ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА ПОДСИСТЕМЫ

«КОФЕЙНЯ: ЗАКАЗЫ»

Л109. 23КП01. 001 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-11		05.12.2024	М.К. Абрамов
	(Группа)	(Подпись)	(Дата)	(И.О. Фамилия)
Преподаватель			05.12.2024	Ю.С. Маломан
		(Подпись)	(Дата)	(И.О. Фамилия)

Архангельск 2024

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

БД – база данных

ИС – информационная система

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

API – интерфейс программирования приложения

ERD – диаграмма «сущность-связь»

HTTP – протокол передачи гипертекста

IDE – интегрированная среда разработки

JSON – текстовый формат обмена данными

SQL – язык структурированных запросов

UML – язык структурированных запросов

ВВЕДЕНИЕ

Актуальность разрабатываемого проекта заключается в том, что он решает важную задачу в сфере автоматизации процессов управления заказами в кофейнях.

В условиях современного рынка, где конкуренция среди заведений общественного питания возрастает, эффективное управление заказами становится ключевым фактором успеха. Кофейни сталкиваются с проблемами, связанными с обработкой заказов, взаимодействием с клиентами и оптимизацией рабочего процесса.

Разработка подсистемы для управления заказами в кофейне позволит значительно упростить процесс приема и обработки заказов, улучшить взаимодействие с клиентами и повысить общую эффективность работы заведения.

Целью курсового проектирования является разработка подсистемы обеспечивающей возможность быстрого и удобного оформления заказов, а также управления ими в реальном времени.

Для достижения поставленной цели требуется решить следующие задачи:

- провести сбор и анализ требований целевой аудитории,
- проанализировать информационные источники по предметной области,
- изучить существующие решения в области автоматизации заказов в кофейнях,
- спроектировать архитектуру подсистемы,
- спроектировать диаграмму вариантов использования подсистемы,
- выбрать состав программных и технических средств для реализации проекта,
- спроектировать БД,
- создать БД в выбранной СУБД,

- разработать API для взаимодействия мобильного приложения и телеграм-бота с БД,
- реализовать разграничение прав доступа пользователей,
- обеспечить защиту данных,
- разработать интерфейс мобильного приложения,
- разработать мобильное приложение,
- реализовать функциональность оформления заказа,
- разработать телеграм-бота,
- реализовать функциональность обработки заказов,
- реализовать функциональность управления заказами,
- выполнить структурное тестирование ПО,
- выполнить функциональное тестирование ПО,
- разработать программную документацию,
- разработать эксплуатационную документацию.

В результате выполнения поставленных задач будет создана подсистема «Кофейня: Заказы», которая значительно упростит процесс управления заказами и повысит уровень обслуживания клиентов.

1 Анализ и разработка требований

1.1 Назначение и область применения

Подсистема предназначена для владельцев и сотрудников кофеен, а также для клиентов, желающих упростить процесс оформления и управления заказами. Подсистема позволит пользователям эффективно оформлять, принимать и обрабатывать заказы в реальном времени, а также отслеживать их статус и просматривать информацию о текущих и завершенных заказах.

1.2 Постановка задачи

Необходимо разработать подсистему "Кофейня: Заказы", которая будет включать в себя телеграм-бот для сотрудников кофейни и мобильное приложение для клиентов. Телеграм-бот будет предназначен для обработки заказов, позволяя сотрудникам эффективно управлять процессом выполнения заказов. Мобильное приложение предоставит клиентам возможность просматривать меню, оформлять заказы и отслеживать их статус.

1.3 Описание алгоритма функционирования системы

При запуске мобильного приложения отображается страница меню кофейни, в нижней панели навигации находится иконка пользователя, после нажатия на иконку открывается окно авторизации и регистрации.

При запуске телеграм-бота командой «/start» бот проводит авторизацию сотрудника в системе, после чего открывается функциональность сотрудника в зависимости от его роли в системе.

Клиенты имеют доступ к функциональности, позволяющей им просматривать список товаров, оформлять заказы и просматривать историю

своих заказов. Для доступа к этим возможностям клиентам необходимо пройти авторизацию или регистрацию.

Сотрудники кофейни могут использовать телеграм-бот для просмотра списка новых заказов, а также для отслеживания заказов, находящихся в работе и выполненных. Они могут изменять статус заказов, выбирая соответствующие состояния, такие как "новый", "в работе" или "готов". Кроме того, работники могут выбирать адреса для работы и указывать свое состояние, например, "встать в работу" или "выйти с работы".

Администраторы системы обладают расширенными правами и могут управлять сотрудниками. Они могут добавлять новых сотрудников, удалять существующих, изменять данные и просматривать список всех сотрудников кофейни. Это обеспечивает эффективное управление персоналом и поддержание актуальности информации в системе.

На рисунке 1 изображена диаграмма вариантов использования мобильного приложения.

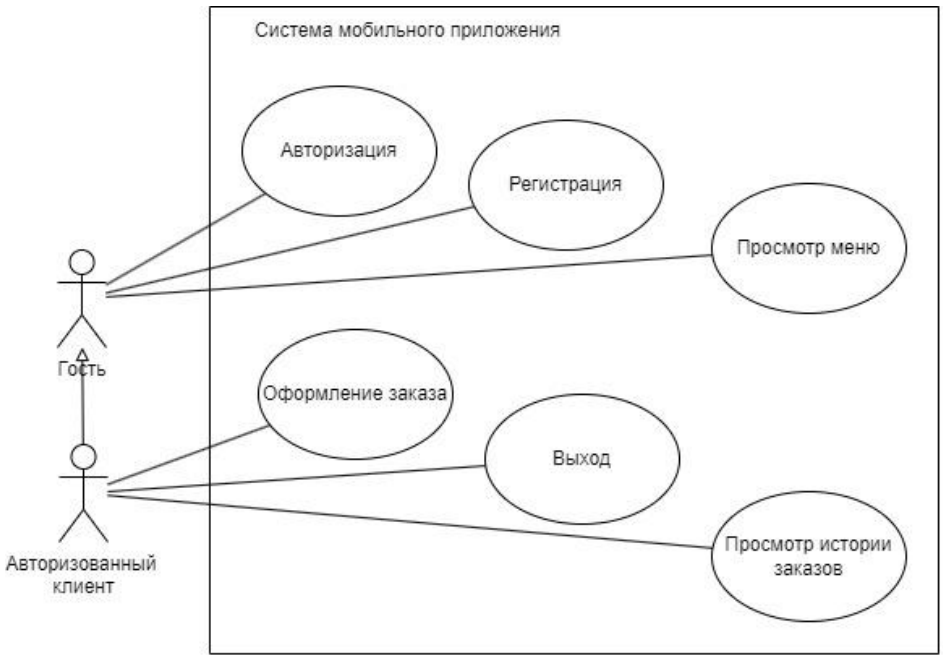

Рисунок 1 – Диаграмма вариантов использования мобильного приложения

На рисунке 2 изображена диаграмма вариантов использования телеграм-бота.

