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 МОБИЛЬНОГО ПРИЛОЖЕНИЯ

«ПОДБОР ПЕРСОНАЛА ДЛЯ ВЫПОЛНЕНИЯ

ЗАДАЧ»

Л109. 24КП01. 020 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-11		05.12.2024	В.С Стрельцов
	(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

ПО – программное обеспечение

API – программный интерфейс

ERD – диаграмма «сущность-связь»

HTTP – протокол передачи гипертекста

IDE – интегрированная среда разработки

JSON – объектная нотация JavaScript

REST – репрезентативная передача состояния

ВВЕДЕНИЕ

Актуальность разрабатываемого проекта заключается в том, что он предоставит решение актуальной проблемы в области фриланса - неэффективного взаимодействия между заказчиками и соискателям.

Современный мир труда претерпевает значительные изменения. Рост популярности удаленной работы и фриланса формирует новую реальность, где заказчики ищут квалифицированных специалистов, а соискатели ищут интересные проекты.

По причине неэффективного взаимодействия между заказчиками и соискателями возникает необходимость в более эффективных способах коммуникации с потенциальными кандидатами. Разработка мобильного приложения позволит значительно упростить процесс поиска заказчиков и сделает его более эффективным, а также улучшит коммуникацию между заказчиками и соискателями. В результате пользователи смогут быстрее находить подходящих кандидатов для выполнения своих задач, что приведет к более успешному завершению проектов и повышению общей удовлетворенности всех участников процесса. Таким образом, создание мобильного приложения направлено на оптимизацию взаимодействия между заказчиками и соискателями, что, в свою очередь, будет способствовать более эффективному решению поставленных задач.

Целью курсового проектирования является разработка мобильного приложения для подбора персонала на выполнение задач, которое позволит просматривать, редактировать и размещать задачи, находить специалистов по областям, вести диалог с заказчиками и экспертами.

Для достижения поставленной цели требуется решить следующие задачи:

- выполнить сбор требований целевой аудитории,
- проанализировать информационные источники по предметной области,

- спроектировать архитектуру приложения,
- спроектировать диаграмму вариантов использования приложения,
- выбрать состав программных и технических средств для реализации мобильного приложения,
- спроектировать БД,
- создать БД в PostgreSQL,
- разработать API для мобильного приложения,
- реализовать разграничение прав доступа пользователей,
- реализовать защиту данных,
- разработать интерфейс мобильного приложения,
- разработать мобильное приложение,
- реализовать работу мобильного приложения с сервером БД при помощи REST API,
- выполнить структурное тестирование ПО,
- выполнить функциональное тестирование ПО,
- разработать программную и эксплуатационную документацию.

В результате выполнения поставленных задач будет разработано мобильное приложения для подбора персонала.

1 Анализ и разработка требований

1.1 Назначение и область применения

Разрабатываемое мобильное приложение предназначено для физических и юридических лиц, желающих оптимизировать процесс поиска экспертов для решения поставленных задач. Мобильное приложение упростит процесс поиска экспертов для решения задач, просмотра информации об экспертах, позволит создавать задачи, редактировать информацию о своих задачах, а также приглашать экспертов для их решения.

1.2 Постановка задачи

Необходимо разработать мобильное приложение, которое предоставит доступ к следующей функциональности:

- авторизации,
- регистрации,
- просмотру информации об эксперте,
- просмотру и фильтрации информации об опыте эксперта,
- созданию и редактированию информации о своих задачах,
- приглашению экспертов к выполнению задачи,
- общению с экспертами,
- отправке файлов и изображений в общении с экспертами,
- выставлению оценки экспертам после закрытия задачи.

1.3 Описание алгоритма функционирования системы

При запуске приложения отображается начальная страница, на которой присутствует описание возможностей мобильного приложения, в верхней панели приложения находится кнопка аутентификации, после нажатия на

кнопку открывается модальное окно аутентификации с возможностью выбора роли заказчика и эксперта.

Эксперт может просматривать, откликаться на задачи, вести диалоги с работодателями, просматривать профиль работодателя.

Работодатель может создавать задачи, просматривать экспертов по областям специальности, приглашать экспертов на задачи, оставлять отзывы экспертам, вести диалоги с экспертами.

На рисунке 1 изображена диаграмма вариантов использования приложения различными категориями пользователей.

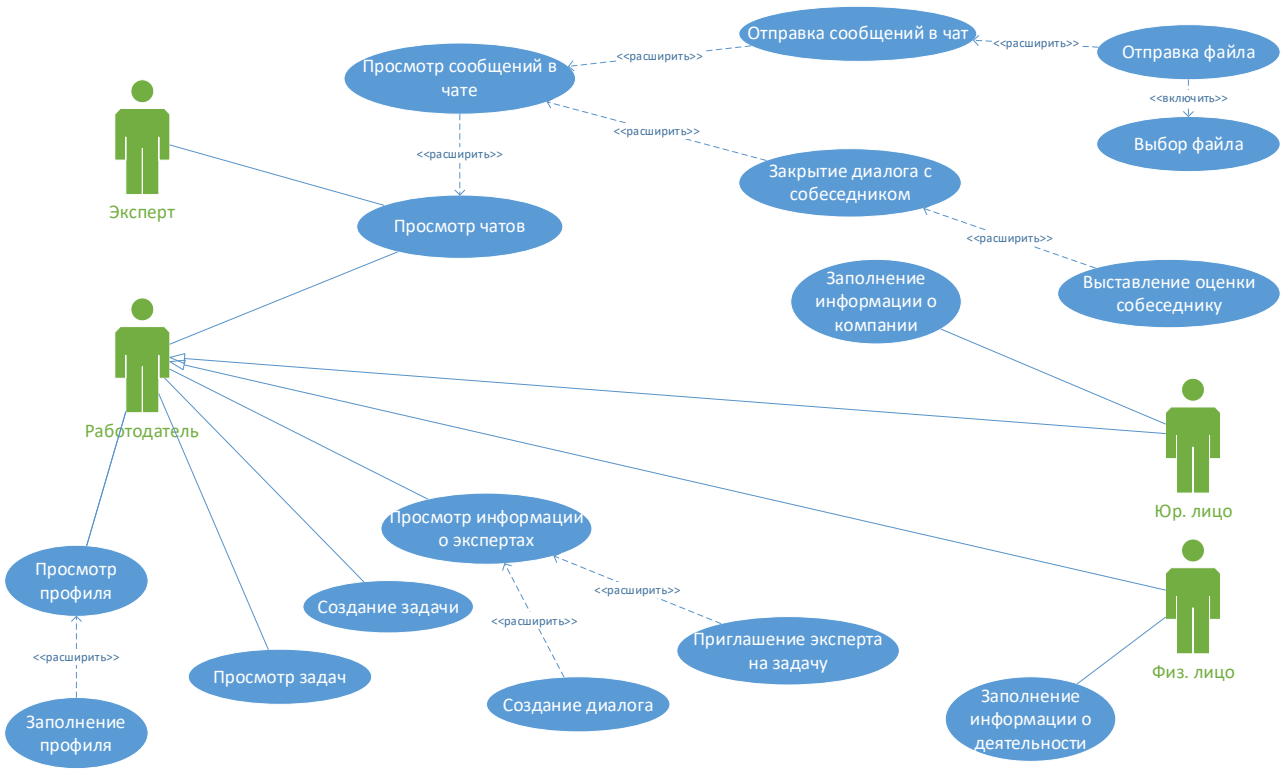

Рисунок 1 – Диаграмма вариантов использования

1.4 Выбор состава программных и технических средств

Согласно цели проекта требуется создать мобильное приложение для подбора персонала на задачу.

