**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Домашнее задание №3

Выполнил:

Соловьева П.А.

Группа К3344

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

**Задача**

Реализовать автодокументирование средствами Swagger и документацию API средствами Postman для системы управления недвижимостью на базе Node.js/Express/TypeORM.

**Ход работы**

### **1. Автодокументирование средствами Swagger**

В маршруты добавлена аннотация для Swagger/OpenAPI. Пример на Пользователях:

**UserRoutes**

**import { Router } from "express";**

**import { UserController } from "../controllers/user.controller";**

**const *router* = Router();**

***/\*\****

***\* @swagger***

***\* tags:***

***\* name: Users***

***\* description: Работа с пользователями***

***\*/***

***/\*\****

***\* @swagger***

***\* /users:***

***\* post:***

***\* summary: Создать нового пользователя***

***\* tags: [Users]***

***\* requestBody:***

***\* required: true***

***\* content:***

***\* application/json:***

***\* schema:***

***\* type: object***

***\* properties:***

***\* first\_name:***

***\* type: string***

***\* last\_name:***

***\* type: string***

***\* email:***

***\* type: string***

***\* format: email***

***\* phone\_number:***

***\* type: string***

***\* nullable: true***

***\* password\_hash:***

***\* type: string***

***\* role:***

***\* type: string***

***\* enum: [owner, tenant, admin]***

***\* required:***

***\* - first\_name***

***\* - last\_name***

***\* - email***

***\* - password\_hash***

***\* responses:***

***\* 201:***

***\* description: Пользователь создан***

***\* content:***

***\* application/json:***

***\* schema:***

***\* $ref: '#/components/schemas/User'***

***\*/***

***router*.post("/", UserController.*create*);**

***/\*\****

***\* @swagger***

***\* /users:***

***\* get:***

***\* summary: Получить список пользователей***

***\* tags: [Users]***

***\* responses:***

***\* 200:***

***\* description: Список пользователей***

***\* content:***

***\* application/json:***

***\* schema:***

***\* type: array***

***\* items:***

***\* $ref: '#/components/schemas/User'***

***\*/***

***router*.get("/", UserController.*findAll*);**

***/\*\****

***\* @swagger***

***\* /users/{id}:***

***\* get:***

***\* summary: Получить пользователя по ID***

***\* tags: [Users]***

***\* parameters:***

***\* - in: path***

***\* name: id***

***\* required: true***

***\* schema:***

***\* type: string***

***\* format: uuid***

***\* responses:***

***\* 200:***

***\* description: Пользователь найден***

***\* content:***

***\* application/json:***

***\* schema:***

***\* $ref: '#/components/schemas/User'***

***\* 404:***

***\* description: Пользователь не найден***

***\*/***

***router*.get("/:id", UserController.*findById*);**

***/\*\****

***\* @swagger***

***\* /users/{id}:***

***\* put:***

***\* summary: Обновить данные пользователя***

***\* tags: [Users]***

***\* parameters:***

***\* - in: path***

***\* name: id***

***\* required: true***

***\* schema:***

***\* type: string***

***\* format: uuid***

***\* requestBody:***

***\* required: true***

***\* content:***

***\* application/json:***

***\* schema:***

***\* type: object***

***\* properties:***

***\* first\_name:***

***\* type: string***

***\* last\_name:***

***\* type: string***

***\* email:***

***\* type: string***

***\* format: email***

***\* phone\_number:***

***\* type: string***

***\* nullable: true***

***\* password\_hash:***

***\* type: string***

***\* role:***

***\* type: string***

***\* enum: [owner, tenant, admin]***

***\* responses:***

***\* 200:***

***\* description: Пользователь обновлен***

***\* content:***

***\* application/json:***

***\* schema:***

***\* $ref: '#/components/schemas/User'***

***\* 404:***

***\* description: Пользователь не найден***

***\*/***

***router*.put("/:id", UserController.*update*);**

***/\*\****

***\* @swagger***

***\* /users/{id}:***

***\* delete:***

***\* summary: Удалить пользователя***

***\* tags: [Users]***

***\* parameters:***

***\* - in: path***

***\* name: id***

***\* required: true***

***\* schema:***

***\* type: string***

***\* format: uuid***

***\* responses:***

***\* 200:***

***\* description: Пользователь удален***

***\* 404:***

***\* description: Пользователь не найден***

***\*/***

***router*.delete("/:id", UserController.