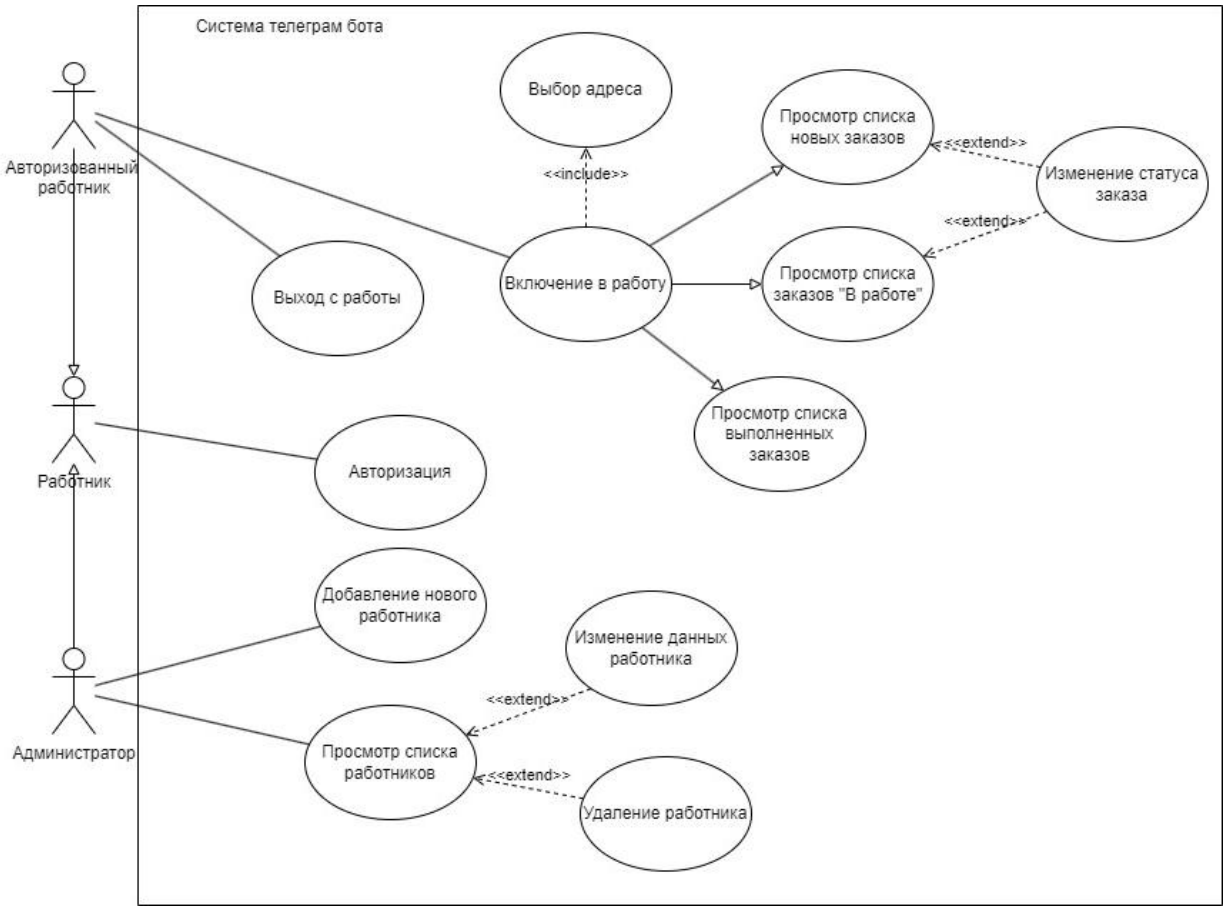

Рисунок 2 – Диаграмма вариантов использования телеграм-бота.

1.4 Выбор состава программных и технических средств

Согласно цели проекта требуется создать подсистему для оформления и обработки заказов.

Работа с мобильным приложением будет осуществляться на мобильных устройствах с установленной операционной системой Android 6.0 или iOS 9.0 и выше с интернет-подключением.

Работа с телеграм-ботом будет осуществляться с помощью приложения «Телеграм» с интернет-подключением.

В качестве СУБД [1] выбрана MySQL 8.0, так как эта СУБД обладает высокой производительностью, простотой в использовании и масштабируемостью, что делает её идеальным выбором для обработки заказов в реальном времени. MySQL также поддерживает множество функций, необходимых для эффективного управления данными, включая транзакции и индексацию.

Мобильное приложение будет написано на языке программирования Dart с использованием фреймворка Flutter, так как он поддерживает кроссплатформенность, обеспечивая высокую производительность и отзывчивость интерфейса. Flutter также предоставляет богатый набор виджетов и инструментов для создания привлекательного пользовательского интерфейса.

Телеграм-бот будет разработан на языке программирования Python, так как этот язык поддерживает почти все ОС и платформы, имеет много фреймворков и простой синтаксис.

Для разработки мобильного приложения будет использоваться IDE Visual Studio Code, так как она поддерживает множество расширений для работы с Dart и Flutter, а также обеспечивает удобные инструменты для отладки и тестирования.

Для разработки телеграм-бота будет использоваться IDE PyCharm Community Edition 2023.3.4, так как эта среда предлагает удобные инструменты для работы с Python, включая менеджер пакетов и средства отладки.

Для функционирования системы на стороне сервера достаточны следующие программные и технические средства:

- ОС Ubuntu версии 24 и выше,
- сервер БД: MySQL версии не ниже 8.0,
- процессор с частотой 2 ГГц,
- свободная оперативная память объемом 8 ГБ,

- ПО для конфигурирования, управления и администрирования сервера БД – MySQL Workbench,
- ПО для работы API – dotnet-sdk версии не ниже 8.0,
- ПО для работы телеграм-бота – Python версии не ниже 3.12.

Для функционирования системы на стороне клиента достаточны следующие программные и технические средства:

- ОС Android версии 6.0 или iOS версии 9.0 и выше,
- процессор с частотой 2 ГГц,
- свободная оперативная память в объеме 2 ГБ,
- свободное место в хранилище 200 МБ,
- постоянное интернет-подключение.

Для функционирования системы на стороне сотрудника достаточны следующие программные и технические средства:

- ОС Android версии 6.0 или iOS версии 9.0 и выше,
- процессор с частотой 2 ГГц,
- свободная оперативная память в объеме 2 ГБ,
- свободное место в хранилище 600 МБ,
- постоянное интернет-подключение,
- приложение «Телеграм».

2 Проектирование ПО

2.1 Проектирование интерфейса пользователя

В рамках разработки мобильного приложения «coffee_shop» создан интерфейс пользователя в виде wireframe и мокапов при помощи инструмента draw.io. Эти визуальные представления позволяют наглядно увидеть структуру приложения, его основные элементы и функциональность.

Мокапы страниц «Меню», «Авторизация/Регистрация», «Личные данные» изображены на рисунке 2.

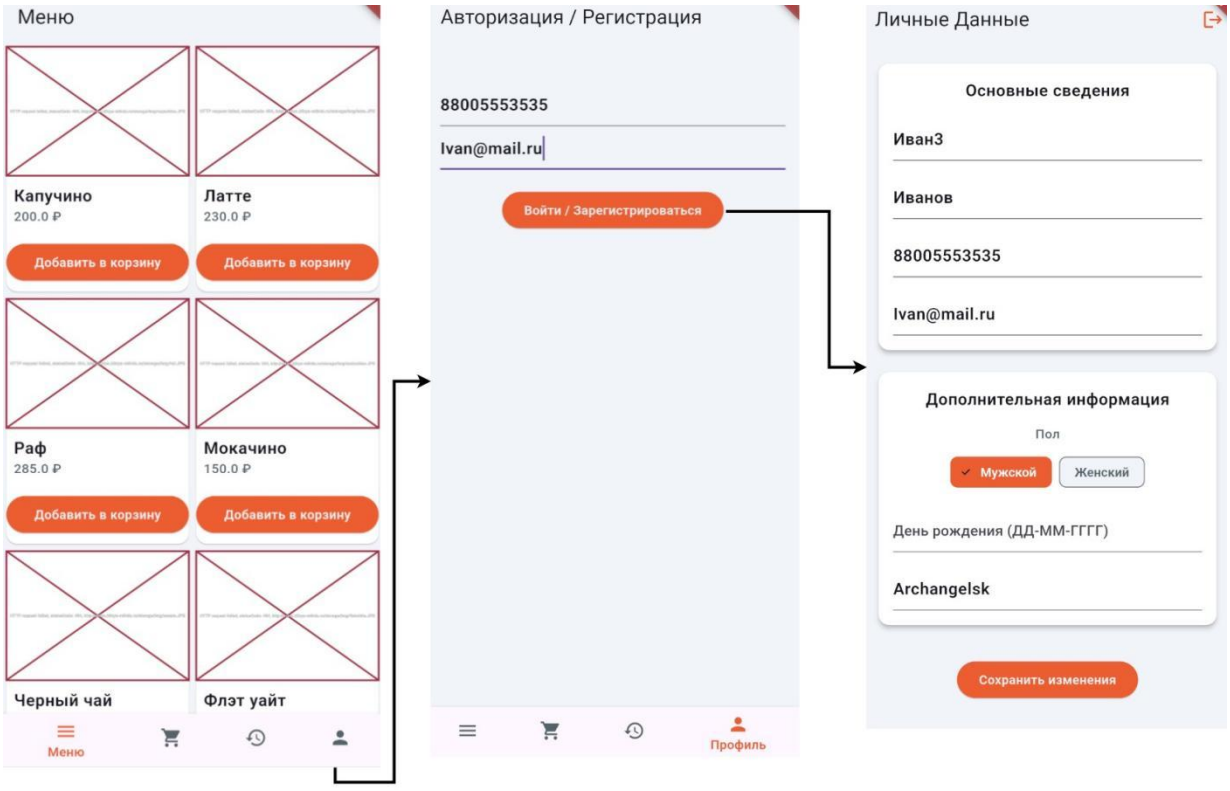

Рисунок 2 – draw.io. Мокап авторизации пользователя

Для мобильного приложения были выбраны следующие цвета:

- основной цвет – #ec5e32,
- цвет фона – #f1f4f8,

- вторичный цвет фона – #FFFFFF,
- основной цвет текста – #14181b,
- вторичный цвет текста – #57636c.