В качестве СУБД выбрана PostgreSQL 12, так как она позволяет определять собственные типы данных, функции и операторы, обеспечивает высокую производительность благодаря оптимизации запросов и индексации, поддерживает различные типы данных, включая JSON.

Приложение будет написано на языке программирования Dart с использованием фреймворка Flutter, так как он поддерживает кроссплатформенность, компилируется в нативный код и выдает высокую производительность.

Для разработки приложения будет использоваться IDE Android Studio 2023.1.1, так как она имеет функцию быстрой перезагрузки, встроенные эмуляторы и поддерживает разные языки программирования, включая Dart.

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

- ОС Ubuntu версии 22.10 и выше,
- сервер БД: PostgreSQL не ниже 12.0,
- процессор с частотой 2 ГГц,
- свободная оперативная память объемом 1 ГБ,
- ПО для конфигурирования, управления и администрирования сервера БД: pgAdmin версией не ниже 4,
- ПО для работы API: Python версией не ниже 3.6, Django версией не ниже 3.2, Django REST Framework версией не ниже 3.10.

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

- операционная система Android версией не ниже 11 или iOS версией не ниже 18,
- процессор с частотой 2 ГГц,
- оперативная память в объеме 2 ГБ,
- свободное место в хранилище 200 МБ,
- постоянное интернет-подключение.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В рамках разработки мобильного приложения «Подбор персонала для выполнения задач» создан интерфейс пользователя в виде wireframe и мокапов при помощи сайта Figma. Эти визуальные представления позволяют наглядно увидеть структуру приложения, его основные элементы и функциональность.

Некоторые (страницы «Поиск», «Задачи», «Создание новой задачи») wireframe интерфейса мобильного приложения представлены на рисунке 2.

Часть некоторых (страницы «Поиск», «Задачи», «Создание новой задачи») мокапов интерфейса мобильного приложения представлены на рисунке 3.

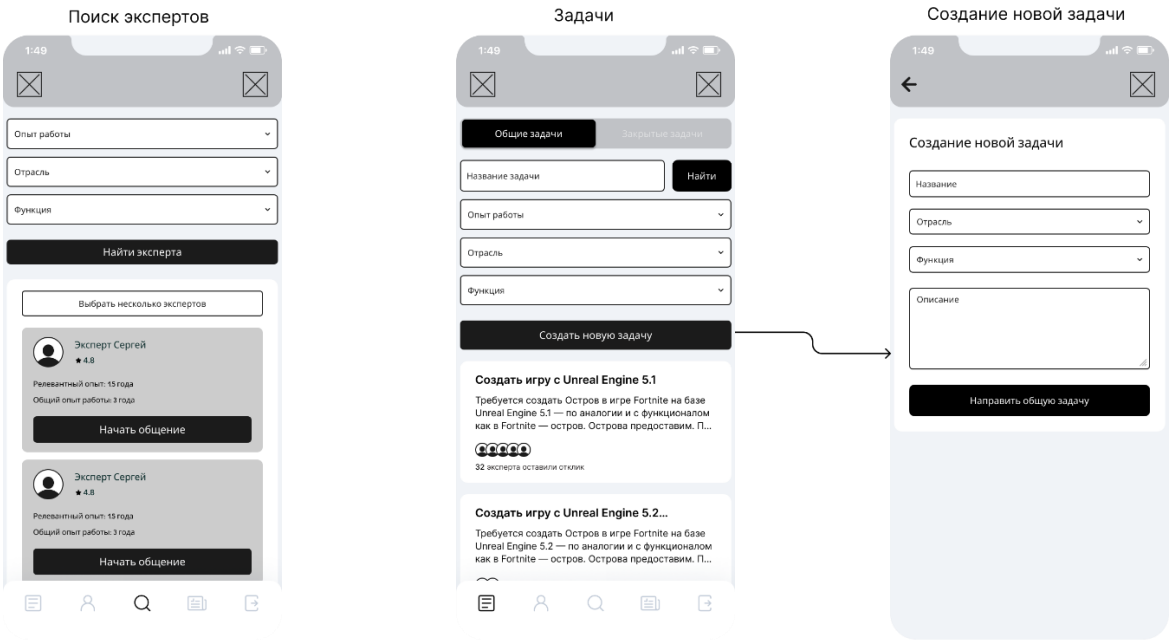

Рисунок 2 – Figma. Wireframe интерфейса пользователя мобильного приложения

Также создана пиктограмма для мобильного приложения, который размещен в верхней панели (представлен на рисунке 3).

