

**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>`',

    },

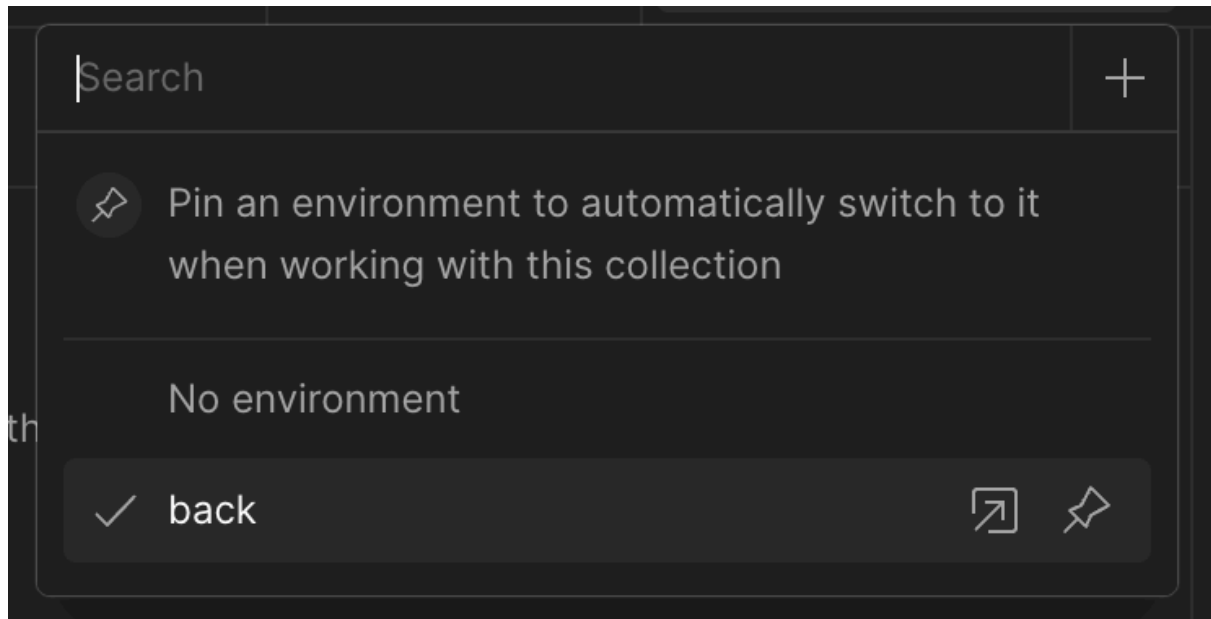
};
```

```
    },  
    ...spec.components.securitySchemes,  
  };  
  
  const uiOptions = {  
    explorer: true,  
    swaggerOptions: {  
      persistAuthorization: true,  
    },  
  };  
  
