САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 3

Реализация REST API на основе boilerplate

Выполнила:

Едигарева Дарья

Группа
К3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Реализовать автодокументирование средствами swagger;

Реализовать документацию API средствами Postman.

Ход работы

1. Автодокументирование средствами swagger

Swagger интегрируется в Express-приложение с помощью библиотеки swagger-ui-express. Основой для документации служит OpenAPI-спека, которая автоматически генерируется TSOA-аннотациями. Конфигурация описана в файле tsoa.json: там задаётся путь к контроллерам, включается схема безопасности bearerAuth для JWT и указывается модуль аутентификации. После выполнения команд tsoa spec и tsoa routes формируется файл swagger.json. Этот файл подключается в app.ts через swagger-ui-express, и интерфейс доступен по адресу /docs. В нём отображаются все контроллеры и методы API, включая вход и регистрацию пользователей, личный кабинет соискателя (профиль, резюме, опыт, образование), личный кабинет работодателя (вакансии, отклики), а также поиск и детали вакансий. В файле swagger.ts динамически подставляются метаданные из SETTINGS (название приложения, версия, описание, URL сервера), затем явно добавляется схема безопасности bearerAuth для JWT в spec.components.securitySchemes, и при инициализации UI указываются опции (explorer: true и persistAuthorization: true), чтобы в интерфейсе была кнопка поиска и сохранялся введённый токен.

Файл swagger.ts:

```
import { Application } from 'express';
import swaggerUi from 'swagger-ui-express';
import rawDocument from '../../swagger.json';
import SETTINGS from '../config/settings';


export function useSwagger(app: Application): void {
```

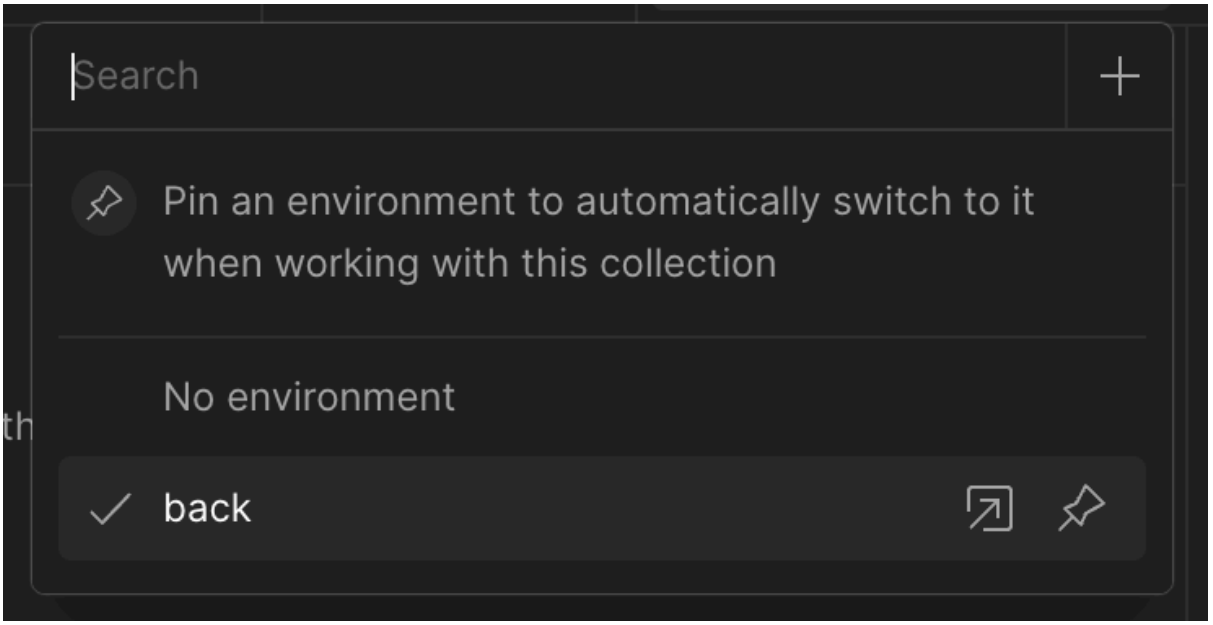
```
                        const        spec        =
JSON.parse(JSON.stringify(rawDocument));
   spec.info = {
       title:       SETTINGS.APP_NAME,
       version:     SETTINGS.APP_VERSION,
       description: SETTINGS.APP_DESCRIPTION,
   };
   spec.servers = [
       {
                                                 url:
`${SETTINGS.APP_PROTOCOL}://${SETTINGS.APP_HOST}:${SE
TTINGS.APP_PORT}${SETTINGS.APP_API_PREFIX}`,
           description: 'API server',
       },
   ];

   spec.components = spec.components || {};

   spec.components.securitySchemes = {
       bearerAuth: {
           type:         'http',
           scheme:       'bearer',
           bearerFormat: 'JWT',
           description:  'Enter: `<token>`',
```

```
        },
        ...spec.components.securitySchemes,
    };

    const uiOptions = {
        explorer: true,
        swaggerOptions: {
            persistAuthorization: true,
        },
    };

                app.use('/docs',      swaggerUi.serve,
swaggerUi.setup(spec, uiOptions));
}
```