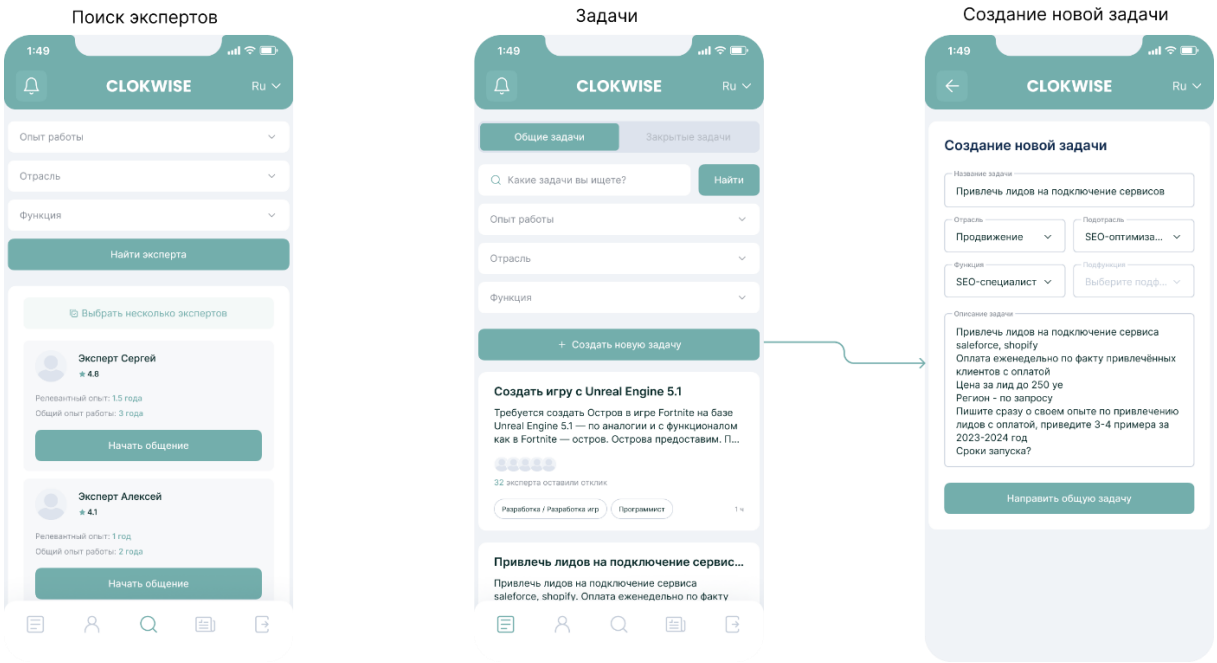

Рисунок 3 – Figma. Мокапы интерфейса пользователя мобильного приложения

2.2 Разработка архитектуры программного обеспечения

Приложение для поиска специалистов предназначено для соединения заказчиков и исполнителей услуг. Архитектура приложения построена на основе клиент-серверной модели и включает в себя несколько ключевых компонентов: серверная часть приложения, мобильное приложение, БД.

Для серверной части будет создан API, позволяющий клиенту взаимодействовать с сервером Диаграмма развертывания компонентов представлена на рисунке 4.

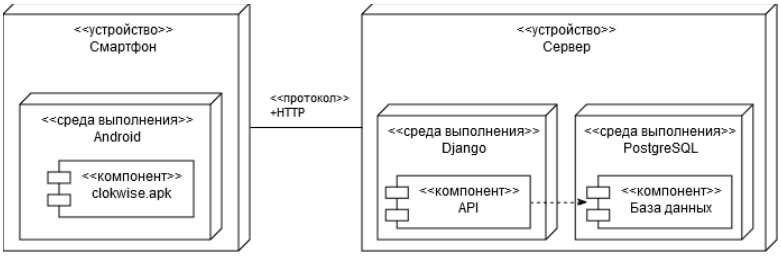

Рисунок 4 – Диаграмма развертывания компонентов

2.3 Проектирование базы данных

В рамках курсового проектирования требуется разработать БД для системы подбора персонала. Система будет использоваться заказчиками в виде физических или юридических лиц [4].

Модели БД созданы при помощи DBDiagram.io. На рисунке 5 в виде ERD показана часть физической модели предметной области, связанной с поиском экспертов для созданной задачи.

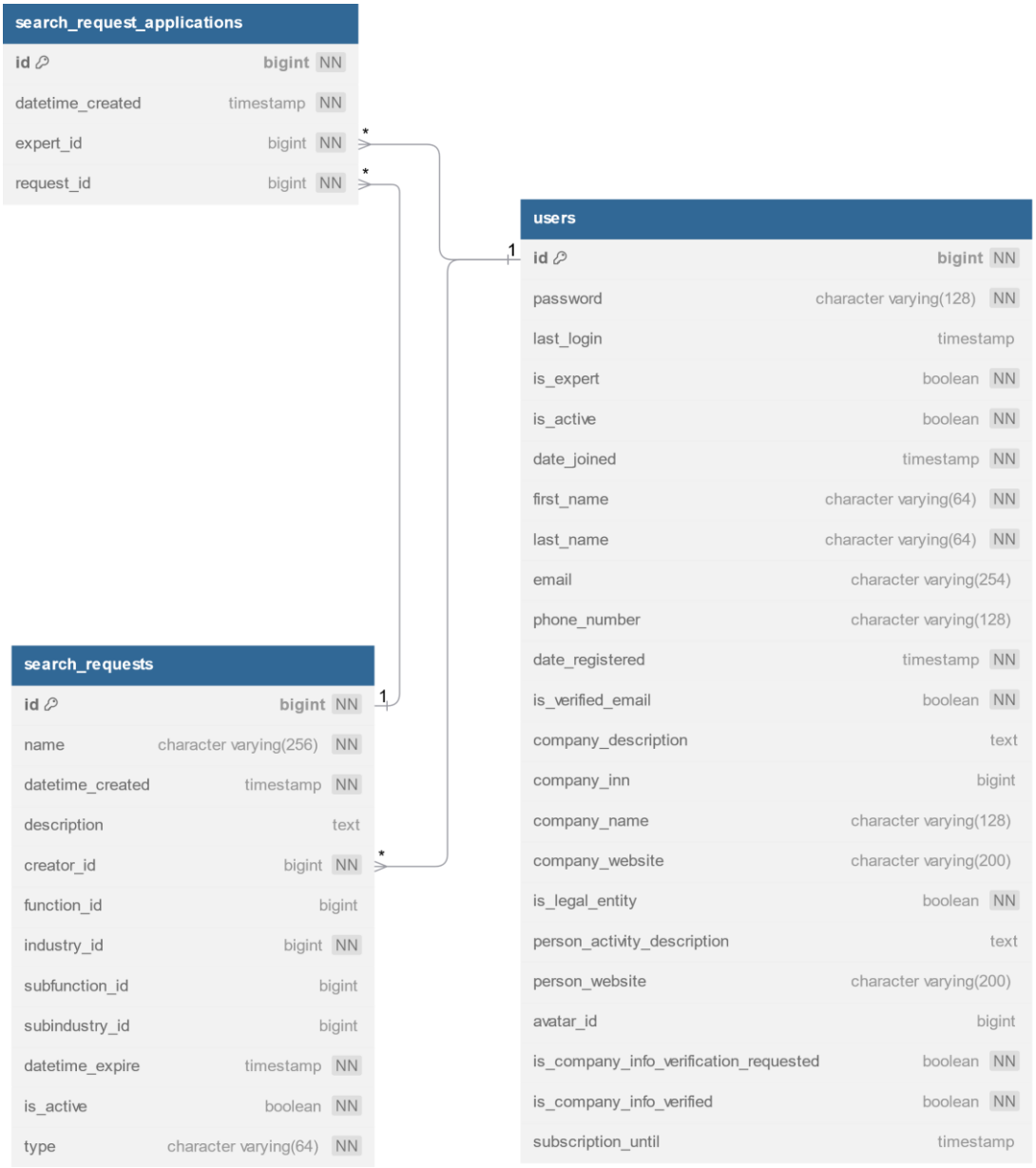

Рисунок 5 – DBDiagram.io. Физическая модель БД (фрагмент)

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

Для курсового проекта разработаны мобильное приложение на Dart с использованием фреймворка Flutter в Android Studio и API на Python с использованием Django REST Framework в качестве ORM использовался Django ORM [2].