  app.use('/docs', swaggerUi.serve,  
    swaggerUi.setup(spec, uiOptions));  
}
```

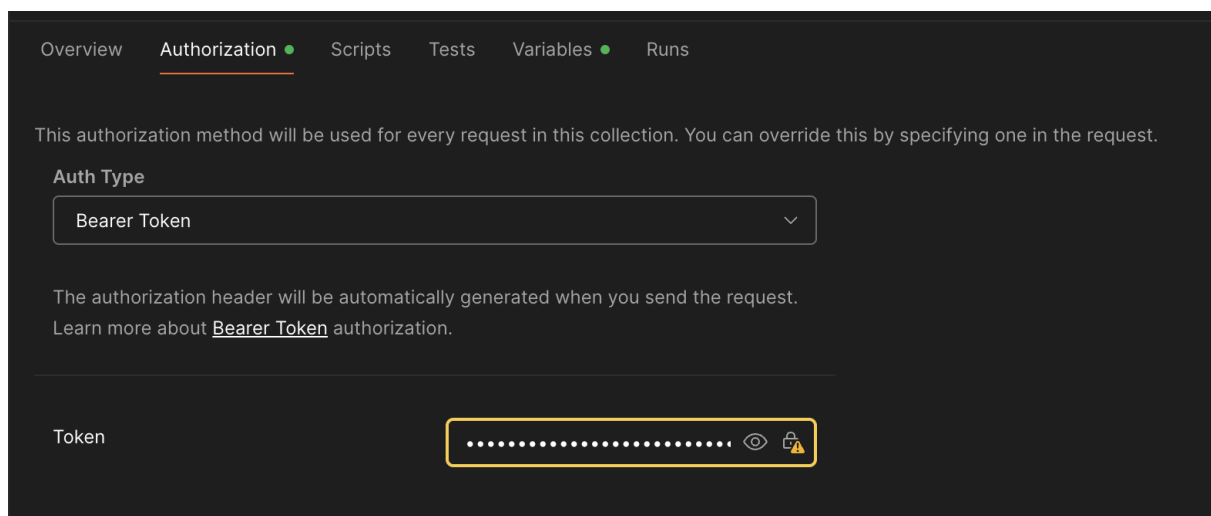
## 2. Документация API средствами Postman.

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.

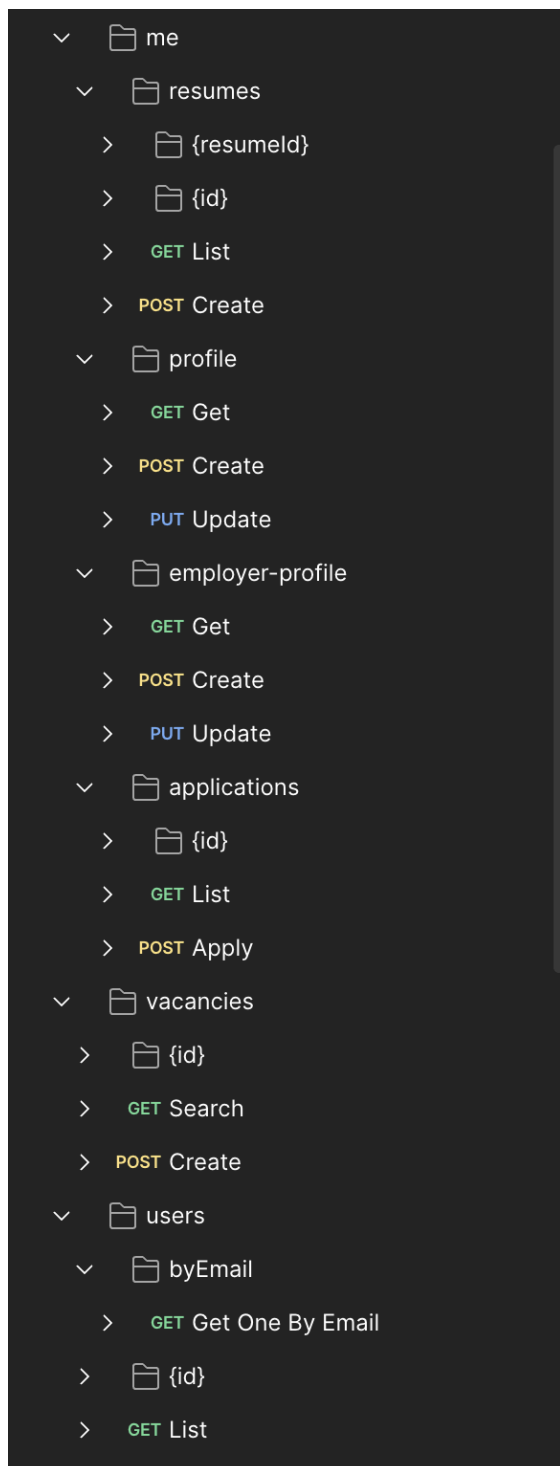
Настройка окружения:



Авторизация Bearer Token:



## Дерево запросов:



# Примеры успешных GET и POST-запросов:

Collections

+

≡

Environments

Flows

History

REST API basics: CRUD, test & variable

hw2 / users / (id) / Detail

GET (baseUri) /users/0778ae37-1413-4ecc-98cb-c96d93033846

Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

Body Cookies Headers (8) Test Results

200 OK 38 ms 523 B Save Response

JSON Preview Visualize

