Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

Курсовая работа

По Разработка приложений для мобильных операционных систем

Тема АИС «Доставка еды из ресторана»

Руководитель

Колпаков А.А.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121
(группа)

Медведева М.Г
(фамилия, инициалы)

(подпись) (дата)

Муром 2024

Данная курсовая работа посвящена разработке Android-приложения доставки еды из ресторана. Целью работы является автоматизация основных процессов управления данными клиентов, оформления заказов для доставки. В качестве средств разработки приложения была использована среда Android Studio. Язык разработки: Kotlin.

This course work is devoted to the development of an Android application for food delivery from a restaurant. The purpose of the work is to automate the main processes of managing customer data and placing orders for delivery. The Android Studio environment was used as the application development tools. Development language: Kotlin.

Содержание

Введение

Развитие технологий и широкое распространение мобильных устройств значительно изменили повседневную жизнь, включая способы заказа товаров и услуг. Одной из таких сфер является индустрия общественного питания, где доставка еды стала не только удобным, но и востребованным сервисом. В ответ на этот запрос появляется необходимость в разработке специализированных приложений, которые обеспечивают удобство, скорость и безопасность взаимодействия пользователей с ресторанным бизнесом.

Целью данной курсовой работы является разработка мобильного приложения для доставки еды из ресторана. Приложение будет предоставлять пользователю возможность выбирать блюда, оформлять заказы, отслеживать их статус, а также управлять своей историей заказов через личный кабинет. Для администраторов предусмотрены функции добавления, редактирования и удаления данных о блюдах, а также управления заказами и пользователями через административный интерфейс. Ключевым элементом системы будет база данных PG, обеспечивающая хранение информации о заказах, пользователях и блюдах.

Особенности реализации включают использование операционной системы Android и языка Kotlin, что позволит обеспечить совместимость приложения с большинством современных мобильных устройств. Также большое внимание будет уделено тестированию приложения, включая модульное тестирование компонентов, функциональное тестирование на виртуальных устройствах и автоматизированное тестирование интерфейса.

Проект представляет собой не только практическую задачу, но и важный шаг в изучении процессов разработки мобильных приложений, проектирования баз данных, работы с современными фреймворками и методами тестирования программного обеспечения.

Основные задачи, которые предстоит решить в ходе разработки: создание интерфейса приложения, проектирование базы данных, интеграция с серверной

частью, обеспечение безопасности данных и корректная работа системы на различных устройствах.

Этот проект может стать основой для дальнейших улучшений и расширений, таких как интеграция с различными системами оплаты, реализация рекомендаций на основе предпочтений пользователей и другие функциональные улучшения, способствующие улучшению качества сервиса в сфере доставки еды.

1 Анализ технического задания

В данной курсовой работе предстоит разработать автоматизированную информационную систему для доставки еды из ресторана. Приложение будет использоваться для оформления заказов пользователями, их отслеживания, а также для управления данными о заказах, блюдах. Система должна включать функционал для клиентов и администраторов, а также обеспечивать удобный и эффективный интерфейс.

Значит при реализации программного обеспечения должны быть выполнены следующие задачи:

–оформление заказа (приложение должно позволять пользователю выбирать блюда, оформлять заказ и отслеживать его статус в реальном времени. Пользователь будет видеть процесс выполнения заказа, что обеспечит ему удобство и уверенность в своевременной доставке)

–личный кабинет пользователя (в личном кабинете клиента должна быть возможность просматривать историю заказов, изменять личные данные и отслеживать текущие заказы)

–кабинет администратора (администраторы должны иметь возможность управлять базой данных блюд, редактировать и добавлять новые позиции, а также контролировать заказы. Важным аспектом является использование SQL для работы с базой данных)

–просмотр заказов (необходимо предусмотреть возможность просмотра подробной информации о заказах, включая текущее состояние, стоимость)

–документирование кода

–система контроля версий (Все исходные коды должны быть размещены на платформе GitHub, что позволит отслеживать изменения, а также обеспечить командную работу и возможность отката к предыдущим версиям кода)

–тестирование (для обеспечения стабильности работы приложения необходимо выполнить модульное тестирование, а также провести функциональное тестирование на нескольких виртуальных устройствах для

проверки совместимости приложения с различными конфигурациями. Автоматизированное тестирование интерфейса также необходимо для проверки его удобства и корректной работы)

В результате анализа технического задания были выделены технические требования:

–операционная система Android, язык Kotlin: приложение разрабатывается для Android, что обусловливает использование Kotlin как основного языка программирования. Kotlin обеспечит высокую производительность и удобство в разработке.

–база данных: для хранения информации о заказах, пользователях и блюдах будет использоваться база данных SQL. Реализация CRUD-операций (создание, чтение, обновление, удаление) будет важной частью работы с данными.

