

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 3

Выполнил:

Якунин Андрей

Группа

К3341

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- реализовать автодокументирование средствами swagger;
- реализовать документацию API средствами Postman.

Ход работы

Сделай документацию в Swagger с помощью swagger-jsdoc

Реализация в коде

```
import swaggerJSDoc from "swagger-jsdoc";
import swaggerUi from "swagger-ui-express";
import { Express } from "express";

const options: swaggerJSDoc.Options = {
  definition: {
    openapi: "3.0.0",
    info: {
      title: "Real Estate API",
      version: "1.0.0",
      description: "Документация API для твоего проекта недвижимости",
    },
    servers: [
      {
        url: "http://localhost:3000/api",
      },
    ],
    components: {
      securitySchemes: {
        bearerAuth: {
          type: "http",
          scheme: "bearer",
          bearerFormat: "JWT",
        },
      },
    },
    security: [
      {
        bearerAuth: [],
      },
    ],
  },
  apis: [".src/routers/*.ts"],
};

const swaggerSpec : object = swaggerJSDoc(options);

export const setupSwagger : (app: Express) => void = (app: Express) => { Show usages
  app.use( path: "/api/docs", swaggerUi.serve, swaggerUi.setup(swaggerSpec));
};
```

Этот файл swagger.ts нужен для автоматической генерации и отображения документации твоего API в формате OpenAPI (Swagger).

Вывод

```
import ...

const router = Router();

/**
 * @swagger
 * tags:
 *   name: Auth
 *   description: Аутентификация и управление пользователями
 */

/**
 * @swagger
 * /auth/register:
 *   post:
 *     summary: Регистрация нового пользователя
 *     tags: [Auth]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             example:
 *               username: "user123"
 *               email: "user@example.com"
 *               password: "strongpassword"
 *     responses:
 *       201:
 *         description: Пользователь зарегистрирован
 */
router.post( path: "/register", register);

/**
 * @swagger
 * /auth/login:
 *   post:
 *     summary: Вход пользователя (авторизация)
 *     tags: [Auth]
 *     requestBody:

```

```
router.post( path: "/login", login);

/**
 * @swagger
 * /auth/update-password:
 *   put:
 *     summary: Обновление пароля пользователя
 *     tags: [Auth]
 *     security:
 *       - bearerAuth: []
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             example:
 *               oldPassword: "oldpassword"
 *               newPassword: "newstrongpassword"
 *     responses:
 *       200:
 *         description: Пароль обновлен
 *       401:
 *         description: Не авторизован или неверный старый пароль
 */
router.put( path: "/update-password", auth, updatePassword);

export default router; Show usages
```

В этом файле ты описал маршруты (эндпоинты) аутентификации с помощью обычного Express Router.

Для каждого маршрута ты добавил специальные комментарии с аннотациями Swagger (@swagger), которые описывают:

- Путь и метод (/auth/register, POST и т.д.)
- Краткое описание (summary)
- Теги (группа эндпоинтов)
- Схему тела запроса (JSON с примерами полей)
- Возможные ответы сервера с кодами и описаниями
- Требования безопасности (JWT токен для обновления пароля)

Эти комментарии считывает swagger-jsdoc (если он настроен) и автоматически генерирует из них структуру OpenAPI.

Затем swagger-ui-express показывает эту структуру в виде красивой веб-страницы с документацией API.

Документация

- Добавил в маршруты специальные JSDoc-комментарии с аннотациями Swagger (теги @swagger), описывающие каждый эндпоинт, параметры и ответы.
- Настроил в проекте swagger-jsdoc и swagger-ui-express (например, в файле swagger.ts), чтобы они читали эти комментарии и строили из них документацию.
- Подключил к своему Express приложению маршрут (например, /api/docs), где отображается сгенерированная из комментариев Swagger документация.

Таким образом, Swagger документация появилась автоматически на основе этих комментариев — не нужно вручную писать JSON или YAML спецификации.

Swagger

Real Estate API

1.0.0OAS 3.0

Документация API для твоего проекта недвижимости

Servers

http://localhost:3000/api

Authorize

CountryHouse

Управление загородными домами

POST/countryHouseСоздать загородный дом

GET/countryHouseПолучить список всех загородных домов

GET/countryHouse/{id}Получить загородный дом по ID

PUT/countryHouse/{id}Обновить загородный дом по ID

DELETE/countryHouse/{id}Удалить загородный дом по ID

Advertisements

Операции с объявлениями

POST/adsСоздать новое объявление

GET/adsПолучить список всех объявлений

GET/ads/{id}Получить объявление по ID

PUT/ads/{id}Обновить объявление

Это уже готовый swagger, где можно проверить и выполнить все запросы по url - <http://localhost:3000/api/docs/>

Вывод по работе

В ходе работы я реализовал документирование API с помощью Swagger, добавив в маршруты специальные аннотации в формате JSDoc. Это позволило автоматически генерировать и отображать удобную интерактивную документацию для всех эндпоинтов. Таким образом, улучшена прозрачность и удобство использования API для разработчиков и клиентов сервиса