Взаимодействие с сервером будет происходить при помощи HTTP-запросов к RESTful API, ответы будут приходить в формате JSON. Для реализации получения данных из БД использован сетевой клиент из плагина Dio для отправления HTTP-запросов на сервер [3]. Код метода репозитория search для получения данных экспертов отправкой POST-запроса на сервер представлен листингом 1.

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

```
/// Отправляет запрос на сервер для получения результатов
/// поиска.
/// Принимает объект [searchRequest], который содержит
параметры поиска.
/// Возвращает список результатов поиска в виде
///[List<ExpertResultModel>].
/// Исключение будет выброшено, если сервер вернет статус-код,
///отличный от 200.
Future<List<ExpertResultModel>> postSearchResult({
  required SearchRequest searchRequest}) async {
  // Выполнение POST-запрос к серверу с данными запроса в
  //формате JSON.
  final response = await _mainNetworkClient.client.post(
    '$_path/',
    data: searchRequest.toJson());
  // Проверка, был ли ответ успешным (статус-код 200).
  if (response.statusCode != 200) {
    throw Exception('Ошибка при выполнении запроса:
${response.statusCode}'); // Выброс исключения при ошибке}
```

```
// Преобразование данных из ответа в список объектов
  final experts = response.data.map<ExpertResultModel>(
(message) => ExpertResultModel.fromJson(message)).toList();
return experts; // Возвращаем список результатов поиска}
```

Код представления в API серверной части для получения профиля пользователем представлен листингом 2.

Листинг 2 – Код представления для профиля пользователем в API

```
class GetClientMainProfileView(APIView):

    """
    Представление API для получения основного профиля клиента
по его идентификатору.    Доступно только для
аутентифицированных пользователей.
    """

    permission_classes = [IsAuthenticated]

   # Ограничение доступа только для аутентифицированных
   # пользователей
    def get(self, request, user_id: int):
        """
        Обработчик GET-запроса для получения основного профиля
клиента.
        Параметры:
        - request: объект запроса.
        - user_id: идентификатор пользователя, чей профиль
нужно получить.
        Возвращает:
        - JSON-ответ с данными профиля и статусом 200 (OK),
если профиль найден.
        - 404 ошибка, если пользователь с указанным
идентификатором не найден.
        """
        # Попытка получить пользователя по его идентификатору
        if user := User.objects.filter(id=user_id).first():
            # Получение типа основного профиля пользователя
            main_profile_type = user.main_profile_type
            # В зависимости от типа профиля, выбираем
            # соответствующий сериализатор
            match main_profile_type:
                case ProfileType.COMPANY:
                    result =
CompanyProfileSerializer(instance=user,
current_user=request.user)
                case ProfileType.PERSON:
```

```
                    result =
PersonProfileSerializer(instance=user,
current_user=request.user)

            # Подготовка данных для ответа
            data = result.data
            # Добавление типа профиля в ответ
            data["profile_type"] = main_profile_type

            # Возвращение ответа с данными профиля и статусом
            # 200
            return Response(data, status=status.HTTP_200_OK)
        else:
            # Если пользователь не найден, возвращается 404
            # ошибку
            raise Http404
```

3.2 Реализация интерфейса пользователя

Интерфейс 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 стэком объектов Route, представляющих страницы в приложении. Для упрощения работы с маршрутами использован пакет go_router.

Для отображения информации об экспертах разработан виджет SearchExpertItem, который представлен рисунком 6.

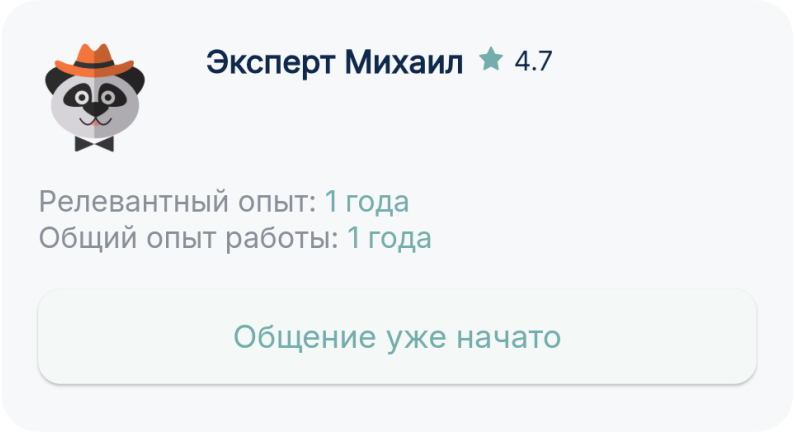

Рисунок 6 – Clokwise. Виджет SearchExpertItem

Код виджета SearchExpertItem представлен листингом 3.

Листинг 3 – Код виджета для SearchExpertItem

```
/// Виджет для отображения информации о эксперте в результате
/// поиска.
class SearchExpertItem extends StatelessWidget {
  final int userId; // Идентификатор пользователя
  final int? chatId; // Идентификатор чата
  final bool isChosen; // Флаг, указывающий, выбран ли эксперт
  final String name; // Имя эксперта
  final String? avatar; // URL аватара эксперта
  final double? rating; // Рейтинг эксперта
  final String relevantExperience; // Релевантный опыт эксперта
  final String totalExperience; // Общий опыт работы эксперта
  final bool isPaidForView; // Флаг, указывающий, оплачен ли
//просмотр
  final VoidCallback onTap; // Обработчик нажатия на элемент

  const SearchExpertItem({
    required this.userId, required this.chatId, required
this.name, required this.relevantExperience, required
this.totalExperience, required this.isChosen, required
this.onTap, required this.avatar, required this.rating,
required this.isPaidForView, super.key});

  @override
  Widget build(BuildContext context) {
    return Column(children: [
      InkWell(
       onTap: onTap,
       child: Ink(
         child: Container(
            width: double.infinity,
            child: Column(
             crossAxisAlignment: CrossAxisAlignment.start,
                 children: [
 // Виджет для отображения аватара и информации об эксперте
                  AvatarSearchWidget(
                   image: avatar,
                   fullName: 'Эксперт $name',
                   rating: rating,
                   isPaidForView: isPaidForView),
                 // Отображение релевантного опыта
                   RichText(
                    text: TextSpan('Релевантный опыт: ',
                    children: [
                     TextSpan(
                       text: relevantExperience),
                        // Отображение общего опыта работы
```

```
                        RichText(text: TextSpan(
                                text: 'Общий опыт работы: ',
                                children: [
                              TextSpan(totalExperience)
                        chatId == null ? CwElevatedButton(
                  text: 'Начать общение',
                  onTap: () { // Запуск события начала чата
                 BlocProvider.of<SearchBloc>(context).
                   add(SearchEvent.startChat(userId: userId));})
                  : CwElevatedButton(
                   text: 'Общение уже начато',
                     onTap: () {
              // Показ детали чата, если чат уже начат
                CustomNavigator.showModal(context: context,
                child: ChatDetailsPage(id: chatId!));}
              )]))))]);}}
```

3.3 Разграничение прав доступа пользователей

В мобильном приложении разработано разграничение прав доступа пользователей при помощи JSON Web Tokens. Для этого в приложении реализованы авторизация и регистрация под ролями «Эксперт» и «Работодатель». Код авторизации в мобильном приложении представлен в листинге 4.