–использование Node.js: для серверной части приложения будет использован Node.js, что обеспечит эффективное взаимодействие с базой данных и сервером. Node.js также хорош для реализации асинхронных операций, таких как обработка заказов и взаимодействие с клиентами в реальном времени.

Для реализации данной курсовой работы необходимо учесть все этапы реализации:

–анализ области применения (важно понять, как приложение будет использоваться конечными пользователями, а также какие потребности бизнеса оно будет удовлетворять)

–проектирование базы данных (на этом этапе необходимо спроектировать структуру базы данных, которая будет эффективно хранить информацию о клиентах, заказах, блюдах и администраторах)

–разработка интерфейса (интерфейс должен быть интуитивно понятным и удобным как для пользователей, так и для администраторов)

–реализация функционала (реализация всех требований ТЗ, включая создание и управление заказами, а также обработку данных пользователей и администраторов)

–тестирование и отладка (приложение должно пройти модульное, функциональное и автоматизированное тестирование, чтобы убедиться в его корректной работе)

Возможные области применения системы:

–управление заказами и доставкой: система позволит улучшить процессы оформления заказов и их отслеживания, что повысит удовлетворенность клиентов и улучшит качество обслуживания.

–управление базой данных блюд: администраторы смогут легко управлять данными о блюдах, добавлять новые позиции и отслеживать информацию о заказах.

–анализ выручки: возможность анализа данных позволит ресторанам лучше понимать, какие блюда пользуются спросом, а какие требуют дополнительного продвижения.

–снижение затрат: автоматизация процессов и упрощение взаимодействия с клиентами снизит затраты на обслуживание и повысит эффективность работы ресторана.

Курсовая работа по теме АИС «Доставка еды из ресторана» представляет собой сложную и многофункциональную систему, требующую тщательного проектирования и тестирования. Приложение будет включать как клиентский, так и административный функционал, обеспечивая удобство пользователей и эффективное управление бизнесом для администраторов. Важно уделить внимание всем аспектам разработки, включая проектирование базы данных, создание интерфейса и обеспечение безопасности данных.

2 Разработка алгоритмов

2.1 Описание предметной области

Предметная область курсовой работы охватывает систему автоматизации заказа еды из ресторана.

Основными элементами системы являются:

–клиенты: люди, которые делают заказы через приложение.

–меню: список блюд, доступных для заказа, с их описаниями и ценами.

–заказы: заявки клиентов на заказ определённых блюд.

–позиции заказа: конкретные блюда, входящие в состав заказа, с указанием их количества и стоимости.

–статусы заказов: статусы, которые показывают текущее состояние заказа.

–пользователи: это администраторы и клиенты системы. Пользователи могут иметь разные роли (клиент или администратор).

–основные бизнес-процессы: клиенты делают заказы, выбирая блюда из меню.

Каждый заказ имеет свой статус, который меняется в процессе выполнения.

Заказ может содержать несколько позиций (блюд), и для каждой позиции фиксируется количество и цена.

Администраторы системы управляют данными заказов и меню.

Клиенты могут просматривать историю своих заказов, а администраторы управлять ими.

2.2 Разработка базы данных

Для хранения данных о пользователях, заказах и позициях будет использоваться база данных PostgreSQL. Структура базы данных будет включать следующие таблицы:

–users: хранит данные о пользователях (логин, пароль).

–customers: хранит информацию о клиентах (имя, телефон, email).

–menu: содержит информацию о блюдах (название, цена, описание).

–orders: хранит заказы клиентов (связь с клиентом, статус, дата, общая цена).

–order_items: хранит детали каждого заказа (связь с блюдами, количество, цена).

–status: хранит возможные статусы заказов (например, "Принят", "Готовится", "Доставляется").

Пример таблицы для клиентов:

```
CREATE TABLE customers (
    idclient SERIAL PRIMARY KEY,
    phone_num VARCHAR(15),
    mail VARCHAR(255),
    fullname VARCHAR(255) NOT NULL
);
```

2.3 Физическая модель данных

Физическая модель данных – это конкретное описание, как информация будет храниться и организована в физической базе данных. Она предоставляет конкретную реализацию сущностей на уровне конкретной базы данных. Она детализирует, как объекты логической модели преобразуются в таблицы, поля и отношения в реальной базе данных.

На физической модели данных, представленной на рисунке 1 приложения 1, отображены все сущности со своими атрибутами и ключами, а также детализированы взаимосвязи между сущностями в контексте конкретной базы данных. Центральное положение занимает сущность "Заказ", поскольку она является ключевым элементом, связывающимся со всеми остальными сущностями в базе данных.

2.4 Клиент-серверное взаимодействие

Приложение использует клиент-серверную архитектуру, в которой мобильное приложение (клиент) взаимодействует с сервером, реализованным на Node.js, для выполнения различных операций, таких как регистрация пользователей, оформление заказов и работа с данными меню. Взаимодействие между клиентом и сервером осуществляется через REST API.