2.2 Разработка архитектуры программного обеспечения

Подсистема предназначена для сотрудников кофейни и их клиентов. Архитектура системы построена на основе клиент-серверной модели и включает в себя несколько ключевых компонентов: серверная часть системы, мобильное приложение, телеграм-бот, БД.

Для серверной части будет создан API, позволяющей клиенту взаимодействовать с сервером. Диаграмма развертывания компонентов [4] изображена на рисунке 3.

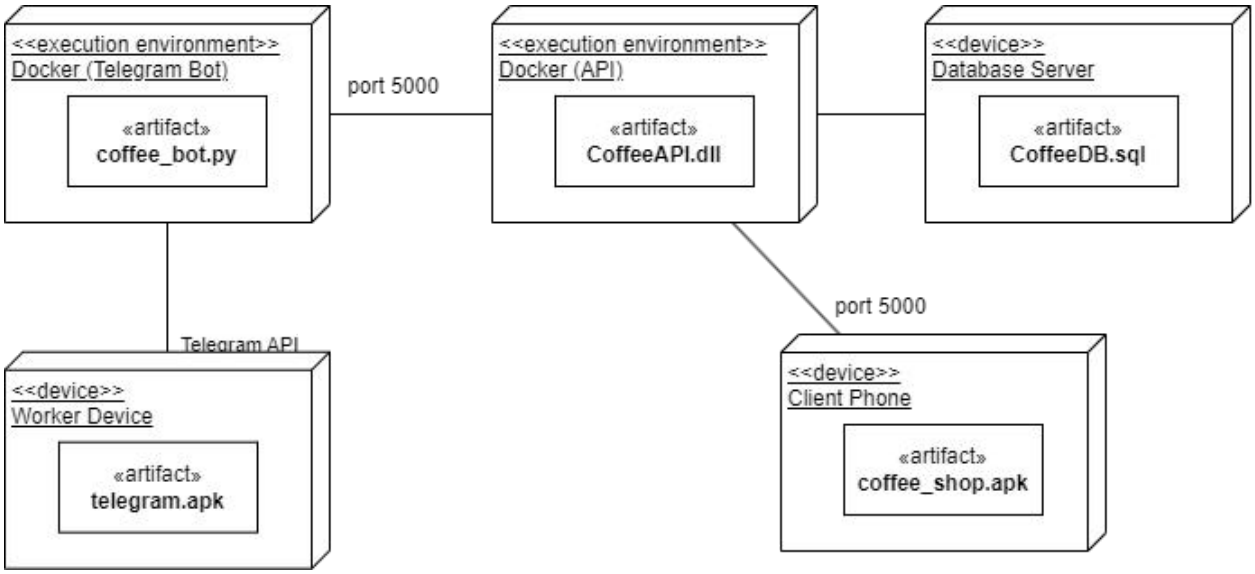

Рисунок 3 – draw.io. Диаграмма развертывания системы

2.3 Проектирование базы данных

В рамках курсового проектирования требуется разработать БД для системы оформления и обработки заказов кофеен. Система будет использоваться заказчиками, работниками и клиентами кофейни.

Модели БД созданы при помощи MySQL Workbench. На рисунке 4 в виде ERD показана физическая модель предметной области.

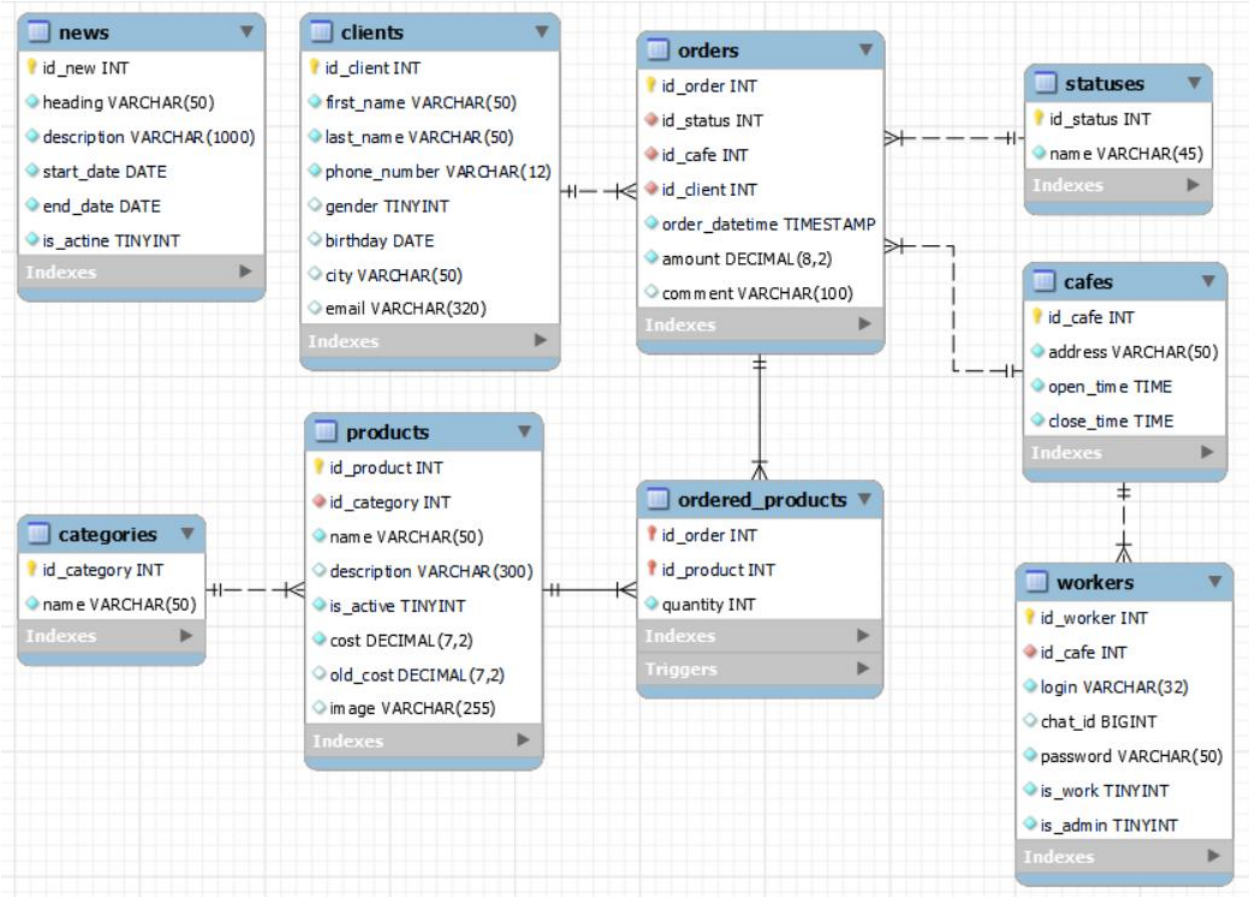

Рисунок 4 – MySQL Workbench. Физическая модель

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

В ходе курсового проектирования были разработаны: мобильное приложение на Dart с использованием фреймворка Flutter в Visual Studio Code, телеграм-бот на Python [3] в PyCharm Community, Web-API приложение на C# [2] в Visual Studio.