Листинг 4 – Код авторизации

```
// Запрос к серверу для авторизации
    final response = await _mainNetworkClient.client.post(
      'login/', data: model.toJson());
    //Проверка статус кода
    if (response.statusCode != 200) {
      String responseError = response.data['error'].toString();
      throw Exception(responseError);
    }

    final loginModel = LoginModel.fromJson(response.data);
    //Добавление токенов в хранилище
    await _secureStorageManager.setAuthToken(token:
loginModel.access);
    await _secureStorageManager.setRefreshToken(token:
loginModel.refresh);
```

4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

Во время курсового проектирования проведено структурное тестирование для search_repository [1]. Для него использована библиотека mockito для создания имитации поведение зависимостей для репозитория и библиотека flutter_test. Код unit-теста для поиска экспертов представлен листингом 3.

Листинг 6 – Код unit-теста для search_repository

```
void main() {
  // Объявление моков для зависимостей
  late MockSearchRepository mockSearchRepository;
  late MockSecureStorageManager mockSecureStorageManager;
  late MockMainNetworkClient mockMainNetworkClient;
  // Инициализация тестовой среды Flutter
  TestWidgetsFlutterBinding.ensureInitialized();

  setUp(() {
    // Создание экземпляры моков для использования в тестах
    mockSecureStorageManager = MockSecureStorageManager();
    mockMainNetworkClient = MockMainNetworkClient();
    mockSearchRepository = MockSearchRepository();
  });

  group('SearchRepository', () {
    test('postSearchResult returns list of experts on success',
() async {
      // Тест для успешного получения списка экспертов
      final searchRequest = SearchRequest(industry: 1,
experience: 0);
      final result = await
mockSearchRepository.postSearchResultMock(searchRequest);

      expect(result, isA<List<ExpertResultModel>>());
      expect(result.length, 2);
      expect(result[0].id, 1);
      expect(result[0].firstName, 'Expert 1');
      expect(result[1].id, 2);
      expect(result[1].firstName, 'Expert 2');
    });
```

```
// Тест для обработки исключений при ошибке
    test('postSearchResult throws exception on error', () async
{
      final searchRequest = SearchRequest(prompt: 'invalid',
industry: 1, experience: 0);
      try {
        // Использование метод мока для выбрасывания исключения
        await
mockSearchRepository.throwExceptionOnPostSearchResult(searchReq
uest);
      } catch (e) {
        // Вывод информации об исключении
        print('Caught an exception: $e');
        // Проверка, что исключение действительно произошло
        expect(e, isA<Exception>());
      }
    });
  });
}
```

На рисунке 8 изображена консоль с результатами тестирования, где отображается информация о перехваченных исключениях и успешном тестировании.

```
PS C:\Users\Vlad\StudioProjects\clokwise> flutter test test/search_repository_test.dart
00:07 +1: SearchRepository postSearchResult throws exception on error
Caught an exception: Exception: Error occurred
00:07 +2: All tests passed!
```

Рисунок 8 – Android Studio. Результаты unit-тестирования

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование при помощи виджет-тестирования из библиотеки integration_test.

Листинг 7 – Код интеграционного тестирования для страницы с авторизацией

```
void main() {
```

```
  // Инициализация интеграционного тестирования.
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();
  /// Тест для проверки функциональности входа в систему.
  ///
  /// Этот тест выполняет следующие шаги:
  /// 1. Запускает приложение.
  /// 2. Находит элементы интерфейса для ввода имени
пользователя и пароля.
  /// 3. Вводит тестовые данные (имя пользователя и пароль).
  /// 4. Нажимает кнопку входа.
  /// 5. Проверяет, что после успешного входа отображается имя
пользователя.
  testWidgets("Login Test", (WidgetTester tester) async {
    // Запускаем основное приложение.
    app.main();
    // Дожидаемся завершения анимаций и обновления интерфейса.
    await tester.pumpAndSettle();

    // Находим кнопку входа по ключу.
    final menuButton = find.byKey(ValueKey('enterButton'));
    // Проверяем, что кнопка входа отображается на экране.
    expect(menuButton, findsOneWidget);

    // Находим поле для ввода имени пользователя и пароля по
ключам.
    final usernameField =
find.byKey(ValueKey('emailTextField'));
    final passwordField =
find.byKey(ValueKey('passwordTextField'));

    // Вводим тестовый адрес электронной почты.
    await tester.enterText(usernameField,
'budrefurta@gufum.com');
    await tester.pumpAndSettle(); // Обновляем интерфейс.

    // Вводим тестовый пароль.
    await tester.enterText(passwordField, 'test123456');
    await tester.pumpAndSettle(); // Обновляем интерфейс.

    // Находим кнопку входа по ключу и нажимаем на нее.
    final enterButton = find.byKey(ValueKey('enterButton'));
    await tester.tap(enterButton);

    await tester.pumpAndSettle(); // Дожидаемся завершения
анимаций.

    // Проверяем, что после успешного входа отображается имя
пользователя.
    expect(find.text('Михаилл'), findsOneWidget);
  });
}
```

5 Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

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

- операционная система –Ubuntu версией не ниже 22.10,
- процессор – 2 ГГц,
- оперативная память – 2 ГБ,
- свободное место на диске – 10 ГБ,
- дополнительные компоненты: Python 3.6 (или выше), PostgreSQL 12 (или выше), Django REST Framework 3.10 (или выше).

Добавление объектов БД происходит через миграции от Django REST Framework.

Для установки серверной части требуется перейти в терминале в корневую папку API и в терминале использовать команду python manage.py runserver.

Для функционирования мобильного приложения достаточны следующие минимальные программные и технические средства:

- операционная система – Android 11 (или выше) или iOS 18 (или выше),
- процессор – 2 ГГц,
- оперативная память – 2 ГБ,
- свободное место в хранилище – 200 МБ,
- дополнительное – постоянное интернет-подключение.

Для установки мобильного приложения требуется выбрать собранный файл clokwise.apk в проводнике и установить его.

В мобильном приложении используются следующие данные для авторизации:

- электронная почта – budrefurta@gufum.com,

- пароль – test123456.

5.2 Инструкция по работе

При запуске приложения, пользователя встречает начальное окно с общей информацией о приложении. Для авторизации требуется нажать на иконку логина в левом верхнем углу страницы, ввести данные для авторизации и выбрать под какой ролью она происходит в появившемся модальном окне, представленном на рисунке 9.