```
1 {
2   "id": "0778ae37-1413-4ecc-98cb-c96d93033846",
3   "createdAt": "2025-08-29T09:19:32.988Z",
4   "updatedAt": "2025-08-29T09:19:32.988Z",
5   "email": "dariaedigareva@gmail.com",
6   "passwordHash": "$2b$10$K0GZAGSFZgCqNv01r8h1KuBLHc01Xjw76771BhdDe3ETWQwV1fxgG",
7   "role": "jobseeker"
8 }
```

My Workspace

New Import

ps POST sen POST sen back GET Det hw2 GET Det POST Cre POST Log POST Cre

back

Collections

+

≡

Environments

Flows

History

REST API basics: CRUD, test & variable

hw2 / companies / Create

POST (baseUri) /companies

Send

Params Authorization Headers (11) Body Scripts Tests Settings Cookies

Request Body

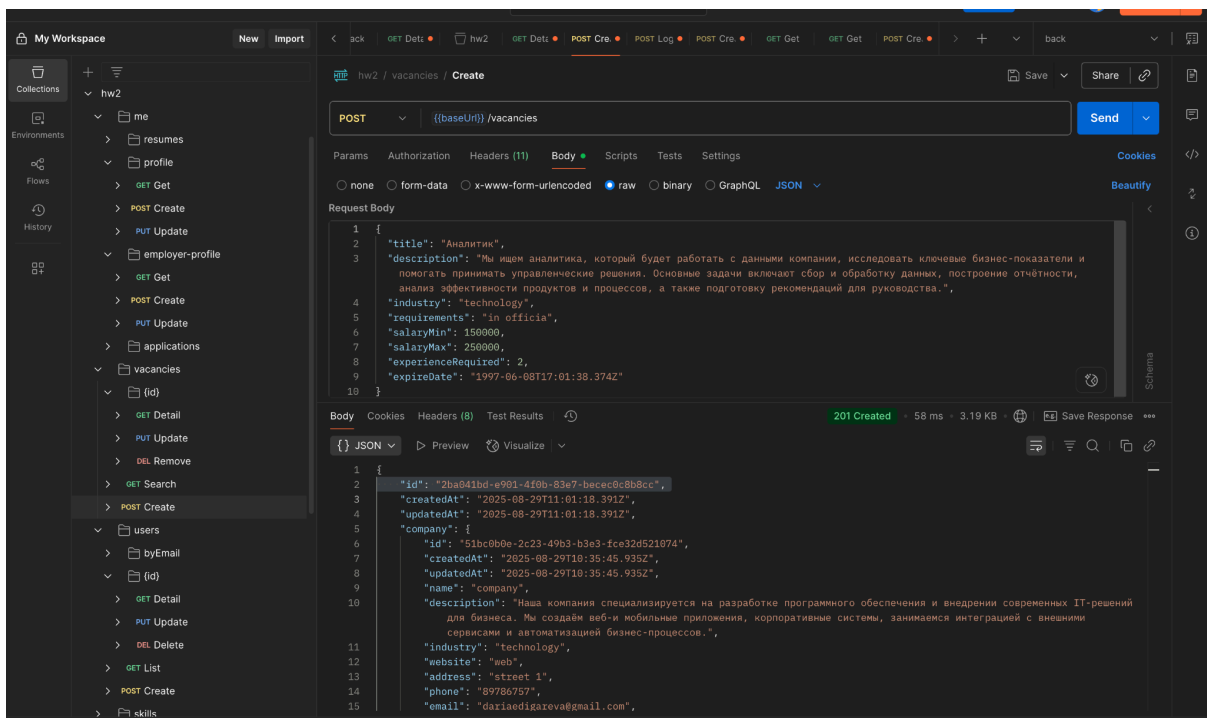
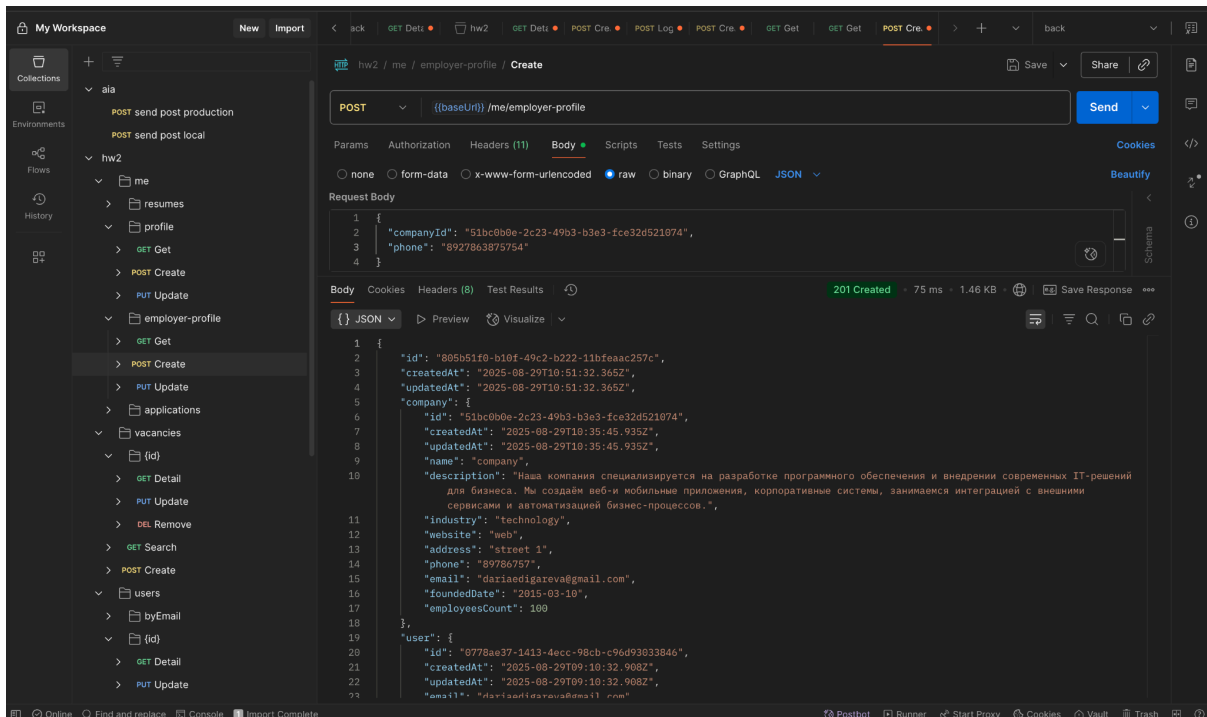
```
1 {
2   "name": "company",
3   "industry": "technology",
4   "description": "Наша компания специализируется на разработке программного обеспечения и внедрении современных IT-решений для бизнеса. Мы создаем веб-и мобильные приложения, корпоративные системы, занимаемся интеграцией с внешними сервисами и автоматизацией бизнес-процессов.",
5   "website": "web",
6   "address": "street 1",
7   "phone": "89786757",
8   "email": "dariaedigareva@gmail.com",
9   "foundedDate": "2015-03-10",
10  "employeesCount": 100
11 }
```

Body Cookies Headers (8) Test Results

201 Created 47 ms 1.05 KB Save Response

JSON Preview Visualize

```
1 {
2   "id": "51bc0b0e-2c23-49b3-b3e3-fee32d521074",
3   "createdAt": "2025-08-29T10:35:45.936Z",
4   "updatedAt": "2025-08-29T10:35:45.936Z",
5   "name": "company",
6   "description": "Наша компания специализируется на разработке программного обеспечения и внедрении современных IT-решений для бизнеса. Мы создаем веб-и мобильные приложения, корпоративные системы, занимаемся интеграцией с внешними сервисами и автоматизацией бизнес-процессов.",
7   "industry": "technology",
8   "website": "web",
9   "address": "street 1",
10  "phone": "89786757",
11  "email": "dariaedigareva@gmail.com",
12  "foundedDate": "2015-03-10",
13  "employeesCount": 100
14 }
```



## 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.