Взаимодействие мобильного приложения с сервером будет происходить при помощи HTTP-запросов к API, ответы будут возвращаться в формате JSON. Для реализации HTTP-запросов использован сетевой клиент из библиотеки http. Код метода для получения списка товаров отправкой GET-запроса на сервер представлен листингом 1.

Листинг 1 – Код метода для отправки POST-запроса на сервер

```
/// Загружает список продуктов с сервера.
///
/// Возвращает:
///   [Future<List<Product>>] - Асинхронный объект, который
/// завершится списком продуктов.
///
/// Исключения:
///   - [Exception] - Если произошла ошибка при загрузке
/// данных или если сервер вернул ошибку.
Future<List<Product>> fetchProducts() async {
  try {
final response = await http.get(
Uri.parse('${baseUrl}products'));
    if (response.statusCode == 200) {
      // Если сервер вернул 200 OK, парсим данные
      List<dynamic> jsonData = json.decode(response.body);
      return jsonData.map((json) =>
Product.fromJson(json)).toList();
    } else {
```

```
// Если сервер вернул ошибку, выбрасываем исключение
     throw Exception('Ошибка загрузки продуктов:
${response.statusCode}');
    }
  } catch (e) {
    // Если произошла ошибка (например, нет подключения),
// выбрасываем исключение с общим сообщением
    throw Exception('Не удалось загрузить продукты. Проверьте
подключение к интернету.');
  }
}
```

Для постоянного получения новых заказов в телеграм-боте была разработана функция представленная листингом 2.

Получение информации из БД осуществляется посредством Web-API приложения [5], код получения заказов по адресу и статусу представлен листингом 3.

Листинг 2 – Код функции непрерывного получения новых заказов

```
async def monitor_new_orders(msg: Message):
    """   Мониторит новые заказы и отправляет сообщения об этих
    заказах пользователю.
    :param msg: Сообщение от пользователя.
    """
    logging.info("Посещение функции monitor_new_orders")
    worker = await APIMethods.get_worker_state(msg.chat.id)
    while worker:
    worker_address_id = await APIMethods.get_worker_address(
    msg.chat.username.replace("_", "\_"))
    worker_address_str = await APIMethods.get_address_dict()
    worker_address_str = worker_address_str.get(
    worker_address_id)
    # Метод для получения новых заказов
    new_orders_items = await APIMethods.monitor_new_orders(
        worker_address_str)
    for order_id, order_msg in new_orders_items.items():
    # Задержка между отправками сообщений
        await asyncio.sleep(0.5)
        # Кнопки для обновления статуса заказа
        order_keyboard = await orders_inline_keyboard(order_id)
        await msg.answer(order_msg, reply_markup=order_keyboard)
    # Задержка перед новым запросом
    await asyncio.sleep(25)
```

Листинг 3 – Код получения заказов по адресу и статусу

```
// GET: api/orders/address=ЦУМ&status=Новый
[HttpGet("address={address}&status={status}")]
public async Task<ActionResult<IEnumerable
<orders_full_info_view>>> GetOrdersByAddressAndStatus(
string address, string status = "Новый")
{
    // Поиск заказов по адресу и статусу
    var orders = await _context.orders_full_info_views
        .Where(o => o.cafe_address == address
        && o.order_status == status)
        .ToListAsync();
    if (orders == null || !orders.Any())
    {
        return NotFound();
    }
    return Ok(orders);
}
```

3.2 Реализация интерфейса пользователя

Интерфейс мобильного приложения разработан с использованием постраничной навигации, в приложении разработаны различные элементы управления, стили и виджеты для упрощения работы. Навигация в приложении реализована с помощью виджета Navigator, который управляет стеком объектов Route, представляющих страницы в приложении.

Для отображения информации о заказе был разработан виджет OrderInfoCard, который изображен на рисунке 5.

Рисунок 5 – coffee_shop. Виджет OrderInfoCard

Код виджета OrderInfoCard представлен листингом 4.

Листинг 4 – Код виджета OrderInfoCard

```
// Виджет для отображения информации о заказе
class OrderInfoCard extends StatelessWidget {
  final Order order;
  final VoidCallback onTap;

  const OrderInfoCard({
    Key? key,
    required this.order,
    required this.onTap,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.all(8.0),
      child: ListTile(
        title: Row(
          children: [
            Text(
              'Заказ №${order.idOrder} ',
              style: TextStyle(
                color: Theme.of(context).primaryColor,
                fontSize: 18,
                fontWeight: FontWeight.w600,
              ),
            ),
            Spacer(),
            Text(
              '${order.getFormattedDate()}',
              style: Theme.of(context).textTheme.bodySmall,
            ),
          ],
        ),
        subtitle: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Адрес: ${order.cafeAddress}'),
            Row(
              children: [
                Text('Сумма: ${order.orderAmount}₽'),
                Spacer(),
                Text(
                  '${order.orderStatus}',
```

```
                style: TextStyle(color: order.getStatusColor()),
              ),
            ],
          ),
        ],
      ),
      onTap: onTap,
),);}}
```

3.3 Разграничение прав доступа пользователей

Разграничение прав доступа реализовано разделением сотрудников и клиентов по разным таблицам. Права администратора у сотрудника появляются при True значении в поле is_admin.

Пример кода разграничения прав пользователей телеграм-бота представлен листингом 5.

Листинг 5 – Код разграничения прав пользователей телеграм-бота

```
async def keyboard_menu(msg: Message) -> ReplyKeyboardMarkup:
    """
    Создает клавиатуру для главного меню.
    :return: Объект ReplyKeyboardMarkup с кнопками меню.
    """
    # Определяем кнопки для клавиатуры
    if await APIMethods.get_admin_state
    (msg.chat.username.replace("_", "\_")):
        keyboard_list = [
            [KeyboardButton(text=variables.new_worker)],
            [KeyboardButton(text=variables.read_workers)],
            [KeyboardButton(text=variables.del_worker)]
        ]
    else:
        keyboard_list = [
            [KeyboardButton(text=variables.address),
            KeyboardButton(text=variables.work)],
            [KeyboardButton(text=variables.home),
            KeyboardButton(text=variables.complete)]
        ]
    # Создаем и возвращаем объект клавиатуры
    keyboard = ReplyKeyboardMarkup(keyboard=keyboard_list,
    resize_keyboard=True,input_field_placeholder="Выберите
    действие:")
    return keyboard
```

4 Тестирование и отладка ПО

4.1 Структурное тестирование

Во время курсового проектирования проведено тестирование методом белого ящика для основных команд телеграм-бота, результаты в таблице 1.

Таблица 1 – Тестирование основных команд телеграм-бота

Действие	Ожидаемый результат	Фактический результат
Ввести команду /start	Появление сообщения с просьбой ввести пароль	Совпадает с ожидаемым
Ввести команду /start, ввод пароля «qwerty»	Появление сообщения об успешной авторизации	Совпадает с ожидаемым
Ввести команду /start, ввод пароля «qwerty1»	Появление сообщения о неверном пароле, просьба ввести пароль	Совпадает с ожидаемым
Ввести команду /address	Сообщение со списком адресов	Совпадает с ожидаемым
Ввести команду /work	Сообщение о начале работы	Совпадает с ожидаемым
Ввести команду /home	Сообщение о завершении работы	Совпадает с ожидаемым
Ввести команду /work, ввести команду /complete	Сообщения о выполненных заказах	Совпадает с ожидаемым

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование мобильного приложения методом черного ящика, результаты тестирования в таблице 2.