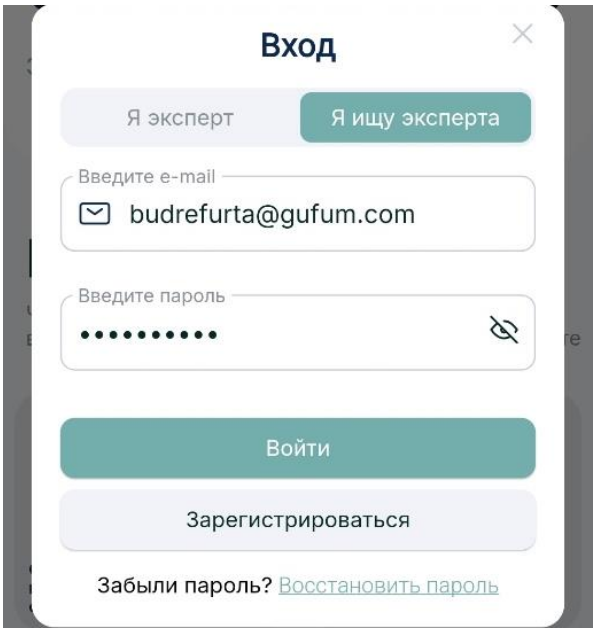

Рисунок 9 – Clokwise. Вид модального окна «Авторизация»

После авторизации пользователь перенаправляется на экран своего профиля. Пользователю в роли «Работодатель» доступны экраны чатов, поиск экспертов и созданные задачи, к которым можно получить доступ через навигационную панель внизу экрана. Страница «Профиль» представлена на рисунке 10.

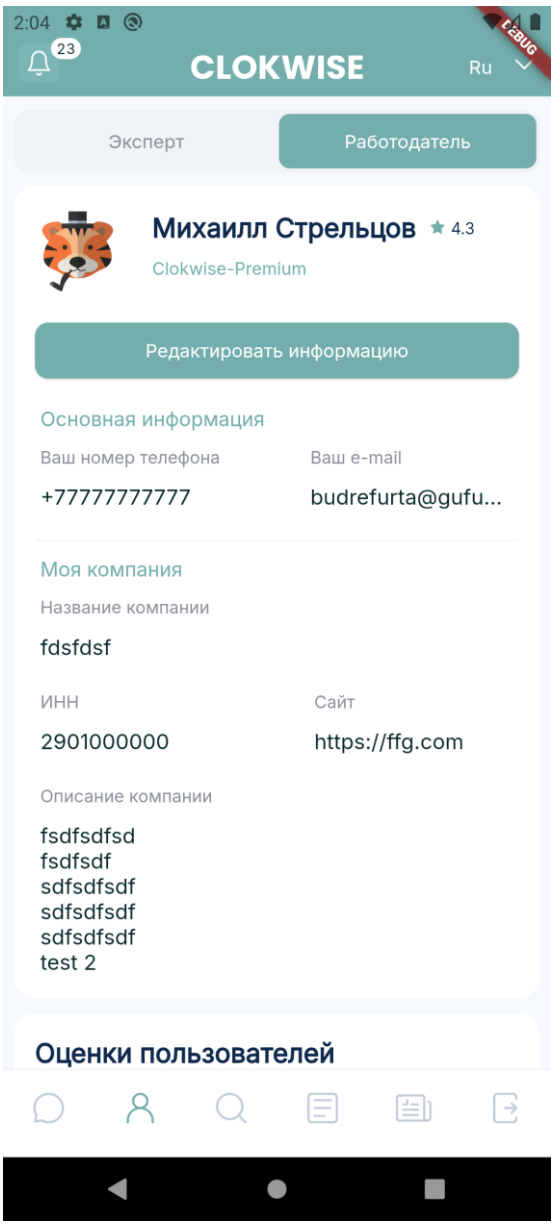

Рисунок 10 – Clokwise. Вид страницы «Профиль»

Для поиска эксперта требуется перейти на страницу «Поиск экспертов» при помощи навигационной панели, нажав на кнопку с иконкой лупы. После этого требуется выбрать требуемые опыт, отрасль, подотрасль, функцию и подфункцию, а также, если нужно, ввести слово для фильтрации. Затем требуется нажать на кнопку «Найти эксперта». Пример заполненной страницы для поиска экспертов представлен на рисунке 11.

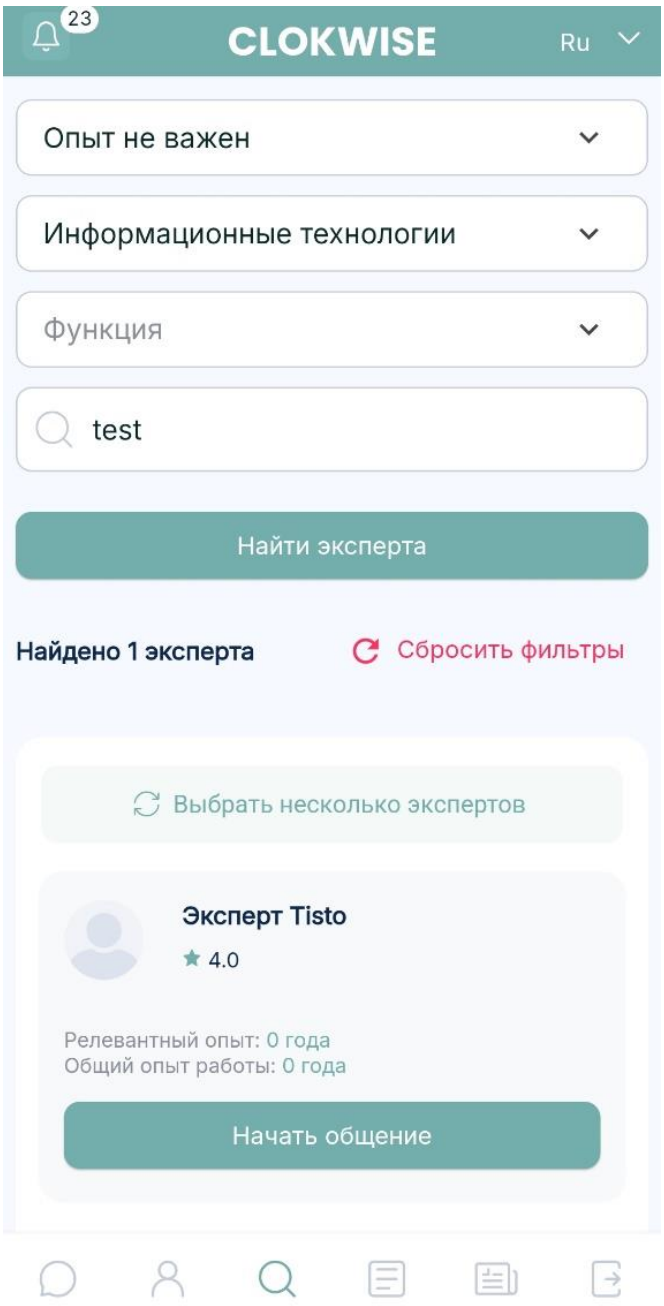

Рисунок 11 – Clokwise. Вид страницы «Поиск эксперта»

При нажатии на пользователя в списке откроется его профиль. Чтобы пригласить эксперта на задачу, необходимо нажать на кнопку «Пригласить на задачу» и выбрать задачу в появившемся модальном окне, представленном на рисунке 12.