*delete*);**

**export default *router*;**

**Swagger.ts**

**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: "Rental Service API",**

**version: "1.0.0",**

**description: "RESTful API для сервиса аренды недвижимости",**

**},**

**servers: [**

**{**

**url: "http://localhost:5000/api",**

**},**

**],**

**components: {**

**components: {**

**schemas: {**

**User: {**

**type: "object",**

**properties: {**

**id: { type: "string", format: "uuid" },**

**first\_name: { type: "string" },**

**last\_name: { type: "string" },**

**email: { type: "string", format: "email" },**

**phone\_number: { type: "string", nullable: true },**

**role: { type: "string", enum: ["owner", "tenant", "admin"] },**

**}**

**},**

**UserCreate: {**

**type: "object",**

**properties: {**

**first\_name: { type: "string" },**

**last\_name: { type: "string" },**

**email: { type: "string", format: "email" },**

**password: { type: "string" },**

**phone\_number: { type: "string", nullable: true },**

**},**

**required: ["first\_name", "last\_name", "email", "password"]**

**},**

**Property: {**

**type: "object",**

**properties: {**

**id: { type: "string", format: "uuid" },**

**owner: { $ref: "#/components/schemas/User" },**

**title: { type: "string" },**

**description: { type: "string" },**

**type: { type: "string" },**

**location: { type: "string" },**

**price\_per\_month: { type: "number" },**

**}**

**},**

**PropertyCreate: {**

**type: "object",**

**properties: {**

**owner\_id: { type: "string", format: "uuid" },**

**title: { type: "string" },**

**description: { type: "string" },**

**type: { type: "string" },**

**location: { type: "string" },**

**price\_per\_month: { type: "number" },**

**},**

**required: ["owner\_id", "title", "description", "type", "location", "price\_per\_month"]**

**},**

**Rental: {**

**type: "object",**

**properties: {**

**id: { type: "string", format: "uuid" },**

**property: { $ref: "#/components/schemas/Property" },**

**tenant: { $ref: "#/components/schemas/User" },**

**start\_date: { type: "string", format: "date" },**

**end\_date: { type: "string", format: "date" },**

**status: { type: "string" }**

**}**

**},**

**RentalCreate: {**

**type: "object",**

**properties: {**

**property\_id: { type: "string", format: "uuid" },**

**tenant\_id: { type: "string", format: "uuid" },**

**start\_date: { type: "string", format: "date" },**

**end\_date: { type: "string", format: "date" },**

**status: { type: "string" }**

**},**

**required: ["property\_id", "tenant\_id", "start\_date", "end\_date", "status"]**

**},**

**MessageCreate: {**

**type: "object",**

**properties: {**

**sender\_id: { type: "string", format: "uuid" },**

**receiver\_id: { type: "string", format: "uuid" },**

**rental\_id: { type: "string", format: "uuid" },**

**content: { type: "string" }**

**},**

**required: ["sender\_id", "receiver\_id", "rental\_id", "content"]**

**},**

**ReviewCreate: {**

**type: "object",**

**properties: {**

**rental\_id: { type: "string", format: "uuid" },**

**reviewer\_id: { type: "string", format: "uuid" },**

**rating: { type: "integer" },**

**comment: { type: "string" }**

**},**

**required: ["rental\_id", "reviewer\_id", "rating", "comment"]**

**},**

**AmenityCreate: {**

**type: "object",**

**properties: {**

**name: { type: "string" },**

**description: { type: "string" }**

**},**

**required: ["name"]**

**},**

**PropertyAmenityCreate: {**

**type: "object",**

**properties: {**

**property\_id: { type: "string", format: "uuid" },**

**amenity\_id: { type: "string", format: "uuid" }**

**},**

**required: ["property\_id", "amenity\_id"]**

**}**

**}**

**}**

**}**

**},**

**apis: ["./src/routes/\*.ts"]**

**};**

**const specs = swaggerJsdoc(options);**

**export const setupSwagger = (app: Express) => {**

**app.use("/api/docs", swaggerUi.*serve*, swaggerUi.setup(specs));**

**};**

Импортирование swagger в index.ts:

**import express from "express";**

**import { *AppDataSource* } from "./config/data-source";**

**import *userRoutes* from "./routes/user.routes";**

**import *propertyRoutes* from "./routes/property.routes";**

**import *rentalRoutes* from "./routes/rental.routes";**

**import *messageRoutes* from "./routes/message.routes";**

**import *reviewRoutes* from "./routes/review.routes";**

**import *amenityRoutes* from "./routes/amenity.routes";**

**import *propertyAmenityRoutes* from "./routes/propertyAmenity.routes";**

**import { loggerMiddleware } from "./middleware/loggerMiddleware";**

**import { errorMiddleware } from "./middleware/errorMiddleware";**

**import { setupSwagger } from "./swagger";**

**const app = express();**

**app.use(express.*json*());**

**// Логирование всех запросов**

**app.use(loggerMiddleware);**

**// Роуты**

**app.use("/api/users", *userRoutes*);**

**app.use("/api/properties", *propertyRoutes*);**

**app.use("/api/rentals", *rentalRoutes*);**

**app.use("/api/messages", *messageRoutes*);**

**app.use("/api/reviews", *reviewRoutes*);**

**app.use("/api/amenities", *amenityRoutes*);**

**app.use("/api/property-amenities", *propertyAmenityRoutes*);**

**// Swagger**

**setupSwagger(app);**

**// Глобальная обработка ошибок**

**app.use(errorMiddleware);**

**const PORT = *process*.env.PORT || 5000;**

***AppDataSource*.initialize()**

**.then(() => {**

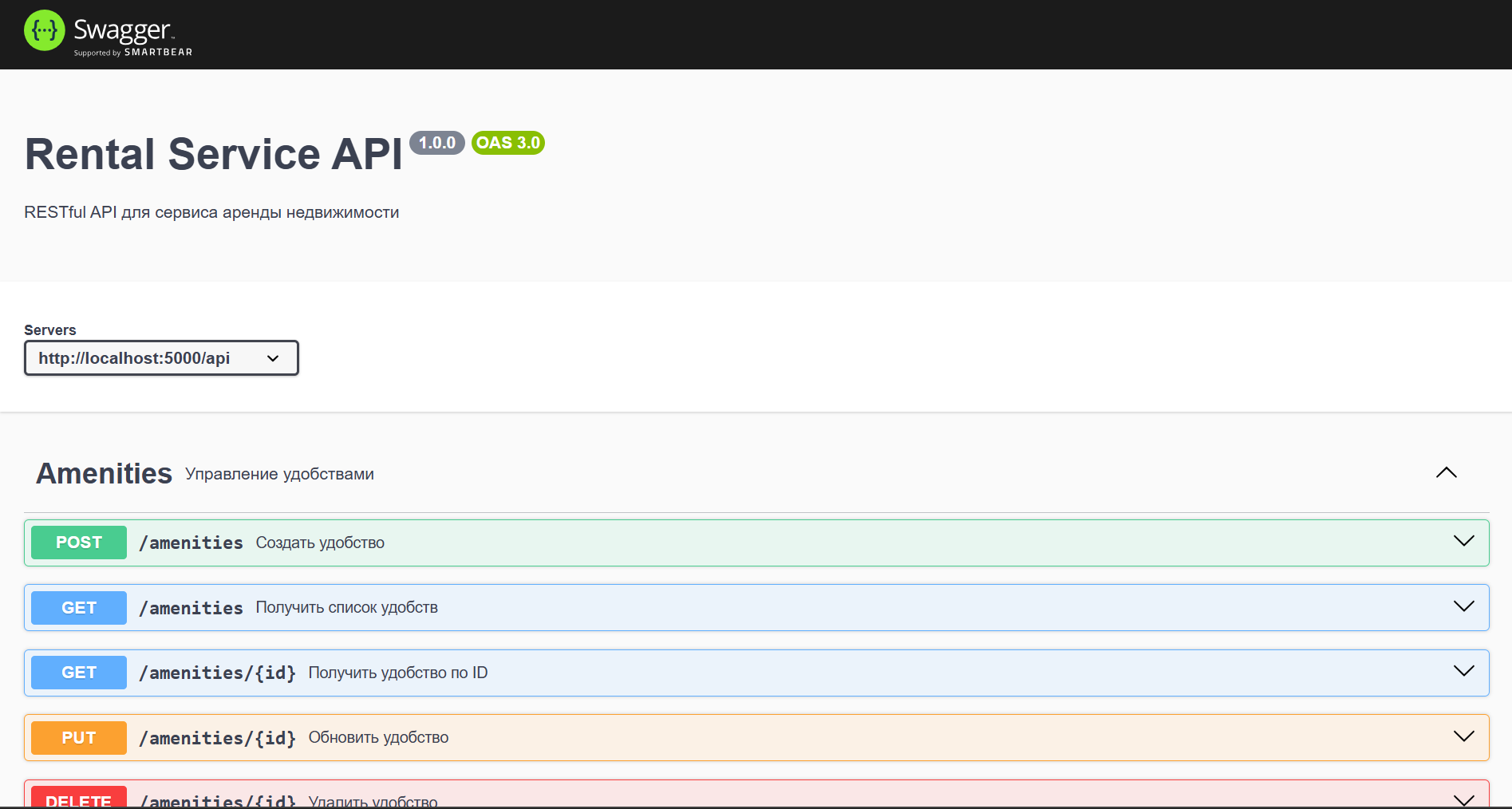
***console*.log("Data Source initialized");**