Основные компоненты системы:

–клиент (мобильное приложение): Android-приложение, написанное на языке Kotlin, которое взаимодействует с сервером для выполнения операций, таких как оформление заказов, регистрация пользователя, получение информации о меню.

–сервер: серверная часть, реализованная с использованием Node.js, обрабатывает HTTP-запросы от клиента и выполняет соответствующие операции с базой данных (например, создание заказа, добавление и удаление позиций из меню, обновление статусов заказов).

–база данных: сервер взаимодействует с PostgreSQL, где хранятся все данные о пользователях, заказах, позициях меню и их статусах.

2.5 Разработка сервера приложения

Мобильное приложение отправляет HTTP-запросы (GET, POST, PUT, DELETE) на сервер, где происходит обработка запросов и возврат данных в формате JSON. Для обмена данными между клиентом и сервером используются следующие HTTP-методы:

–GET: Используется для получения данных (например, списка меню, истории заказов, статусов заказов).

–POST: Применяется для отправки данных на сервер (например, регистрация пользователя, создание нового заказа).

–PUT: Используется для обновления данных (например, изменение статуса заказа).

–DELETE: Применяется для удаления данных (например, удаление позиции из заказа).

Пример запроса для добавления блюда в меню:

```
router.post('/', async (req, res) => {
   const { namedish, price, description } = req.body;
   try {
      const result = await pool.query(
         'INSERT INTO menu (namedish, price, description) VALUES ($1, $2, $3) RETURNING *',
         [namedish, price, description]
      );
      res.status(201).json(result.rows[0]);
   } catch (err) {
      console.error(err.message);
      res.status(500).send('Ошибка сервера');
   }
});
```

2.6 Разработка клиентской части приложения

Мобильное приложение на Android отправляет данные через Retrofit или другие HTTP-клиенты и получает ответ от сервера. Ответ в формате JSON обрабатывается и отображается в интерфейсе приложения.

Пример кода для добавления блюда в меню в Android (Kotlin):

```
@POST("menu")
      suspend fun addDish(@Body dish: Menu): Menu
```

Для реализации клиентской части используется схема Model-View-Adapter, которая обеспечивает структурированность и упрощает поддержку:

–Model: отвечает за хранение данных, полученных из внешнего источника (например, REST API или локальной базы данных).

–View: отвечает за визуальное отображение данных и взаимодействие с пользователем.

–Adapter: связывает данные из модели с компонентами представления.

Реализованные этапы разработки клиентской части:

–Создание модели (Model): разработка классов данных, которые соответствуют структурам, используемым на сервере, реализация методов для получения данных через REST API, работу с локальной базой данных или файлом.

–Разработка представления (View): создание XML-файлов с макетами для экранов и элементов списка, разработка активностей или фрагментов, которые инициализируют представление и взаимодействуют с адаптером.

–Реализация адаптера (Adapter): создание адаптера для управления отображением данных в списках (RecyclerView или ListView), настройка привязки данных модели к элементам интерфейса.

3 Руководство программиста

Требования:

–Android Studio: Версия Arctic Fox (или выше) — рекомендуемая версия для полной поддержки Kotlin и современных библиотек.

–Kotlin: Версия 1.7.0 или выше.

–PostgreSQL: Версия 13 или выше для стабильной работы с ORM и REST API.

–Gradle: Минимальная версия 7.0.0.

Зависимости:

–Retrofit для работы с REST API.

–Gson для работы с JSON.

–Room для локальной базы данных (по необходимости).

–RecyclerView и ViewModel для работы с UI.

Основные методы и классы программы, и их назначение:

–Класс Category.kt – класс модели, представляющий категорию.

–Класс Customer.kt – класс модели, описывающий клиента.

–Класс Menu.kt – класс модели, представляющий пункт меню.

–Класс Order.kt – класс модели, описывающий заказ.

–Класс OrderItem.kt – класс модели, описывающий отдельный пункт заказа.

–Класс OrderResponse.kt – класс модели, представляющий ответ на запрос информации о заказе.

val formattedOrderDate: String – метод, возвращающий отформатированную дату заказа

–LoginActivity.kt – активность, управляющая экраном входа пользователя.

loginUser – метод, отправляющий запрос на сервер для авторизации пользователя.

onCreate – метод, вызываемый при создании активности, инициализирует поля и настраивает обработчики событий.

–MainActivity.kt – главная активность, управляющая экраном меню и пользовательской сессией.

onCreate – метод, вызываемый при создании активности, инициализирует интерфейс и настраивает кнопки и события.

fetchUserId – метод для получения идентификатора пользователя с сервера.

checkIfUserIsAdmin – метод, проверяющий, является ли пользователь администратором.

hideAdminButtons – метод для скрытия кнопок администрирования для пользователей, не являющихся администраторами.

