# САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 3

Выполнил:

Якунин Андрей

Группа

K3341

Проверил: Добряков Д. И.

Санкт-Петербург

2025 г.

#### Задача

Вывод

- 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: {
           description: "Документация АРІ для твоего проекта недвижимости",
        servers: [
                url: "http://localhost:3000/api",
              bearerAuth: {
                 type: "http",
                  scheme: "bearer",
                  bearerFormat: "JWT",
               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).

```
import ...
const router = Router();

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

/**
     * @swagger
     * /auth/register:
     * post:
     * summary: Pesucrpaция 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.

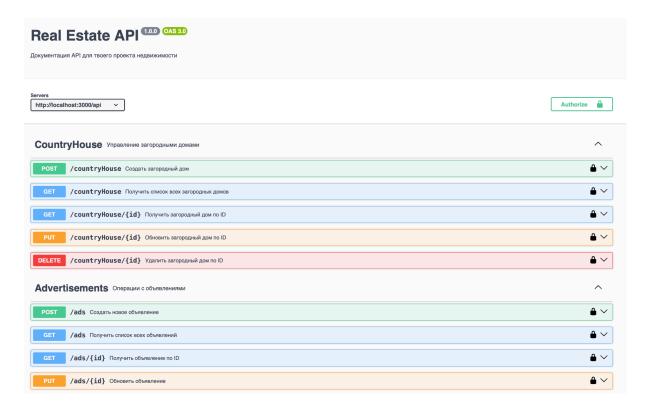
Затем swagger-ui-express показывает эту структуру в виде красивой веб-страницы с документацией API.

### 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



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

# 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