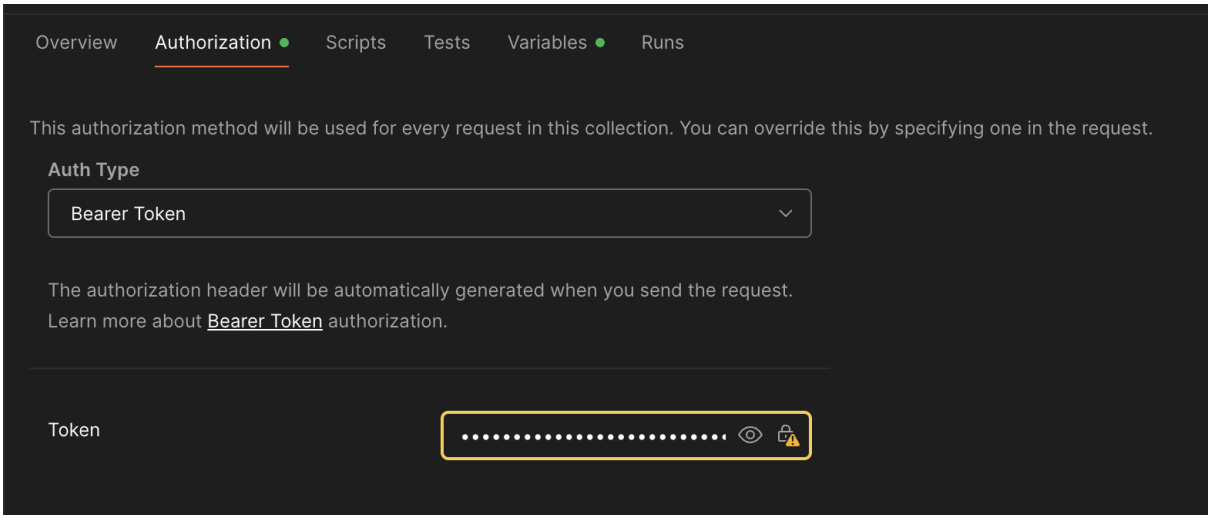
2. Документация API средствами Postman.

Спецификацию OpenAPI, сгенерированную TSOA, я импортировала напрямую, получив готовую коллекцию с методами API. Создано окружение с переменными baseUrl (http://localhost:8000/api) и token, чтобы все запросы шли по правильному адресу и автоматически подставляли JWT после входа. Коллекция отражает все реализованные маршруты и позволяет выполнять CRUD-операции с профилем соискателя, резюме, профилем работодателя, вакансиями и откликами.