Таблица 2 – Тестирование мобильного приложения методом черного ящика

Действие	Ожидаемый результат	Фактический результат
Нажать на иконку меню	Открытие окна с меню кофейни, загрузка и отображение карточек товаров	Совпадает с ожидаемым
Нажать на иконку меню, нажать на кнопку «добавить а корзину» у карточки товара	Добавление товара в корзину, замена кнопки «добавить в корзину» на счетчик	Совпадает с ожидаемым
Нажать на иконку меню, нажать на кнопку «добавить а корзину» у карточки товара, нажать на иконку плюс у счетчика в карточке товара	Увеличение числа на счетчике на 1, увеличение кол-ва выбранного товара в корзине	Совпадает с ожидаемым
Нажать на иконку корзины	Переход на экран корзины	Совпадает с ожидаемым
Нажать на иконку корзины, нажать кнопки «Далее»	Переход на экран оформления заказа	Совпадает с ожидаемым
Нажать иконку профиля	Переход на экран авторизации/регистрации	Совпадает с ожидаемым
Нажать на иконку профиля, ввести «88005553535» в поле ввода телефона, ввести «Ivan@mail.ru» в поле Email	Открытие экрана профиля с данными пользователя Ивана	Совпадает с ожидаемым

5 Инструкция по эксплуатации ПО

5.1 Установка программного обеспечения

Для функционирования системы на стороне сервера достаточны следующие программные и технические средства:

- ОС Ubuntu версии 24 и выше,
- процессор с частотой 2 ГГц,
- свободная оперативная память объемом 8 ГБ,
- свободное место на диске – 10 ГБ,
- дополнительные компоненты: dotnet-sdk версии не ниже 8.0, Python версии 3.12 или выше,
- наличие интернет-соединения,
- MySQL Server 8.0,
- Docker,
- docker-compose.

Процесс создания БД:

- авторизация на сервере, в СУБД,
- выполнить импорт / запуск скрипта базы данных CoffeShopDB.sql.

Для функционирования мобильного приложения на стороне клиента достаточны следующие программные и технические средства:

- ОС Android версии 6.0 или iOS версии 9.0 и выше,
- процессор с частотой 2 ГГц,
- свободная оперативная память в объеме 2 ГБ,
- свободное место в хранилище 200 МБ,
- постоянное интернет-подключение.

Для функционирования системы на стороне сотрудника достаточны следующие программные и технические средства:

- ОС Android версии 6.0 или iOS версии 9.0 и выше,
- процессор с частотой 2 ГГц,

- свободная оперативная память в объеме 2 ГБ,
- свободное место в хранилище 600 МБ,
- постоянное интернет-подключение,
- приложение «Телеграм».

Для установки мобильного приложения требуется выбрать собранный coffee_shop.apk файл с приложением в проводнике и установить его.

В мобильном приложении используются следующие данные для авторизации:

- номер телефона – 88005553535,
- электронная почта – Ivan@mail.ru.

В телеграм-боте:

- логин – задан по умолчанию как логин телеграм,
- пароль – passwd.

5.2 Инструкция по работе

Для запуска мобильного приложения «coffee_shop» требуется нажать на иконку этого приложения. При запуске мобильного приложения, пользователя встречает меню кофейни. Для авторизации требуется нажать на иконку профиля в правом нижнем углу страницы, ввести данные для авторизации и нажать кнопку «Войти/Зарегистрироваться», пользователь войдет как клиент. Страница «Авторизация/Регистрация» изображена на рисунке 6.

После авторизации пользователь перенаправляется на экран своего профиля. Пользователю мобильного приложения доступны экран меню, экран корзины, экран истории заказов, экран профиля, к которым можно получить доступ через навигационную панель внизу экрана. Перечисленные страницы изображены на рисунке 7.

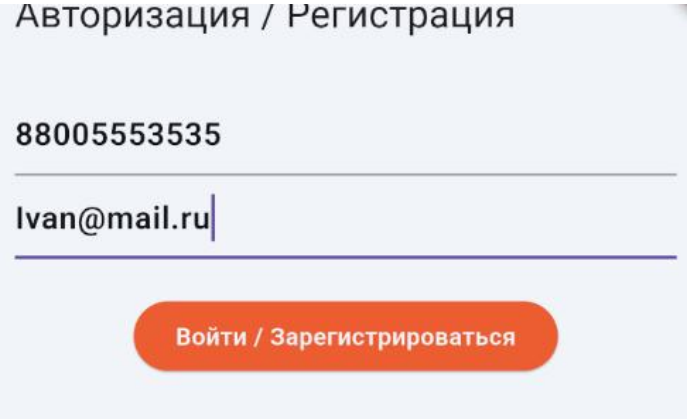

Рисунок 6 – coffee_shop. Вид страницы «Авторизация/Регистрация»

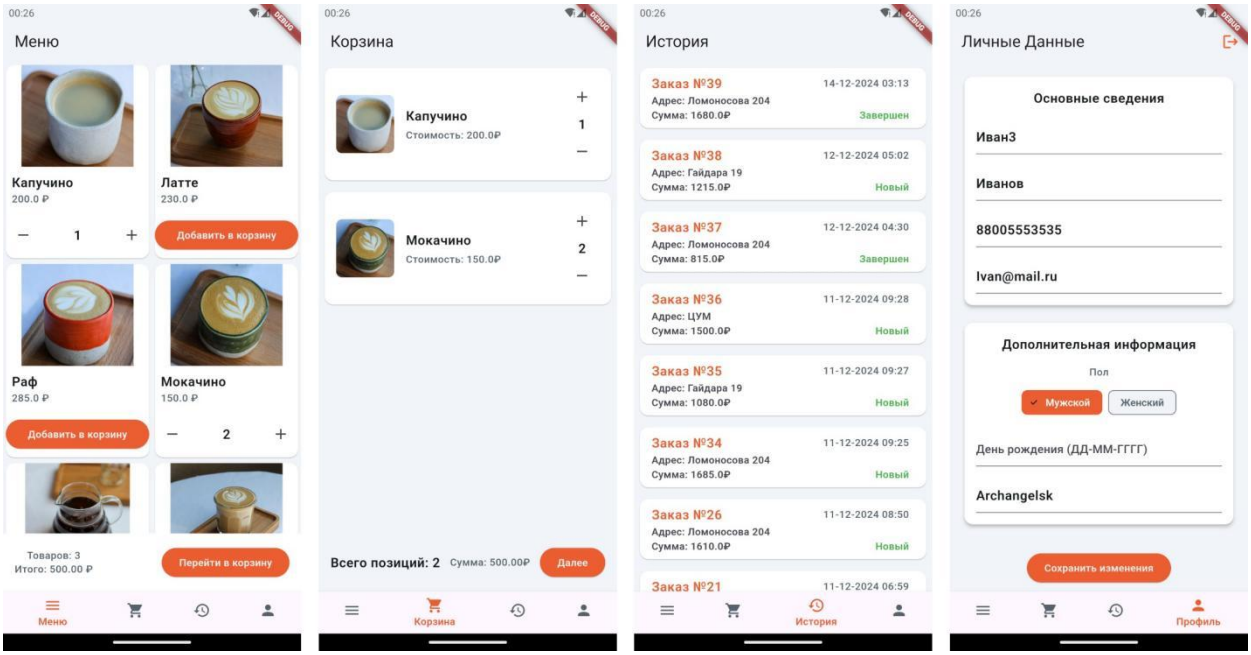

Рисунок 7 – coffee_shop. Вид страниц «Меню», «Корзина», «История», «Профиль»

Для оформления заказа требуется перейти на страницу «Меню» при помощи навигационной панели, нажав на иконку меню. После требуется выбрать необходимые товары нажав на кнопку «Добавить в корзину», если требуется несколько одинаковых товаров, то выбрать необходимое количество при помощи кнопок-иконок «Плюс» и «Минус». Затем требуется нажать на кнопку «Перейти в корзину» на странице «Меню» или на иконку корзины в навигационной панели. На странице «Корзины» необходимо

нажать на кнопку «Далее». Пользователя перенаправит на страницу оформления заказа, где необходимо выбрать адрес кофейни, где будет выполнятся заказ и нажать на кнопку «Оформить заказ». Процесс оформления заказа изображен на рисунке 8.