–LoginActivity.kt – класс, реализующий экран для входа пользователя, где происходит проверка введенных данных и выполнение входа с использованием API для авторизации.

loginUser – метод, который выполняет запрос к серверу для авторизации пользователя с помощью введенных данных (имя пользователя и пароль), а затем сохраняет токен и ID пользователя в SharedPreferences для дальнейшего использования.

btnLogin.setOnClickListener – обработчик нажатия на кнопку "Войти", который инициирует процесс логина при заполненных полях для имени пользователя и пароля.

btnRegister.setOnClickListener – обработчик нажатия на кнопку "Регистрация", который открывает экран регистрации пользователя.

–MainActivity.kt – главный экран приложения, который отображает меню и детали о пользователе, включая возможность проверки, является ли пользователь администратором.

fetchUserId – метод для получения ID пользователя с сервера по имени пользователя с использованием Retrofit и сохранения его в SharedPreferences.

checkIfUserIsAdmin – метод для проверки на сервере, является ли пользователь администратором. В случае успеха сохраняет информацию об администраторе в SharedPreferences.

RecyclerView для меню – отображает список блюд с возможностью добавления выбранных позиций в заказ с учетом количества. Кнопки "Меню", "Заказ", "Профиль" обеспечивают переходы в соответствующие экраны.

logoutButton – кнопка для выхода пользователя из системы, которая удаляет токен из SharedPreferences и перенаправляет на экран входа.

menuButton – кнопка для открытия экрана меню с передачей данных о пользователе (имя и ID).

orderButton – кнопка для перехода на экран оформления заказа с передачей данных о пользователе.

profileButton – кнопка для открытия экрана профиля, если пользователь авторизован. В противном случае выводится сообщение об ошибке.

sharedPreferences – используется для хранения и получения информации о пользователе, например, токен, ID и имя пользователя.

–ProfileActivity.kt – экран, отображающий профиль пользователя, где отображаются его данные (ФИО, номер телефона, электронная почта) с возможностью редактирования и сохранения изменений.

fetchCustomer – метод, который выполняет запрос на сервер для получения информации о пользователе по его ID. В случае успешного получения данных отображает их на экране.

saveOrUpdateCustomer – метод, который выполняет запрос на сервер для сохранения или обновления данных пользователя (например, ФИО, телефон, email). Вызывается при нажатии кнопки "Сохранить" после редактирования информации.

enableEditing – метод, который включает или отключает возможность редактировать поля профиля (ФИО, номер телефона, email) и соответственно управляет видимостью кнопок редактирования и сохранения.

showErrorDialog – метод, который отображает диалоговое окно с ошибкой, если произошла ошибка при загрузке данных или сохранении изменений.

btnEdit.setOnClickListener – обработчик нажатия на кнопку редактирования, который включает режим редактирования полей профиля.

btnSave.setOnClickListener – обработчик нажатия на кнопку сохранения, который собирает данные из полей и вызывает метод сохранения данных на сервере.

btnCancel.setOnClickListener – обработчик нажатия на кнопку отмены редактирования, который отменяет изменения и выключает режим редактирования.

–RegistrationActivity.kt – экран для регистрации нового пользователя, где пользователь вводит имя пользователя и пароль для создания аккаунта.

registerUser – метод, который отправляет запрос на сервер для регистрации нового пользователя, передавая введенные данные (имя пользователя и пароль). В случае успешной регистрации выводит сообщение об успешном завершении и закрывает экран.

btnRegister.setOnClickListener – обработчик нажатия на кнопку "Зарегистрироваться", который проверяет введенные данные (имя пользователя и пароль), а затем вызывает метод для регистрации.

–MenuAdapter.kt – адаптер для отображения списка блюд в меню, который отображает название блюда, цену и количество выбранных порций.

setMenuList – метод, который обновляет список блюд в адаптере, вызывается при получении данных с сервера или изменении списка.

onBindViewHolder – метод, который связывает данные блюда с соответствующим элементом списка (ViewHolder).

–OrderAdapter.kt – адаптер для отображения списка заказов, который отображает информацию о заказах и предоставляет кнопки для изменения статуса или отмены заказа (для администратора).

onCreateViewHolder – метод, создающий и возвращающий новый элемент для отображения заказа (ViewHolder), с привязанными UI элементами.

onBindViewHolder – метод, который связывает данные заказа с элементами UI для каждого заказа в списке.

OrderViewHolder – класс, представляющий элемент списка заказа. В нем привязаны элементы UI, такие как ID заказа, дата, цена и статус.

btnChangeStatus.setOnClickListener – обработчик нажатия на кнопку изменения статуса заказа, который вызывает функцию для изменения статуса заказа.

btnCancelOrder.setOnClickListener – обработчик нажатия на кнопку отмены заказа, который вызывает функцию для отмены заказа.