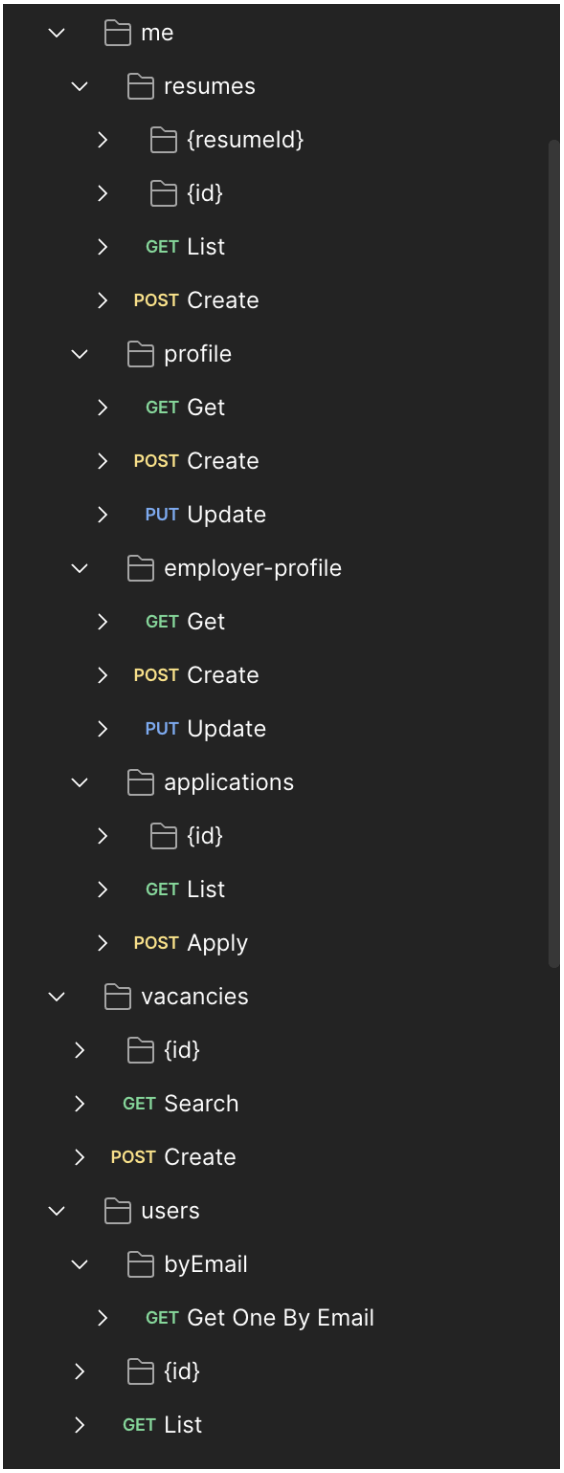
Настройка окружения:



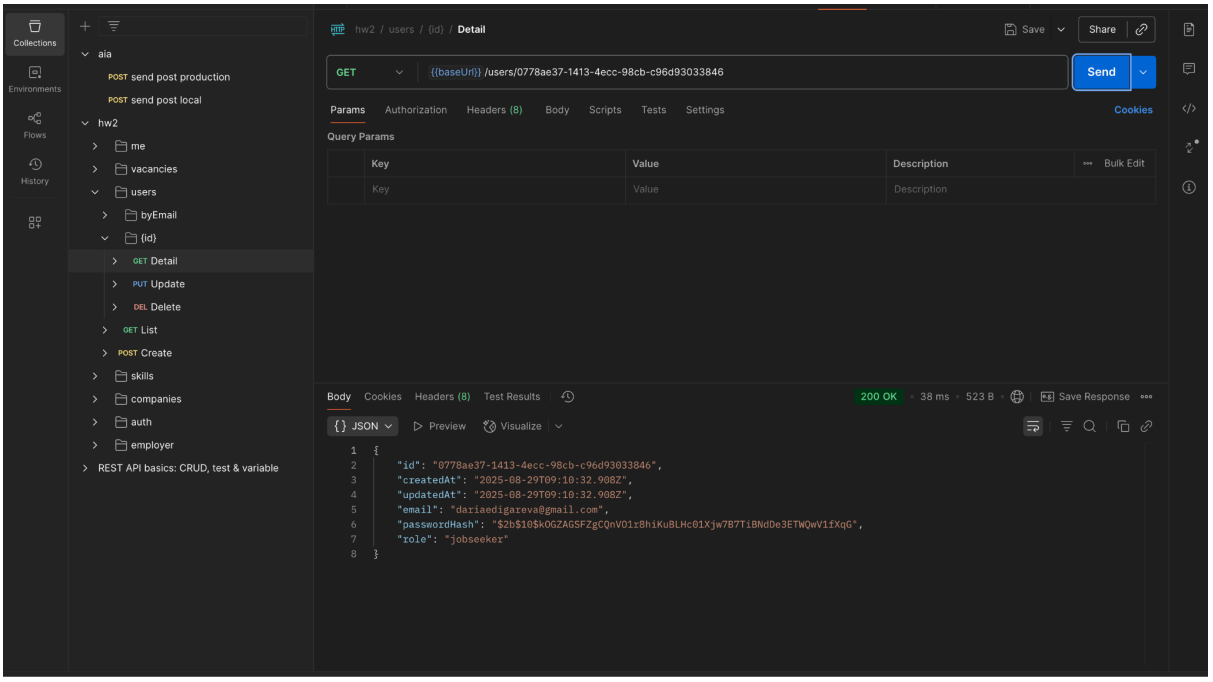
Авторизация Bearer Token:

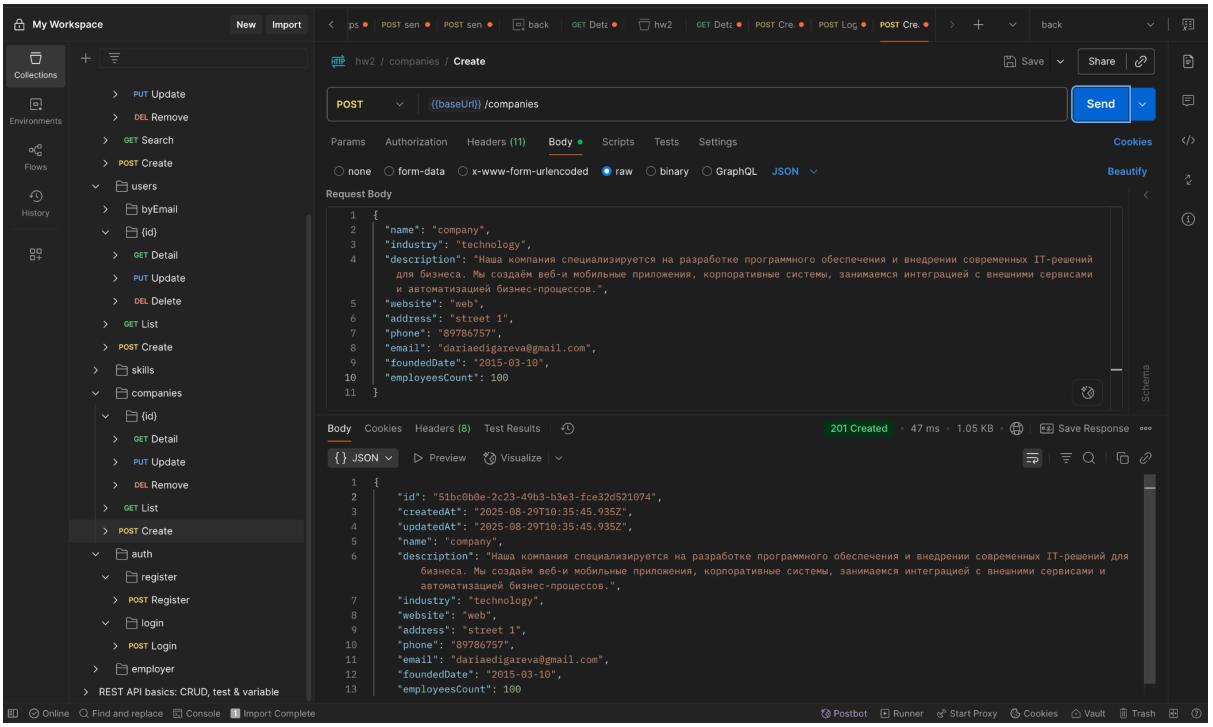


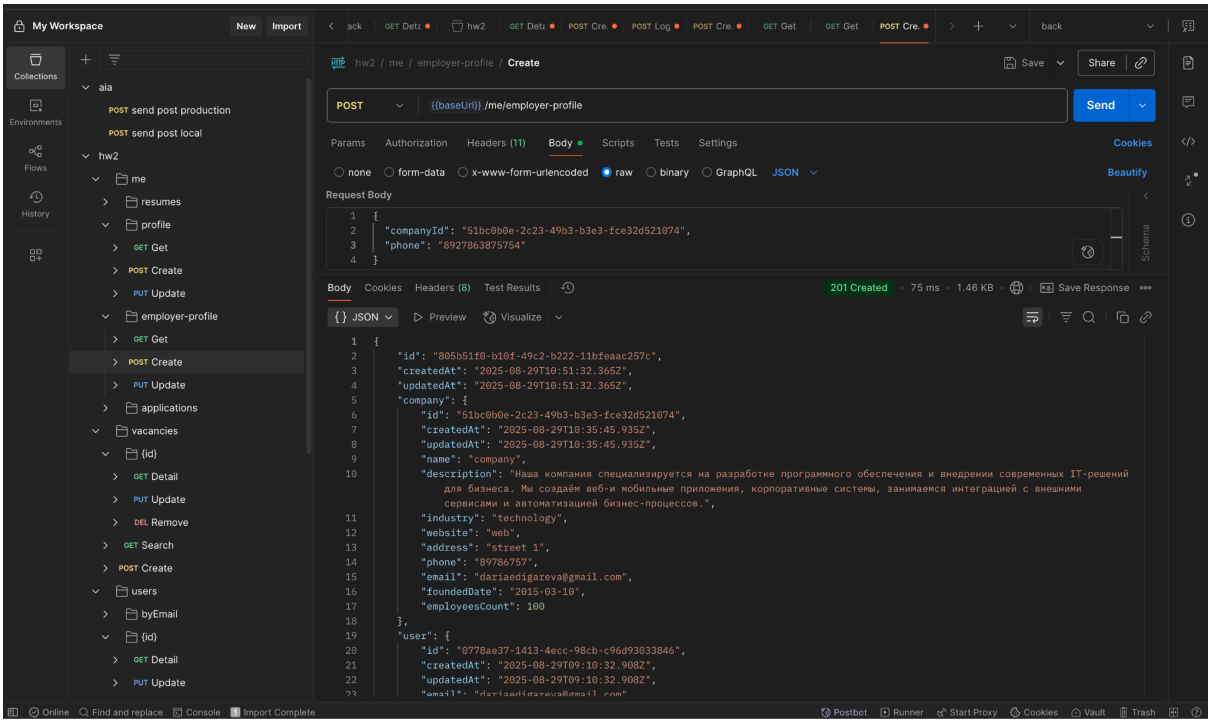
Дерево запросов:

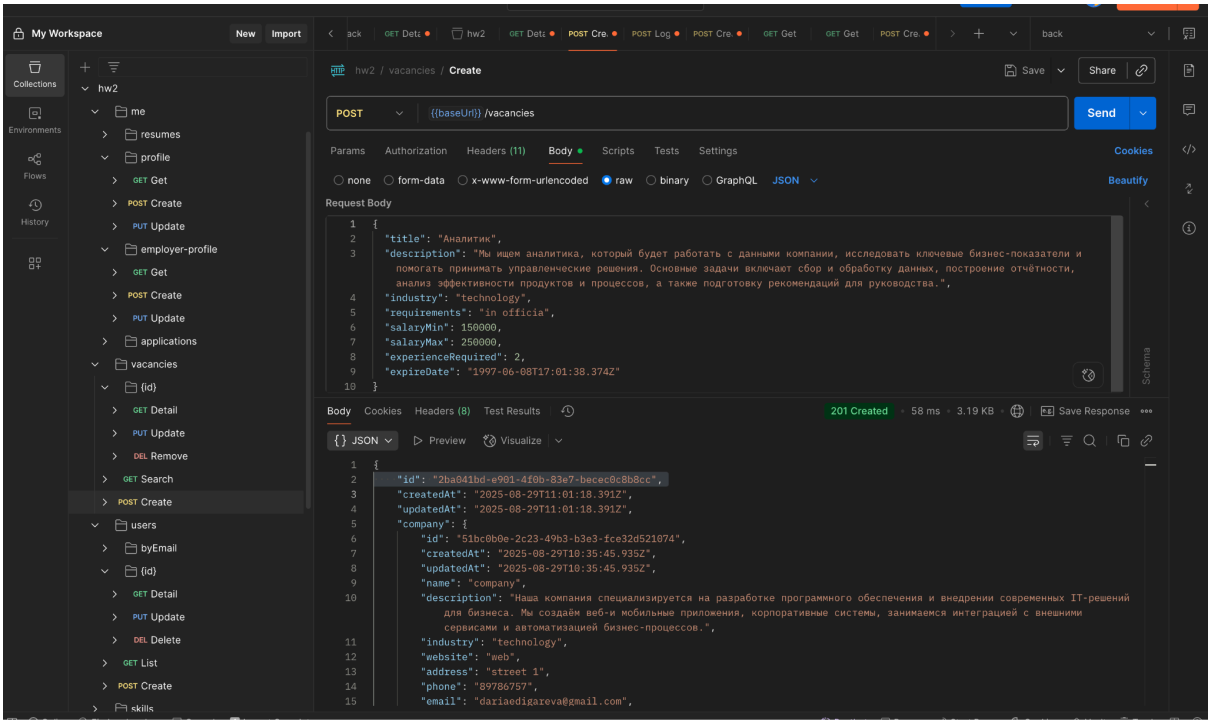


Примеры успешных GET и POST-запросов:









Вывод:

В ходе работы было выполнено автодокументирование API с помощью Swagger, что обеспечило автоматическую генерацию спецификации OpenAPI и удобный доступ к описанию всех реализованных эндпоинтов. Дополнительно спецификация была импортирована в Postman, где настроено окружение с переменными для базового URL и токена

авторизации. Это позволило получить рабочую коллекцию запросов, выполнять и тестировать операции с API, а также использовать коллекцию как готовую документацию для дальнейшей работы и демонстрации.