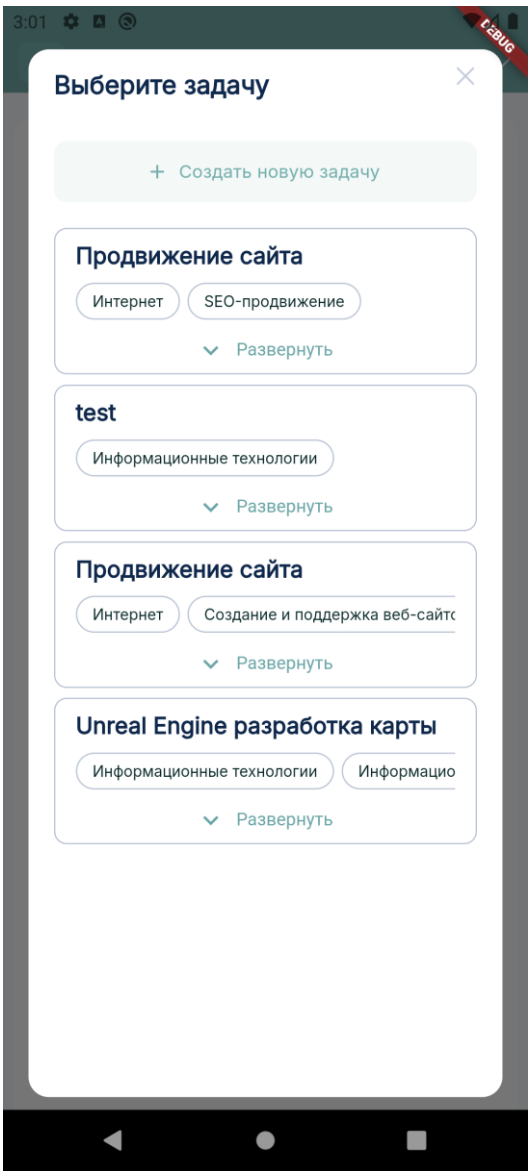

Рисунок 12 – Clokwise. Вид модального окна «Выбор задачи»

В системе существуют две категории задач. Общие задачи доступны для всех экспертов, позволяя им просматривать и откликаться на них самостоятельно. Закрытые задачи требуют приглашения от работодателя. Эксперты могут получить доступ к таким задачам только в том случае, если их пригласит работодатель. Для просмотра всех задач, нужно перейти на страницу «Задачи», нажав на кнопку с иконкой карточки. Пример страницы «Задачи» представлен на рисунке 13.

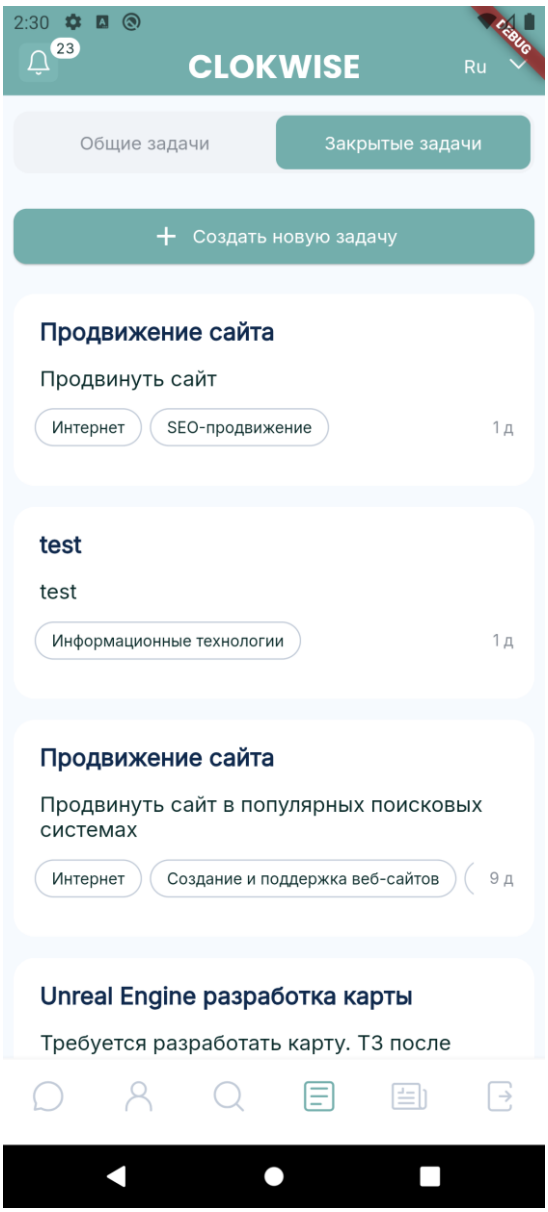

Рисунок 13 – Clokwise. Вид страницы «Задачи»

Для создания задачи требуется нажать на кнопку «Создать новую задачу». В открытой странице требуется заполнить данные для создания задачи: название задачи, отрасль, подотрасль, функция, подфункция и описание задачи. После чего нажать на кнопку «Направить запрос».

Пример заполненной страницы «Создание новой задачи» представлен на рисунке 14.

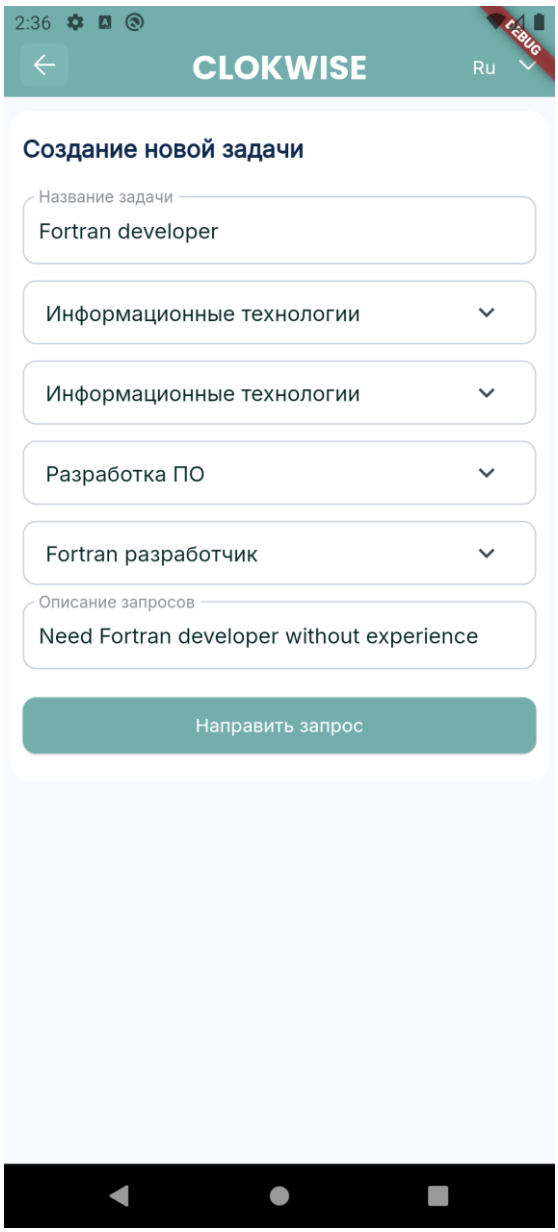

Рисунок 14 – Clokwise. Вид страницы «Создание новой задачи»