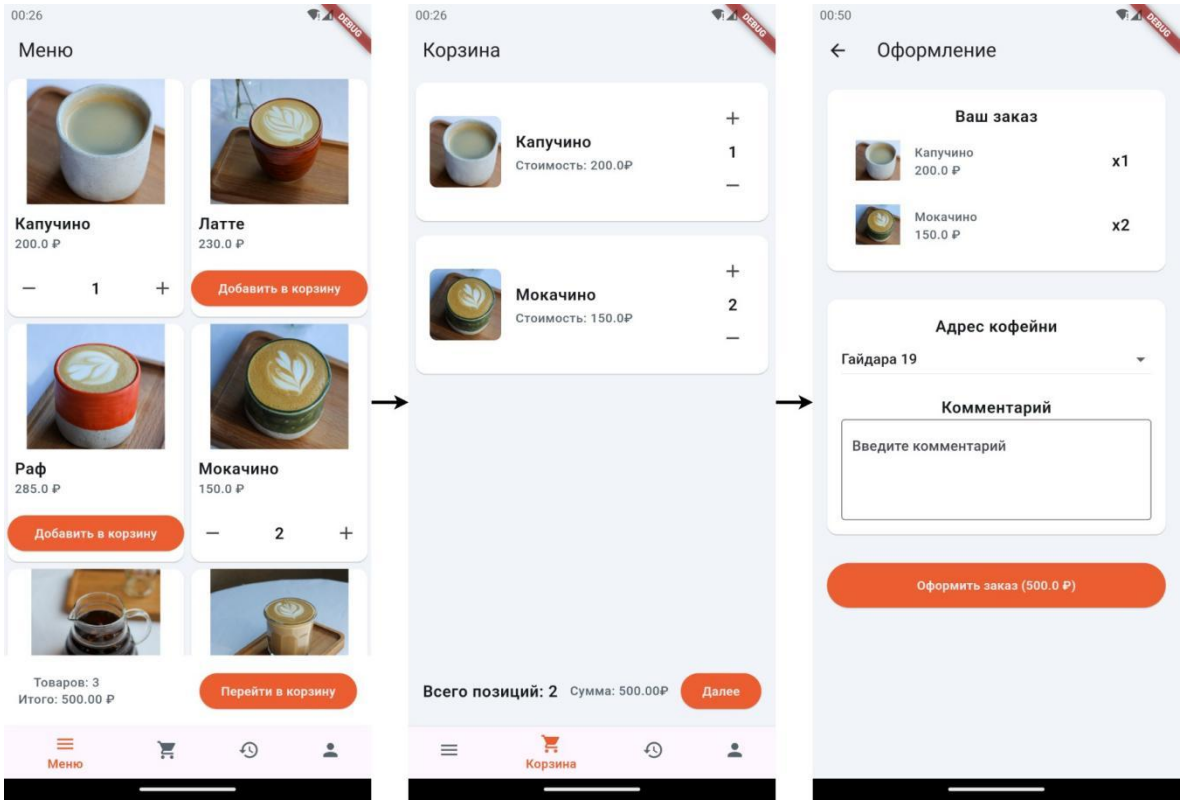

Рисунок 8 – coffee_shop. Процесс оформления заказа

Пользователю мобильного приложения так же доступен просмотр истории заказов и выход из аккаунта. Чтобы просмотреть историю заказов и их состав пользователю необходимо перейти на страницу истории и нажать на интересующий заказ. Чтобы выйти из аккаунта пользователю необходимо перейти на страницу профиля, нажать на иконку выхода из аккаунта в верхнем правом углу и подтвердить выход. Просмотр истории заказов изображен на рисунке 9, выход из аккаунта изображен на рисунке 10.

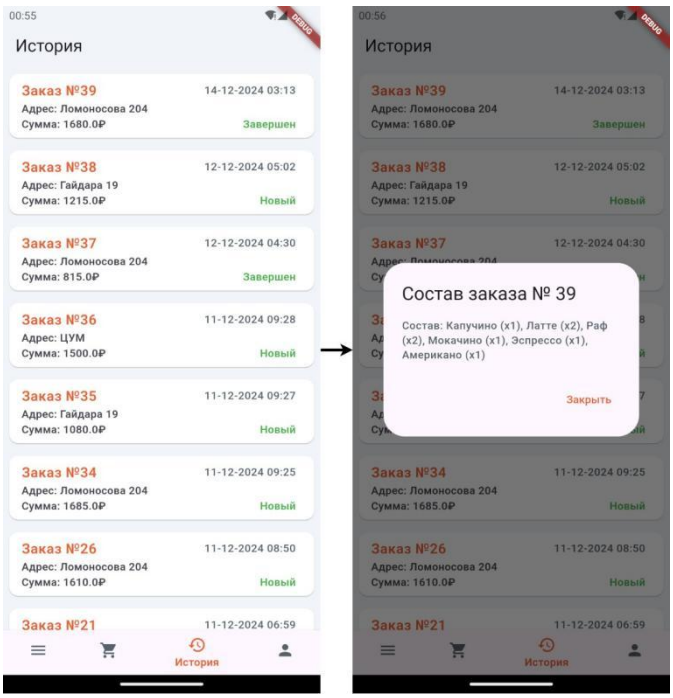

Рисунок 9 – coffee_shop. Просмотр истории заказов и состава заказа

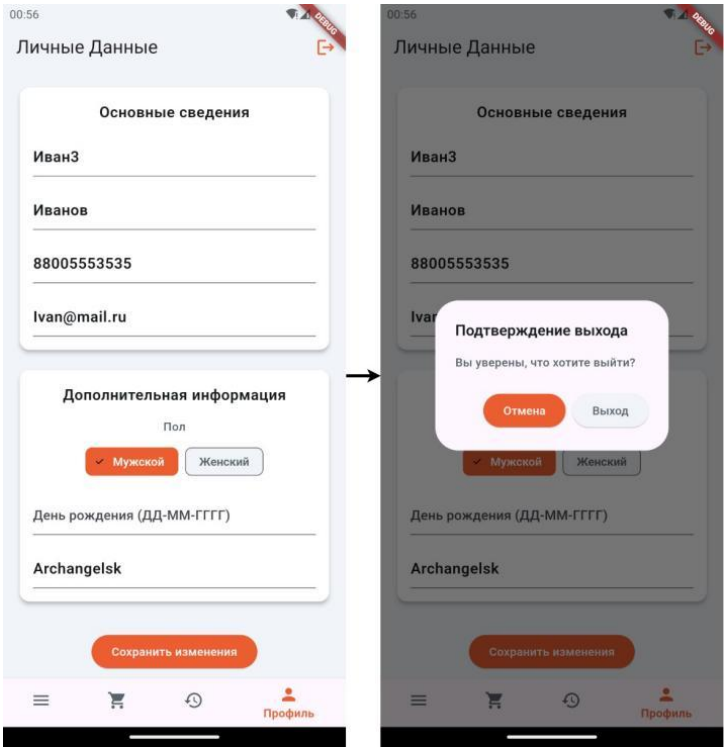

Рисунок 10 – coffee_shop. Процесс выхода из аккаунта

При запуске телеграм-бота при помощи команды «/start», бот установит имя пользователя в телеграм как логин для авторизации и попросит ввести

пароль. Сотруднику необходимо ввести и отправить пароль, ранее выданный администратором. Процесс авторизации изображен на рисунке 11.

