в корзину из меню

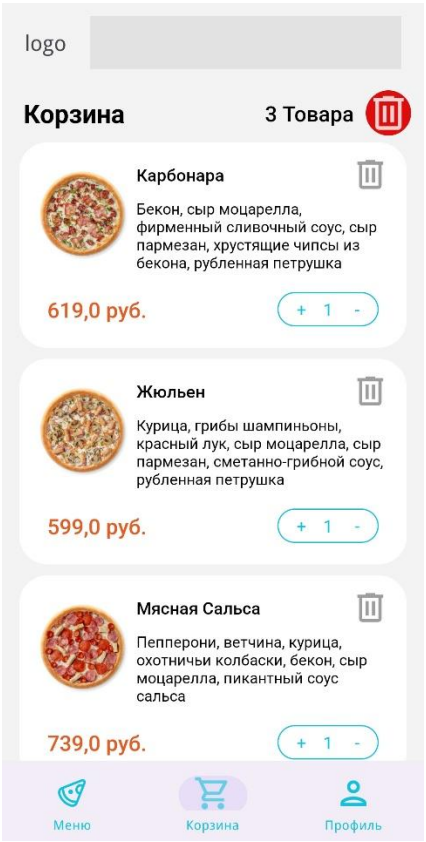

Рисунок 19 – Корзина пользователя, часть 1

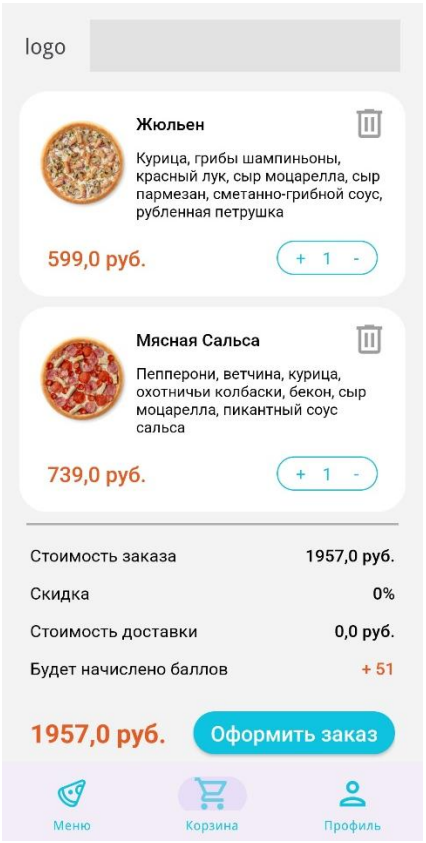

Рисунок 20 – Корзина пользователя, часть 2

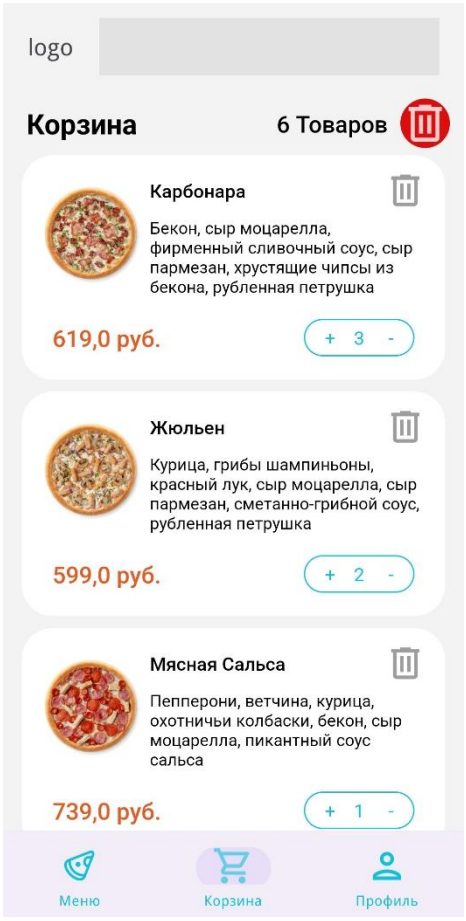

Рисунок 21 – Изменение количеств элементов корзины

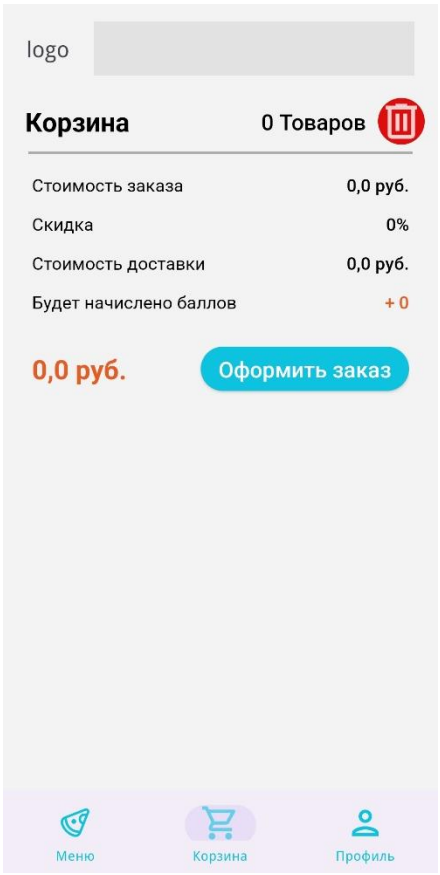

Рисунок 22 – Удаление элементов из корзины

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы было разработано мобильное приложение пиццерии, предназначенное для автоматизации заказов и повышения качества обслуживания ее клиентов. Были сформированы пользовательские и функциональные требования, рассмотрены модели в нотациях IDEF0, DFD, выявлены и обоснованы цели создания системы, ее назначение. Изучены аналоги данной системы, их проблемы и особенности. В результате был создан прототип системы, произведено его тестирование и верификация.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Остроух, А.В. Интеллектуальные информационные системы и технологии: монография / А.В. Остроух, А.Б. Николаев. — СПб: Лань, 2019. — 308 с.

2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие / Т.М. Зубкова. — СПб: Лань, 2019. — 324 с.

3. Миндалев И.В. Моделирование бизнес-процессов с помощью IDEF0, DFD, BPMN за 7 дней: учеб. пособие / И.В. Миндалев; Краснояр. гос. аграр. ун-т. — Красноярск.: Изд-во Краснояр. гос. аграр. ун-т, 2018. — 123 с.

4. Рочев К. В. Информационные технологии. Анализ и проектирование информационных систем: учебное пособие / Рочев К. В. — СПб.: Лань, 2019. — 128 с.

5. Neil Smyth. Android Studio 4.1 Development Essentials – Kotlin Edition / Neil Smyth – Cary (North Carolina (US)), Payload Media, Inc., 2020 – 1042 с.

6. Retrofit [Электронный ресурс] – 2022 – Режим доступа: https://square.github.io/retrofit/, свободный. Язык: Английский. Дата обращения: 28.05.2022.

7. Documentation for app developers [Электронный ресурс] – 2022 – Режим доступа: https://developer.android.com/docs, свободный. Язык: Английский. Дата обращения: 28.05.2022.

8. Руководство по языку Kotlin [Электронный ресурс] – 2022 – Режим доступа: https://kotlinlang.ru/docs, свободный. Дата обращения: 28.05.2022.

ПРИЛОЖЕНИЕ

Репозиторий разрабатываемого прототипа мобильного приложения находится по адресу: https://github.com/Krezerenko/pizzeria-app

Репозиторий разрабатываемого прототипа серверной части приложения находится по адресу: https://github.com/Krezerenko/trpp_database