–ApiClient.kt - объект, который предоставляет методы для создания и инициализации Retrofit-сервисов для различных API в приложении.

createService - универсальный метод для создания экземпляра сервиса на основе переданного класса. Используется для создания сервисов различных типов (например, ApiService, CustomerApiService).

createCustomerService - метод, создающий экземпляр CustomerApiService для работы с API пользователя.

apiService - инициализатор для создания экземпляра ApiService, который используется для взаимодействия с общими API (например, для работы с заказами и меню).

menuApiService - инициализатор для создания экземпляра MenuApiService, который используется для работы с API меню.

–RetrofitClient.kt - объект, который предоставляет экземпляр Retrofit с настройками для взаимодействия с API, а также настроенным логированием запросов и ответов.

instance - инициализатор для создания экземпляра Retrofit, который используется для создания запросов к серверу.

loggingInterceptor - объект, настроенный для логирования запросов и ответов, используется для отслеживания данных, передаваемых между клиентом и сервером.

–ApiService.kt - интерфейс, содержащий методы для работы с заказами, меню и пользователями через API.

getUserId - метод для получения ID пользователя по имени пользователя. Использует параметр username и возвращает результат в виде объекта UserIdResponse.

isAdmin - метод для проверки, является ли пользователь администратором, на основе его ID.

getMenu - метод для получения списка блюд из меню. Возвращает список объектов Menu.

addDish - метод для добавления нового блюда в меню. Принимает объект Menu и возвращает его после добавления.

getOrder - метод для получения информации о заказе по его ID. Возвращает объект Order.

createOrder - метод для создания нового заказа, принимая объект Order и возвращая созданный заказ.

addOrderItem - метод для добавления позиции в заказ. Принимает объект OrderItem и возвращает добавленный элемент заказа.

getOrdersByUserId - метод для получения списка заказов пользователя по его ID.

changeOrderStatus - метод для обновления статуса заказа, принимает ID заказа и возвращает результат в виде карты с информацией о статусе.

cancelOrder - метод для отмены заказа. Принимает ID заказа и возвращает результат в виде карты с информацией о статусе.

–CustomerApiService.kt - интерфейс, предоставляющий методы для работы с данными клиентов через API.

getCustomerByUserId - метод для получения данных о клиенте по его ID. Возвращает объект Customer.

addCustomer - метод для добавления нового клиента, принимает ID клиента и объект Customer, возвращает добавленного клиента.

updateCustomer - метод для обновления данных клиента, принимает ID клиента и объект Customer, возвращает обновленного клиента.

–UserApiService.kt - интерфейс, предоставляющий методы для регистрации и входа пользователя.

register - метод для регистрации нового пользователя. Принимает объект User и возвращает результат в виде Call<Void>.

login - метод для входа пользователя, принимая объект User и возвращая результат в виде объекта User с данными пользователя после успешного входа.

4 Руководство пользователя

Руководство пользователю поможет вам освоиться с основными функциями и страницами Android-приложения. В нем описаны ключевые экраны и действия, которые доступны пользователю. Просмотреть изображения и принцип работы можно в приложении 2.

4.1 Руководство для пользователя-клиента

Вход в систему:

Для того чтобы войти в приложение, выполните следующие шаги:

–Откройте Android-приложение.

–Перед вами откроется форма входа, как в приложении 2 рисунке 2. Введите ваше имя пользователя и пароль, если вы уже имеете аккаунт, а после нажмите кнопку Войти. Если введенные данные верны, вы будете перенаправлены на главный экран (пример представлен в приложении 2 рисунки 3-4). Если данные неверны, будет отображено сообщение об ошибке.

Регистрация:

Если у вас еще нет аккаунта, вы можете зарегистрироваться:

–Перейдите на экран Регистрация, нажав кнопку Регистрация на экране входа. Введите требуемые данные, такие как имя пользователя, пароль и другие данные. Нажмите кнопку Зарегистрироваться для создания нового аккаунта.

Просмотр меню:

–На главном экране приложения вы можете просматривать доступные блюда. Пример формы для меню представлен в приложении 2 рисунке 4.

Оформление заказа:

Для оформления заказа выполните следующие шаги:

–Нажмите кнопку Меню на главном экране. Выберите необходимые позиции меню в нужном количестве. Нажмите кнопку Оформить заказ для завершения процесса. Вы получите информацию о статусе вашего заказа и

сможете отслеживать его выполнение. В случае, если в процессе оформления заказа вы передумали, то нажмите кнопку Отмена. Пример представлен в приложении 2 рисунке 6.

Просмотр заказов:

–На главном экране нажмите кнопку Заказы, чтобы просматривать информацию о ваших предыдущих и действующих заказах. Пример представлен в приложении 2 рисунке 7.

Ввод данных о себе, редактирование аккаунта:

–На главном экране нажмите кнопку Профиль, чтобы посмотреть данные о своем аккаунте. Если необходимо их изменить, то нажмите кнопку Редактировать, а после изменения данных нажмите кнопку Сохранить. Пример представлен в приложении 2 рисунке 9.