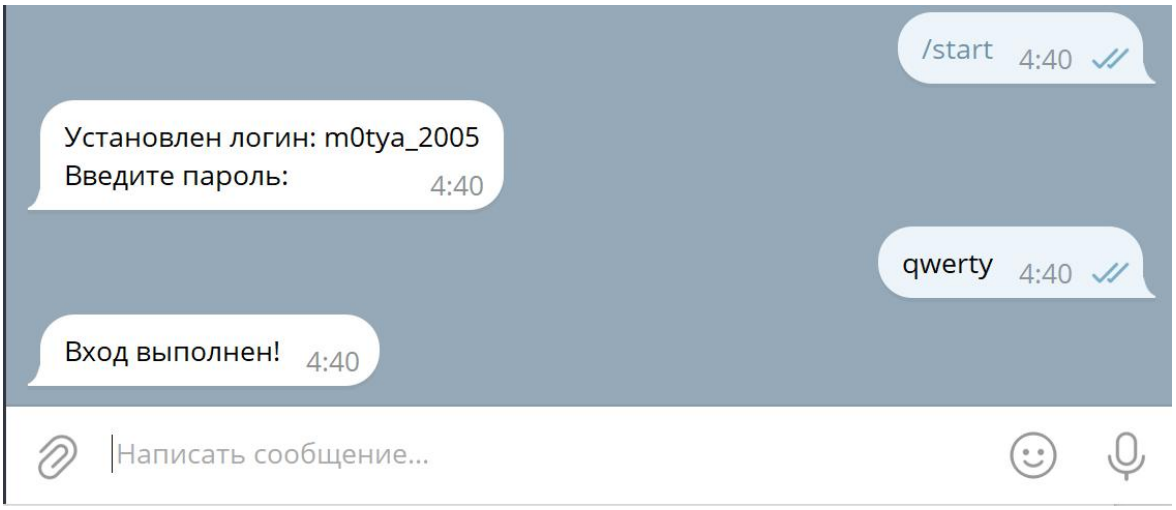

Рисунок 11 – Телеграм-бот. Процесс авторизации сотрудника

В зависимости от заранее предустановленной настройки ролей происходит вход в систему от имени работника или от имени администратора.

После успешной авторизации, сотруднику доступны следующие команды: /address Выбрать кофейню, /work Встать в работу, /home Выйти с работы, /complete Список заказов на выдачу.

При выборе команды /address, сотрудник может выбрать адрес на котором будет производить обработку заказов.

При выборе команды /work, сотрудник автоматически начнет получать новые заказы в режиме реального времени.

При выборе команды /home, сотрудник перестанет получать новые заказы.

При выборе команды /complete, сотрудник получит список готовых заказов, которые можно выдать.

Примеры выполнения команд изображены на рисунке 12.

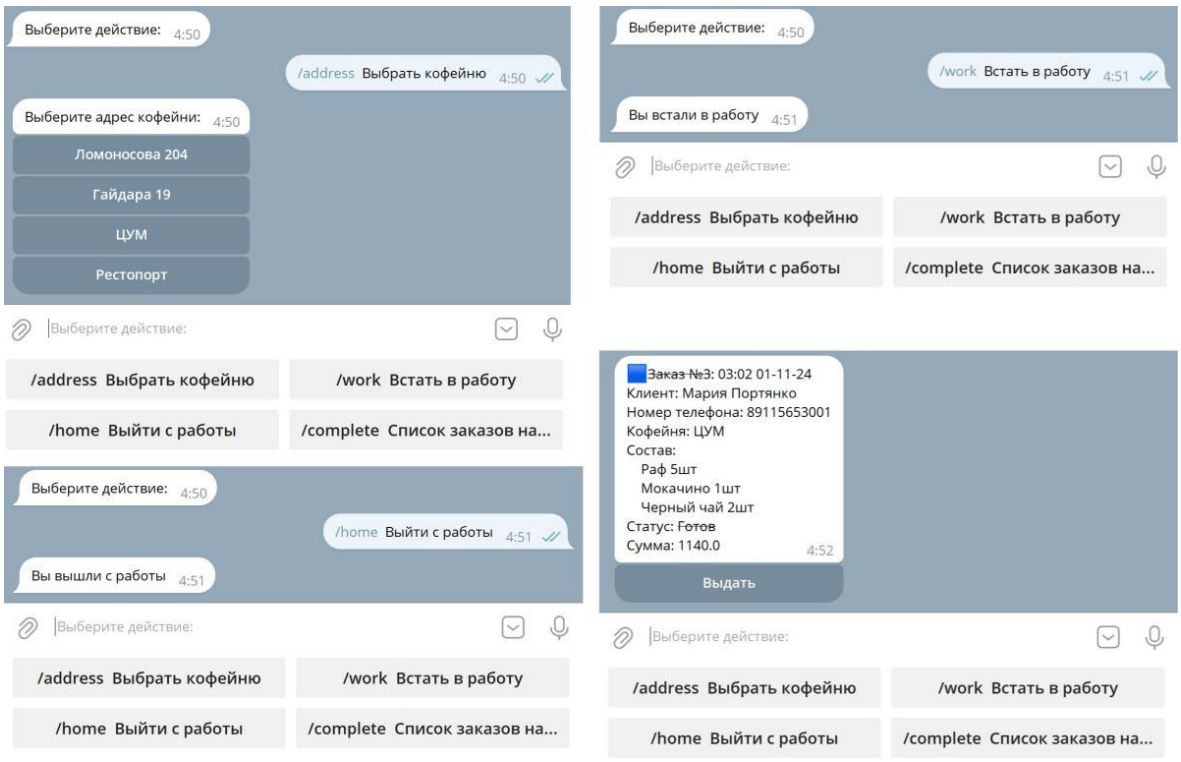

Рисунок 12 – Телеграм-бот. Примеры выполнения команд

Сотрудник может обрабатывать новые заказы при помощи нажатия кнопки «Обновить статус» под сообщением о заказе. При нажатии на эту кнопку статус заказа изменится на «В работе», старое сообщение удалится и появится новое с измененным статусом и внешним видом. При повторном нажатии на кнопку, статус обновится на «Готов» и отобразить этот заказ можно помощью команды /complete. Изображения сообщений о заказах и их статусах на рисунке 13.

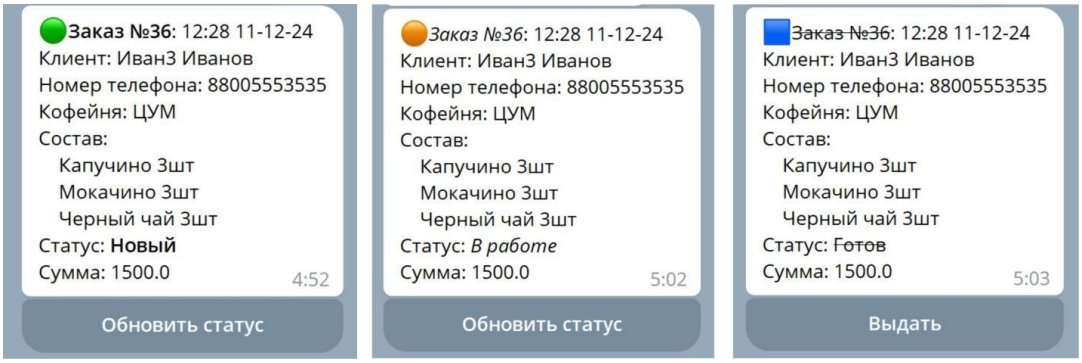

Рисунок 13 – Телеграм-бот. Изображения заказов и их статусов

После успешной авторизации, администратору доступны следующие команды: /newWorker Добавить работника, /readWorkers Получить список всех работников, /delWorker Удалить работника.

При выборе команды /newWorker, администратор может создать нового работника, путем ввода требуемых ботом текстов. Пример выполнения команды /newWorker на рисунке 14.