**app.listen(PORT, () => *console*.log(`Server running on port ${PORT}`));**

**})**

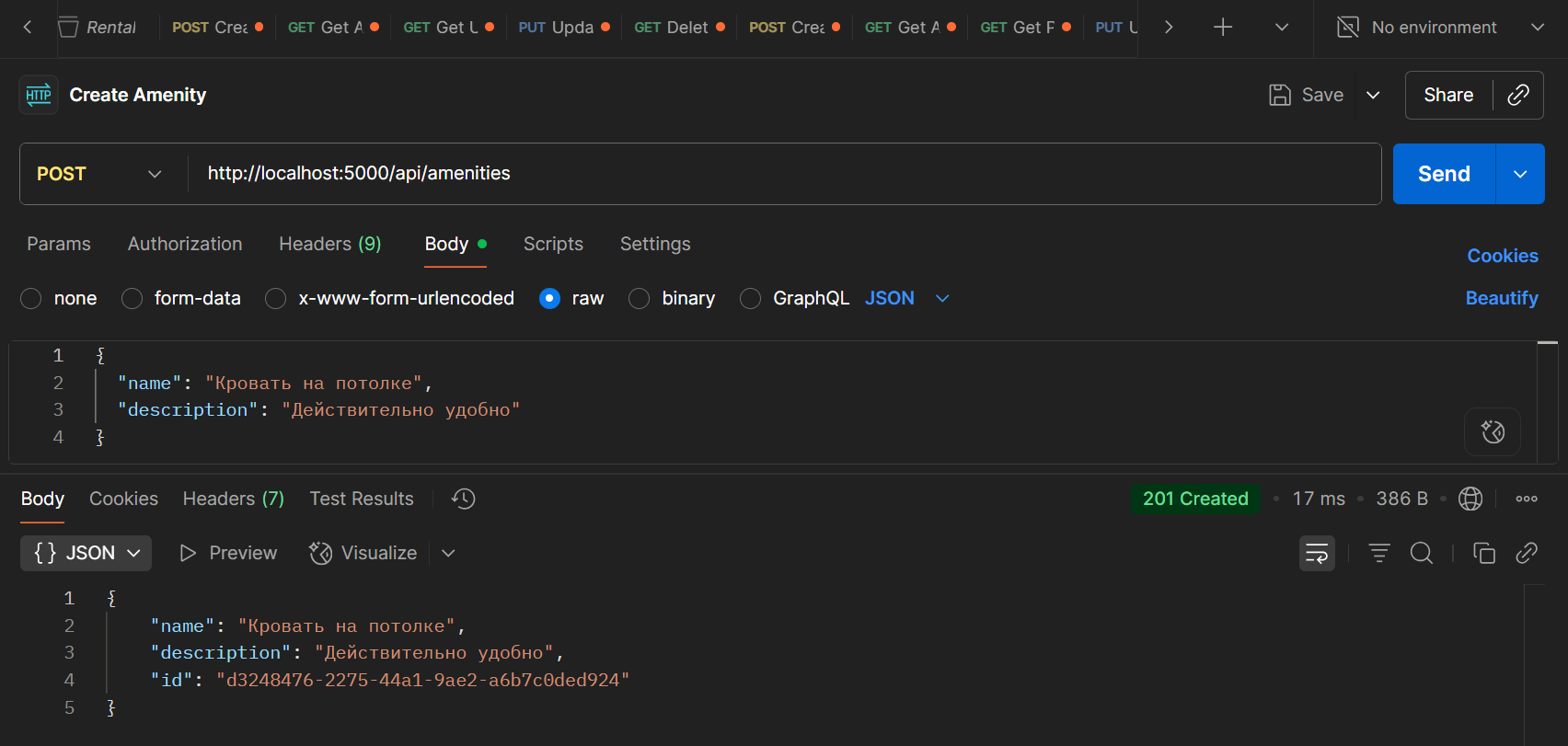
**.catch(err => *console*.error("Error during Data Source initialization", err));**