Для перехода на страницу с чатами требуется нажать на навигационной панели на кнопку с иконкой облачка сообщения. На странице доступны чат с поддержкой и чаты с заказчиками, которые сортируются по последнему сообщению. Чат с галочкой и временем означает, что он прочитан.

Пример страницы «Чаты» представлен на рисунке 15.

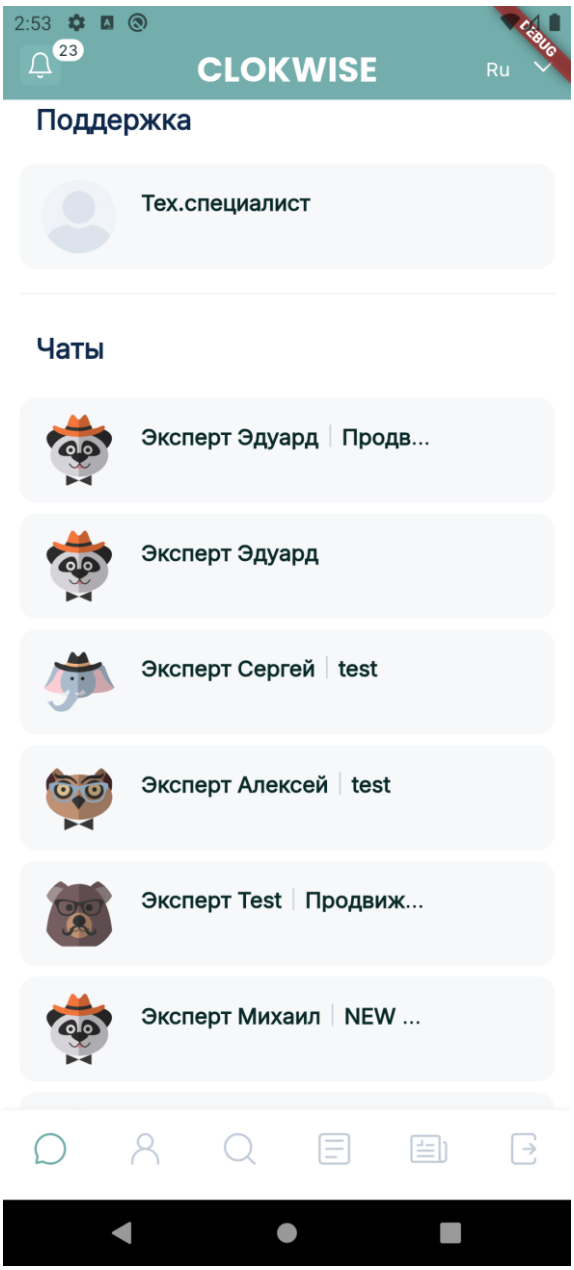

Рисунок 15 – Clokwise. Вид страницы «Чаты»

При нажатии на чат откроется переписка с выбранным пользователем. В чате с пользователем можно отправлять сообщения и прикреплять изображения или файлы. Пример страницы «Чат с пользователем» представлен на рисунке 16.

Рисунок 16 – Clokwise. Вид страницы «Чат с пользователем»

Пользователю с ролью «Эксперт» доступны только экраны чатов и своего профиля. Для смены роли требуется зайти на страницу профиля и в верхней части страницы выбрать роль. Переключатель ролей можно увидеть на странице «Профиль», представленный на рисунке 17.

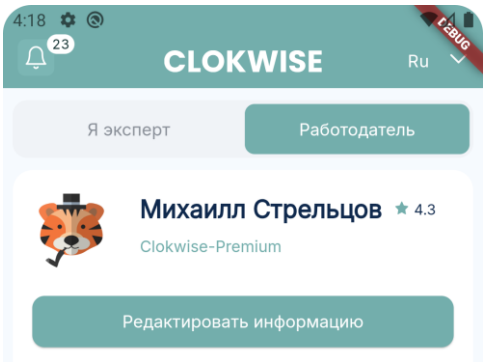

Рисунок 17 – Clokwise. Вид виджета «Переключатель ролей»

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования достигнута поставленная цель: спроектировано и разработано мобильное приложение «Подбор персонала для выполнения задач». Разработанное мобильное приложение поможет в подборе персонала на задачи. Кроме того, решены все поставленные задачи:

- выполнен сбор требований целевой аудитории,
- проанализированы информационные источники по предметной области,
- спроектирована архитектура приложения,
- спроектирована диаграмма вариантов использования приложения,
- выбран состав программных и технических средств, используемых для реализации мобильного приложения,
- спроектирована БД,
- создана БД в PostgreSQL,
- разработано API для мобильного приложения,
- реализовано разграничение прав доступа пользователей,
- реализована защита данных,
- реализована работа мобильного приложения с сервером при помощи REST API,
- выполнено структурное тестирование ПО,
- выполнено функциональное тестирование ПО,
- разработана программная документация,
- разработана эксплуатационная документация.

В результате выполнения поставленных задач было разработано мобильное приложение для подбора персонала, отвечающее современным тенденциям и требованиям заказчика, что позволит значительно улучшить процесс взаимодействия между всеми участниками и повысить эффективность работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading (дата обращения: __.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

2. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2023. – 400 с. – URL: https://znanium.com/catalog/product/1895679 (дата обращения: __.11.2024). – Режим доступа: по подписке. – Текст : электронный.

3. Гивакс, Д. Д. Паттерны проектирования API. – Санкт-Петербург : Питер, 2023. – 512 с. – URL: https://ibooks.ru/bookshelf/390212/reading (дата обращения: __.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

4. Дадян, Э. Г. Данные: хранение и обработка : учебник. – Москва : ИНФРА-М, 2020. – 205 с. – URL: https://znanium.com/catalog/product/1045133 (дата обращения: __.11.2024). – Режим доступа: по подписке. – Текст : электронный.

5. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL- и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. — Москва : ФОРУМ : ИНФРА-М, 2023. — 368 с. - URL: https://znanium.com/catalog/product/1912454 (дата обращения: __.11.2024). – Режим доступа: по подписке. — Текст : электронный.