4.2 Руководство для пользователя-администратора

Вход в систему:

Для того чтобы войти в приложение, выполните следующие шаги:

–Откройте Android-приложение.

–Перед вами откроется форма входа в приложение. Введите ваше имя пользователя и пароль, если вы уже имеете аккаунт, а после нажмите кнопку Войти. Если введенные данные верны, вы будете перенаправлены на главный экран. Если данные неверны, будет отображено сообщение об ошибке.

Работа с заказами:

–Нажмите кнопку Заказ на главном экране. Перед вами откроется список все заказов. Изначально они имеют статус “Принят”. При нажатии кнопки Изменить статус для выбранного заказа он меняет свой статус на “В процессе”, при повторном нажатии меняет статус на “В пути”, при следующем нажатии меняет статус на “Доставлен”. В случае необходимости можно отменить заказ, нажав кнопку Отменить заказ. В таком случае произойдет смена статуса заказа на “Отменен”. Пример представлен в приложении 2 рисунке 8.

Ввод данных о себе, редактирование аккаунта:

–На главном экране нажмите кнопку Профиль, чтобы посмотреть данные о своем аккаунте. Если необходимо их изменить, то нажмите кнопку Редактировать, а после изменения данных нажмите кнопку Сохранить. Пример представлен в приложении 2 рисунке 9.

Работа с меню:

–Нажмите кнопку Меню на главном экране. При необходимости добавления новой позиции нажмите кнопку “Добавить позицию”. После введите необходимые данные и сохраните. Пример представлен в приложении 2 рисунке 5.

Заключение

В ходе выполнения курсовой работы была разработана автоматизированная информационная система для доставки еды из ресторана, которая значительно улучшает процессы оформления заказов, их отслеживания и управления данными. Внедрение автоматизации в ресторанах не только оптимизирует внутренние бизнес-процессы, но и способствует повышению качества обслуживания клиентов и улучшению управления рестораном.

Разработанная система позволяет ускорить выполнение рутинных задач, уменьшить вероятность ошибок при обработке данных и обеспечить более высокую точность и надежность в учете заказов и блюд. Автоматизация процессов также способствует улучшению взаимодействия с клиентами, предоставляя им возможность отслеживать статус заказов в реальном времени и получать актуальную информацию о блюдах и предложениях ресторана.

Ключевым элементом системы является база данных, которая обеспечивает хранение всей необходимой информации о пользователях, заказах и блюдах. Это позволяет оперативно работать с данными, ускорять их обработку, и анализировать эффективность работы ресторана.

Для реализации проекта была использована база данных Postgre для хранения данных, а также среда разработки Android Studio с использованием языка Kotlin для создания мобильного приложения. Такой выбор технологий обеспечил высокую производительность системы и создание удобного интерфейса как для клиентов, так и для администраторов ресторана.

В результате, разработанная система представляет собой важное решение, которое способствует повышению эффективности работы ресторана и улучшению качества обслуживания клиентов. Внедрение такой системы помогает ресторану укрепить свои позиции на рынке, повысить конкурентоспособность и расширить клиентскую базу.

Список литературы

1) Введение в разработку приложений для смартфонов на ОС Android / А.Семакова Национальный открытый Университет «ИНТУИТ», 2016

2) Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВ-Петербург, 2015. — 303 с.

3) 2016. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с.

4) Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. - 375 с.