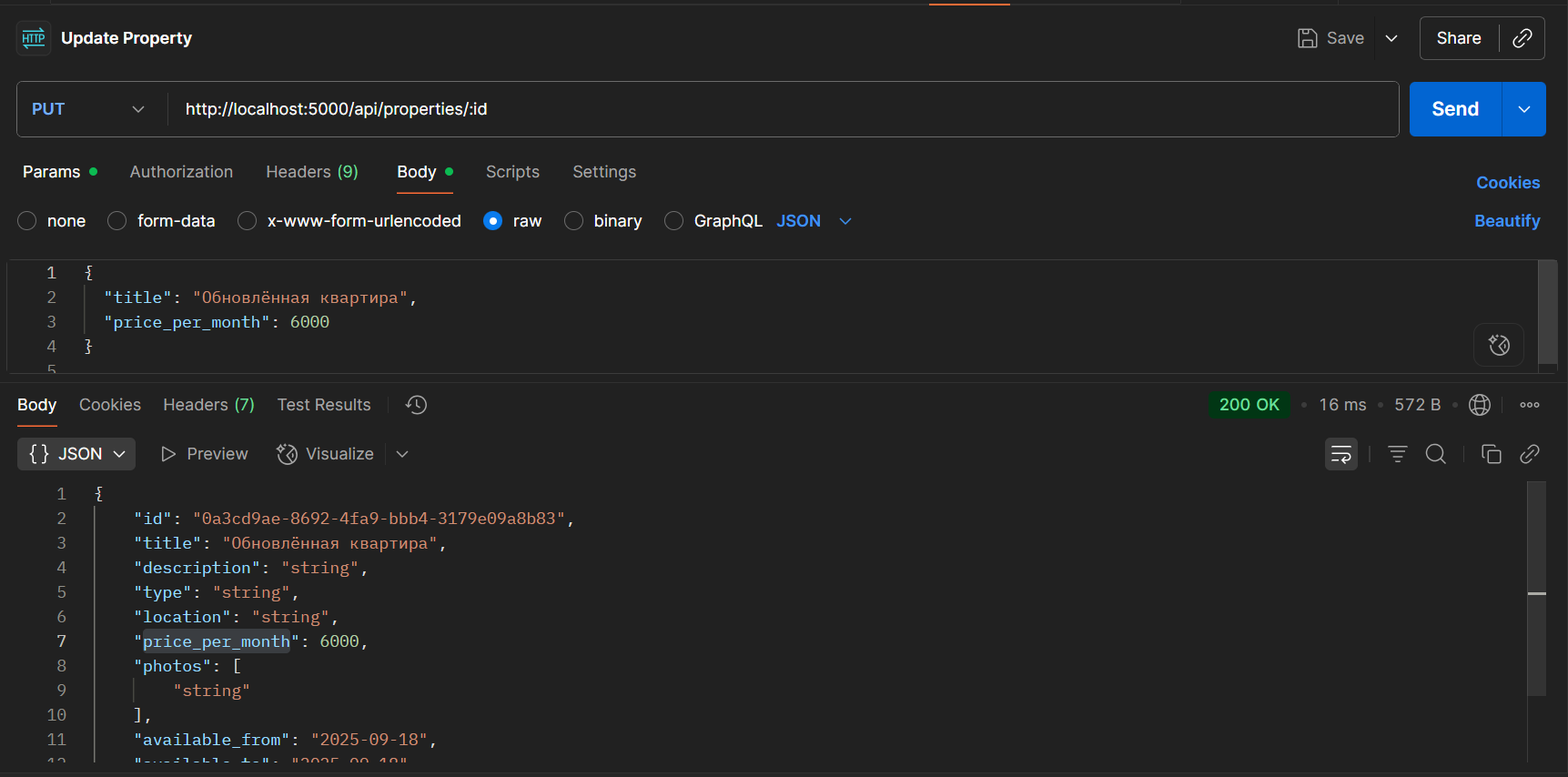
**Документация**

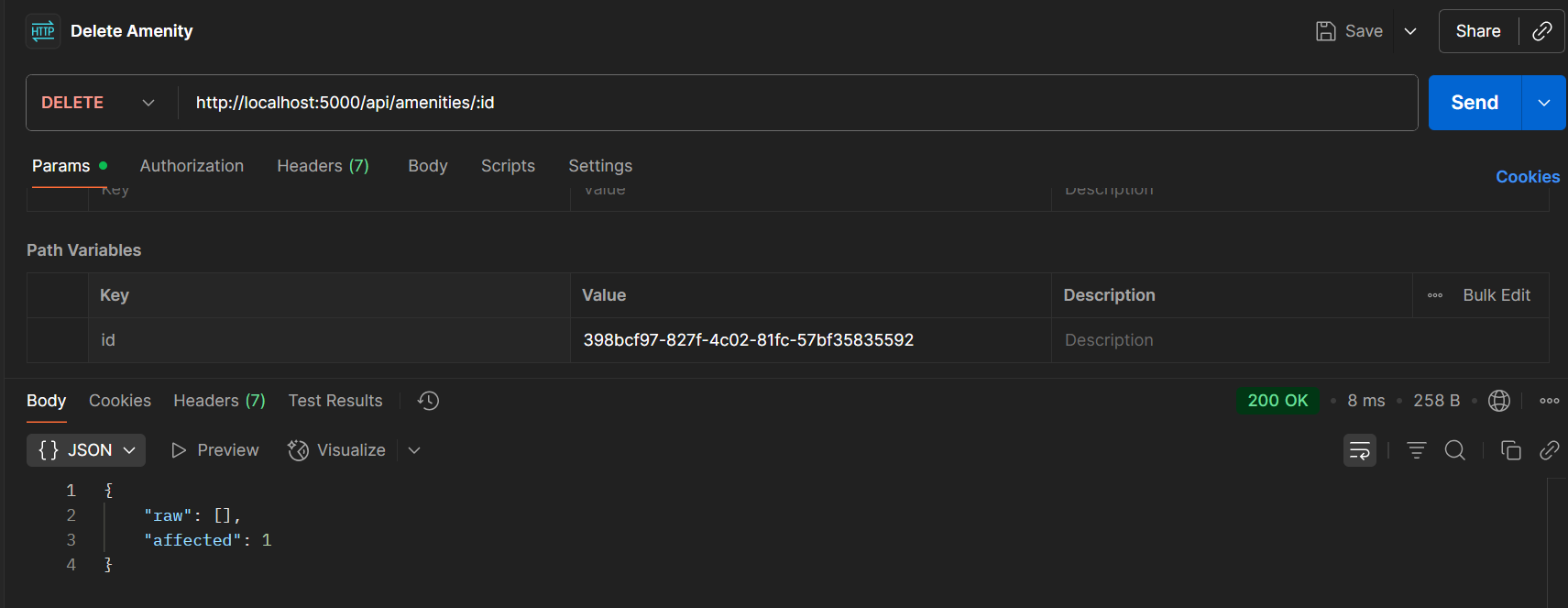
****

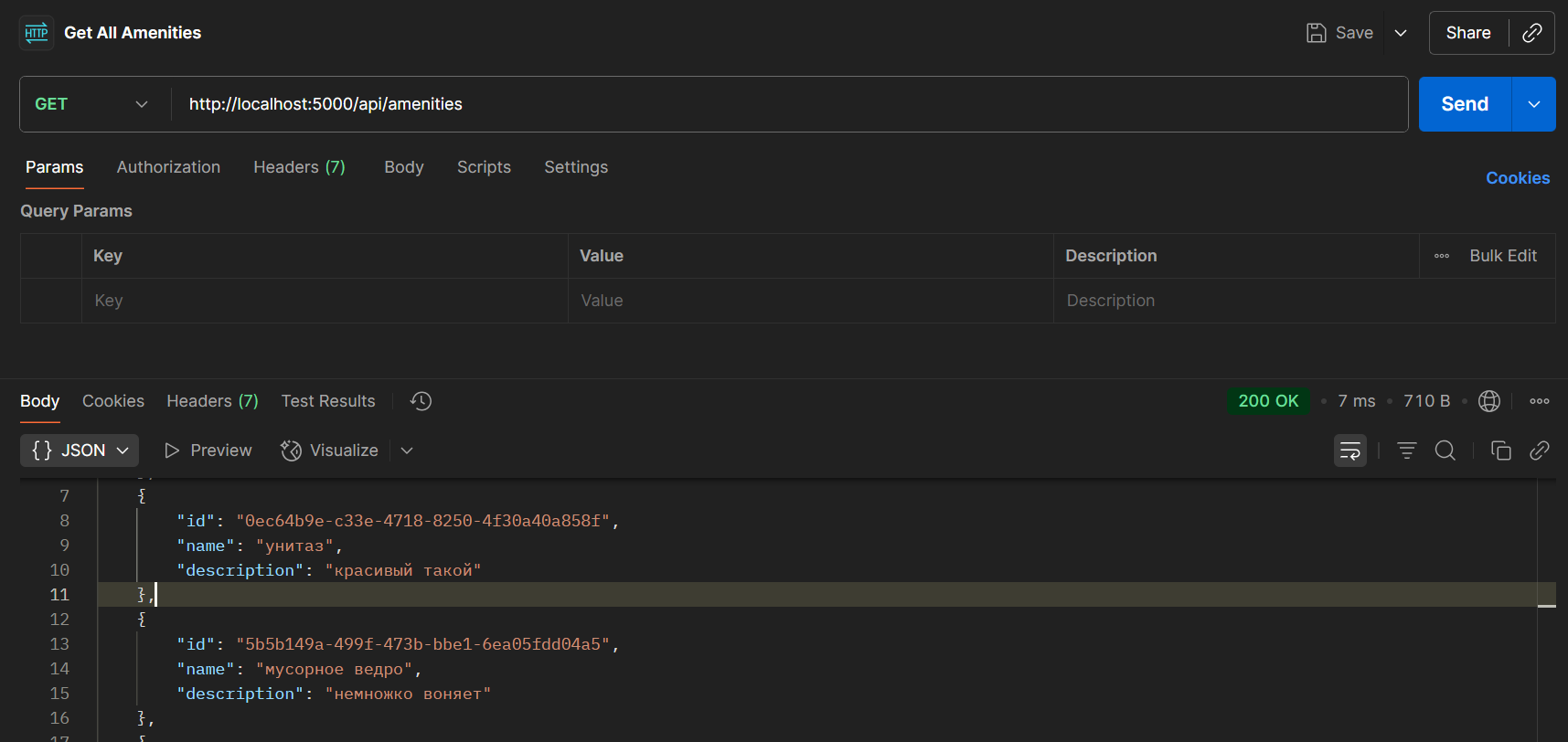
**2. Документирование с помощью Postman**

Примеры запросов:









**Вывод**

Swagger (OpenAPI 3.0):

* Настроена интеграция swagger-jsdoc и swagger-ui-express.
* Для всех сущностей (User, Property, Rental, Message, Review, Amenity, PropertyAmenity) описаны схемы (components.schemas).
* Для каждого маршрута добавлены аннотации @swagger, которые автоматически формируют документацию.
* Документация доступна по маршруту /api/docs и позволяет не только просматривать описание API, но и отправлять тестовые запросы.

Postman:

* Создана коллекция с запросами для всех эндпоинтов API.
* Для каждого метода (GET, POST, PUT, DELETE) настроены примеры входных и выходных данных.
* Документация в Postman позволяет удобно тестировать работу API и использовать готовые запросы для взаимодействия с сервером.

Таким образом, реализовано удобное и наглядное документирование API, которое обеспечивает:

* быстрый доступ к спецификации;
* прозрачность структуры запросов и ответов;
* простоту тестирования и интеграции для разработчиков и сторонних пользователей.