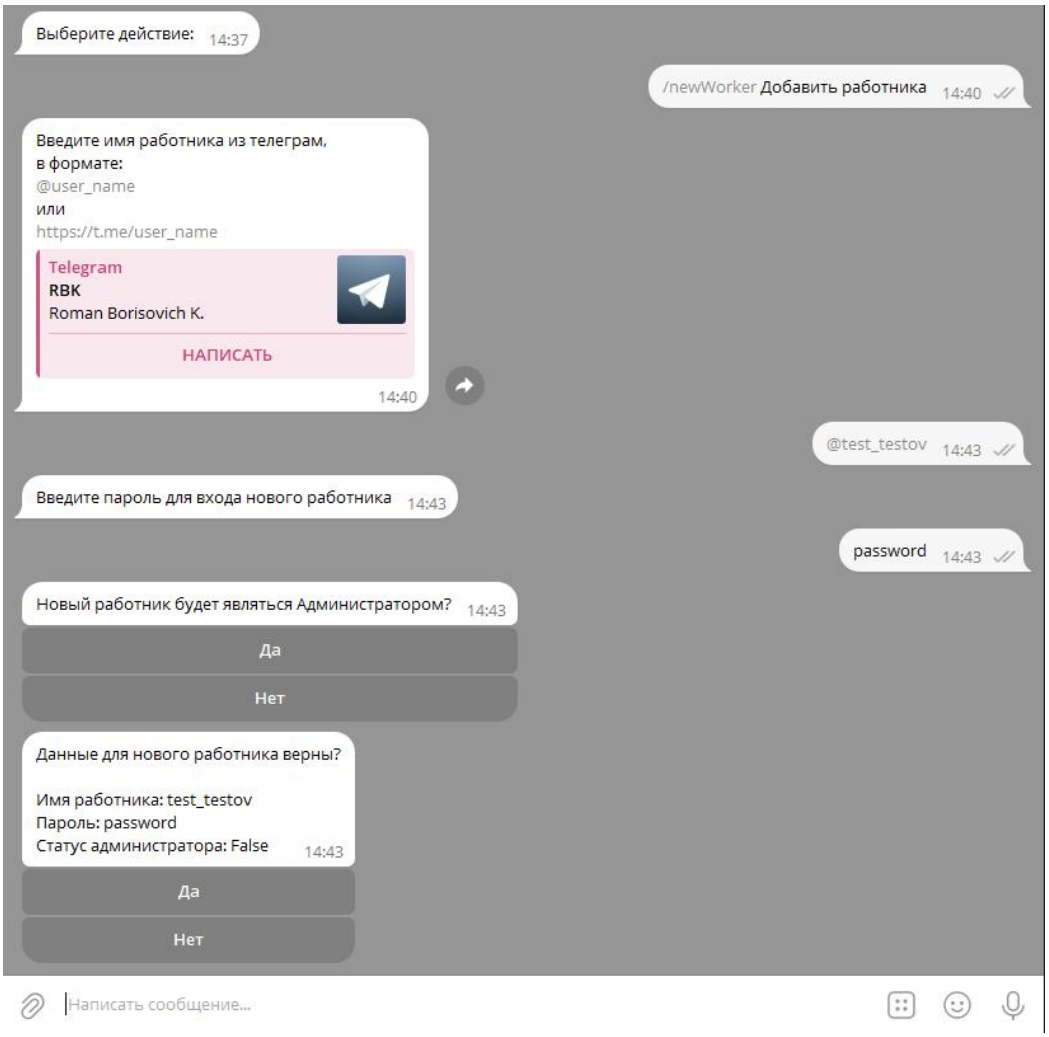

Рисунок 14 – Телеграм-бот. Выполнение команды newWorker

При выборе команды /readWorkers, администратор получит список работников с их данными.

При выборе команды /delWorker, администратор может выбрать какого работника удалить.

Примеры выполнения команд delWorker и readWorkers изображены на рисунке 15.

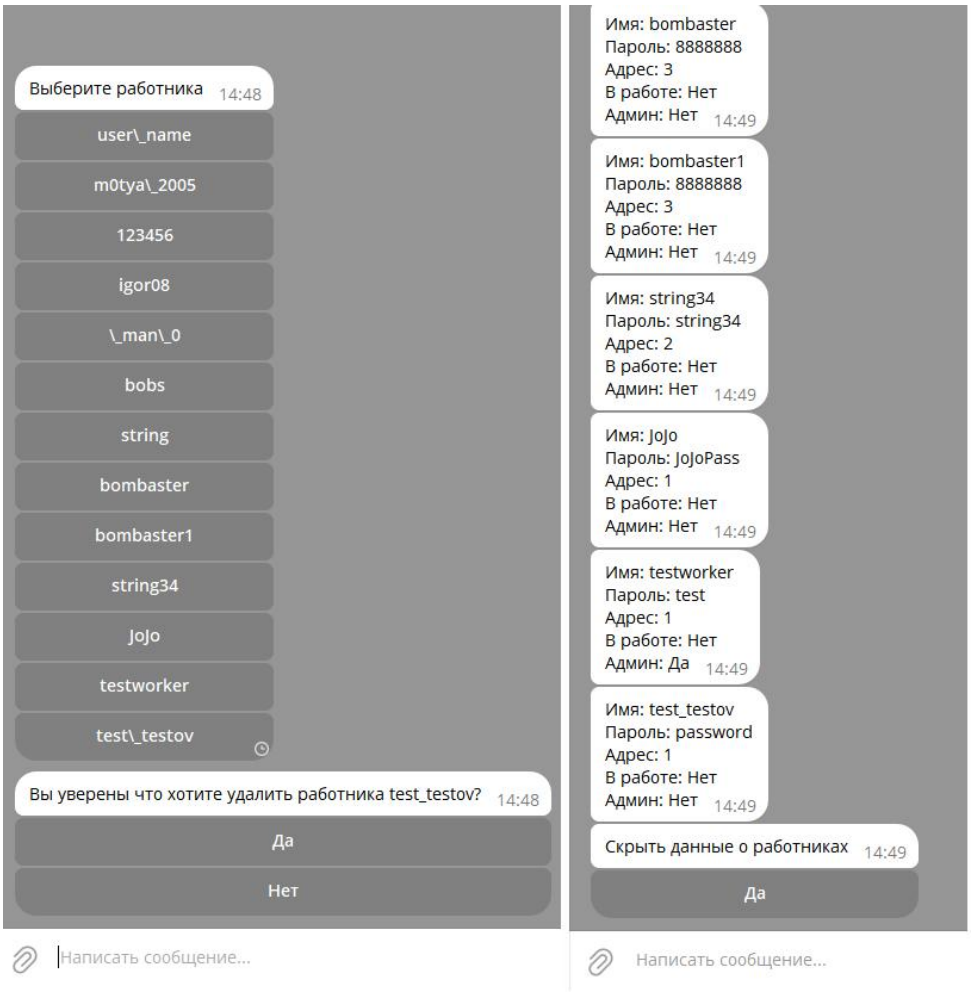

Рисунок 15 – Телеграм-бот. Выполнения команд delWorker и readWorkers

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования достигнута поставленная цель: разработана подсистема «Кофейня: Заказы».

Разработанная подсистема поможет обеспечить возможность быстрого и удобного оформления заказов, а также управления ими в реальном времени. Кроме того, решены все поставленные задачи:

- выполнен сбор и анализ требований целевой аудитории,
- проанализированы информационные источники по предметной области,
- изучены существующие решения в области автоматизации заказов в кофейнях,
- спроектирована архитектура подсистемы,
- спроектирована диаграмма вариантов использования подсистемы,
- выбран состав программных и технических средств для реализации проекта,
- спроектирована физическая схема БД,
- разработано API для взаимодействия мобильного приложения и телеграм-бота с БД,
- реализовано разграничение прав доступа пользователей,
- реализована защита данных,
- разработано мобильное приложение,
- разработан телеграм-бот,
- выполнено структурное тестирование ПО,
- выполнено функциональное тестирование ПО,
- разработана программная документация,
- разработана эксплуатационная документация.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Голицына, О. Л. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва : ФОРУМ : ИНФРА-М, 2023. – 400 с. – (Высшее образование: Бакалавриат). – URL: https://znanium.com/catalog/product/1937956 (дата обращения: 06.11.2024). – Режим доступа: по подписке. – Текст : электронный.

2. Прохоренок, Н. А. Python 3 и PyQt 6. Разработка приложений. – (Профессиональное программирование) / Н. А. Прохоренок, В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2023. – 832 с. – URL: https://ibooks.ru/bookshelf/386513/reading (дата обращения: 09.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

3. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. — Москва : КУРС : ИНФРА-М, 2023. — 336 с. – URL: https://znanium.com/catalog/product/1896457 (дата обращения: 10.11.2024). – Режим доступа: по подписке. – Текст : электронный.

4. Фленов, М. Е. Библия C#. – 5-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2022. – 464 с. – URL: https://ibooks.ru/bookshelf/380047/reading (дата обращения: 08.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

5. Хорев, П. Б. Объектно-ориентированное программирование с примерами на C#. – Москва : Форум, 2020. – 200 с. – URL: https://ibooks.ru/bookshelf/361450/reading (дата обращения: 11.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.