Приложение 1. Модели данных

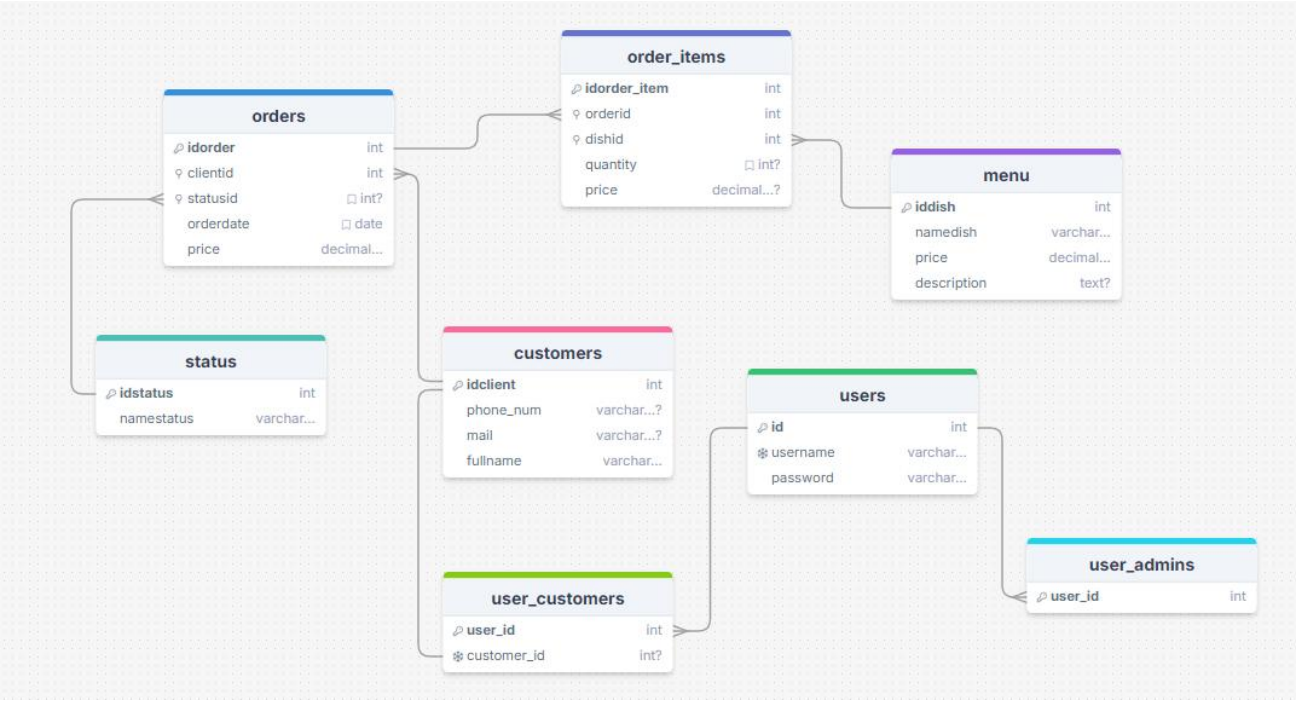

Рисунок 1 – физическая модель данных

Приложение 2. Скриншоты программы

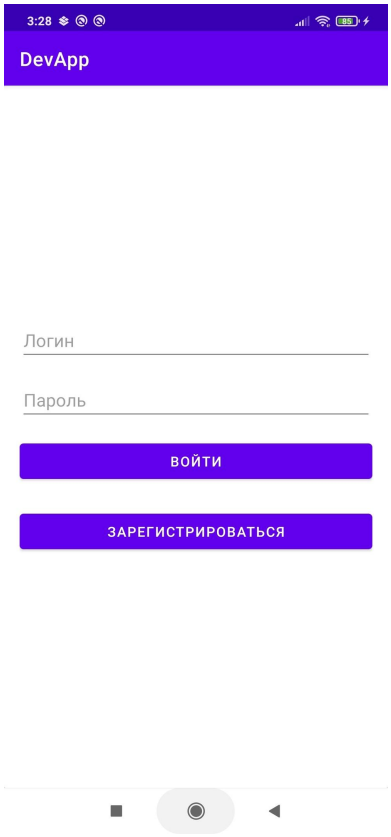

Рисунок 2 – форма входа

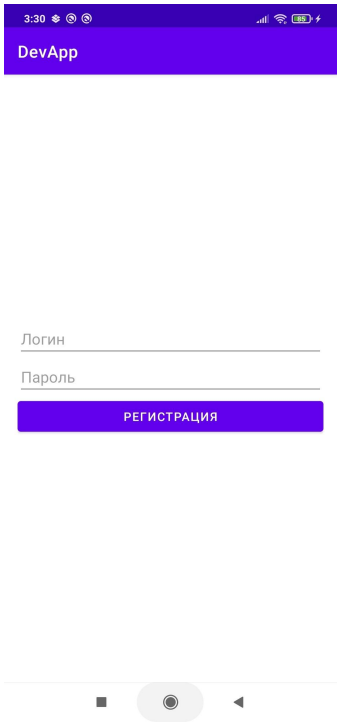

Рисунок 3 – форма регистрации

Рисунок 4 – главный экран

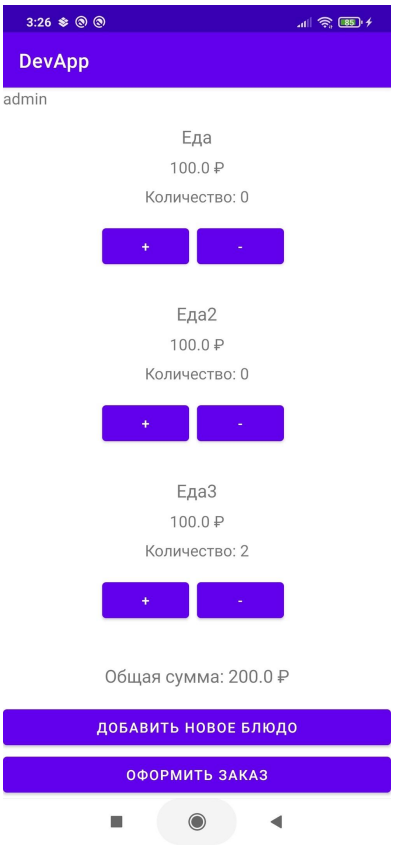

Рисунок 5 – форма “Меню” для администратора

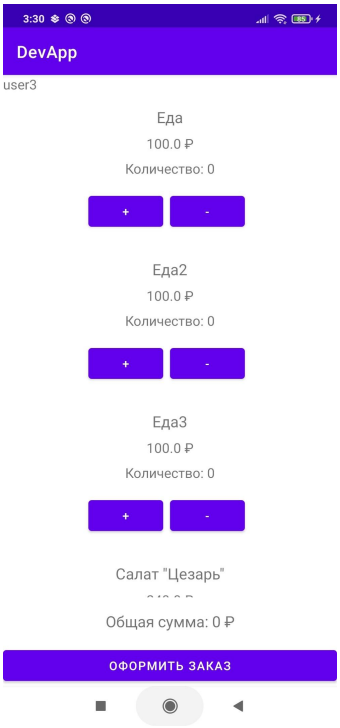

Рисунок 6 – форма “Меню” для клиента

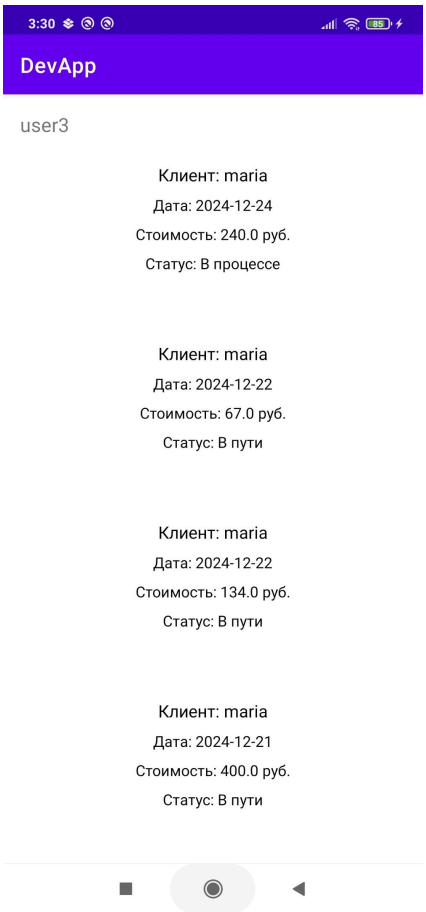

Рисунок 7 – форма “Заказ” для клиента

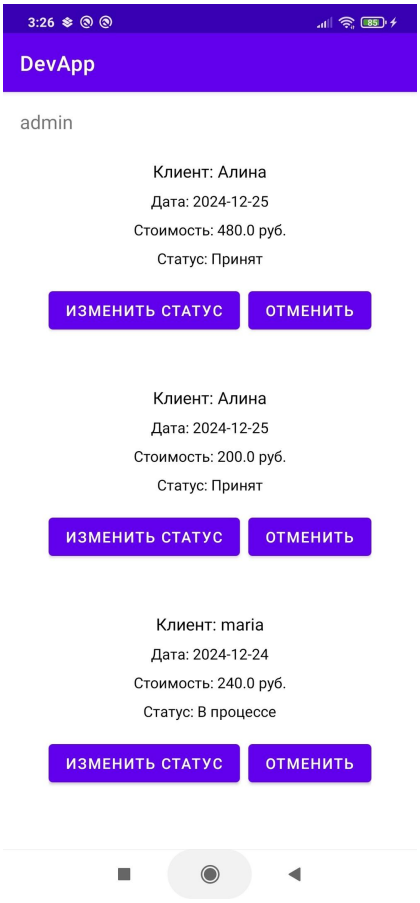

Рисунок 8 – форма “Заказ” для администратора

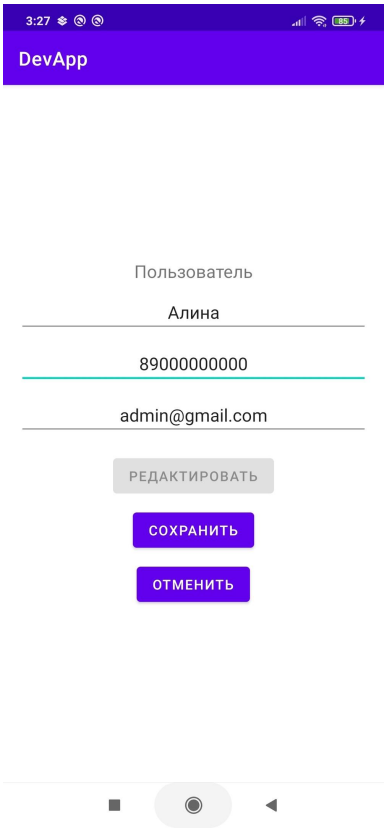

Рисунок 9 – форма “Профиль”

Приложение 3. Исходный код

Ссылка на GitHub страницу репозитория с исходным кодом программы: https://github.com/MariMedvedeva/DevApp