САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа № 3

Выполнил:

Гуторова Инна

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

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

Ход работы

1. Интеграция Swagger в Express-приложение

Для интеграции Swagger был создан файл swagger.ts, содержащий настройку схемы документации OpenAPI:

```
const options = {
    definition: {
      openapi: "3.0.0",
      info: {
        title: "Travel Service API",
        version: "1.0.0",
        description: "REST API for Travel App",
      },
      components: {
        securitySchemes: {
          bearerAuth: {
            type: "http",
            scheme: "bearer",
            bearerFormat: "JWT",
          },
        },
      },
      security: [
        {
          bearerAuth: [],
        },
      ],
    },
    apis: ["./src/routes/*.ts"],
  };
const swaggerSpec = swaggerJsDoc(options);

export function setupSwagger(app: Express) {
  app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));
  app.use('/swagger.json', (_req, res) => {
    res.setHeader('Content-Type', 'application/json');
    res.send(swaggerSpec);
  });
}
```

Функция setupSwagger добавлена в основной файл приложения index.ts:

```ts
import { setupSwagger } from "./swagger";

dotenv.config();

const app = express();
const PORT = process.env.PORT || 3000;
setupSwagger(app);
```

Swagger UI теперь доступен по пути: http://localhost:3000/api-docs.

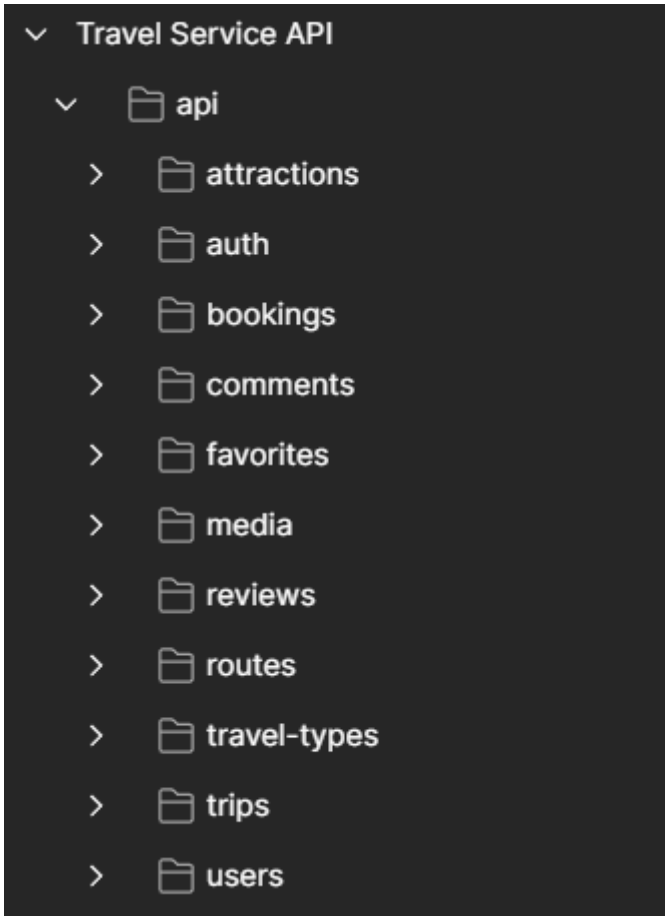
2. Документирование маршрутов с помощью JSDoc-аннотаций

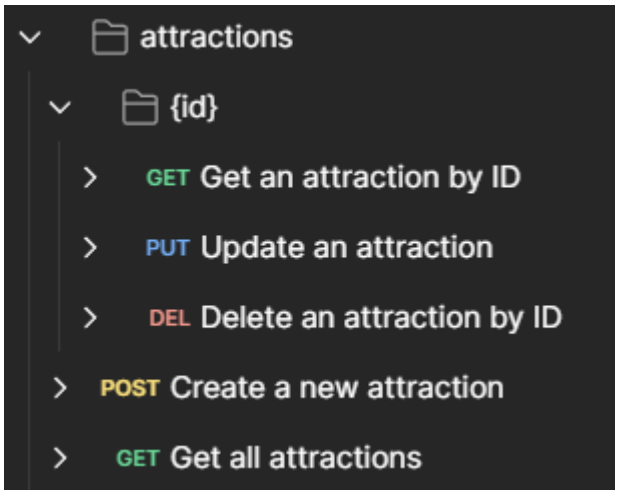
По всем роутам для всех операций (GET, POST, PUT, DELETE) были добавлены Swagger-аннотации. Пример для создания новой достопримечательности в attractionRoute.ts:

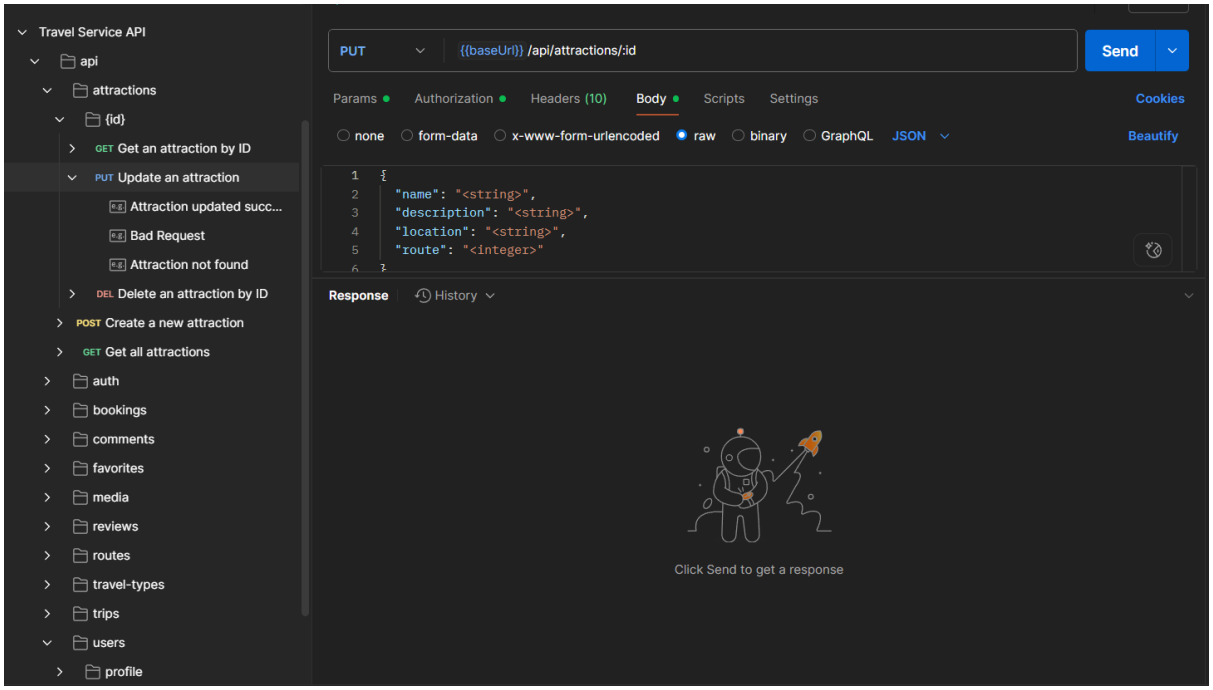
```ts
 * @swagger
 * tags:
 *   name: Attractions
 *   description: Attraction management for tourist routes
 */

/**
 * @swagger
 * /api/attractions:
 *   post:
 *     summary: Create a new attraction
 *     tags: [Attractions]
 *     security:
 *       - bearerAuth: []
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             required:
 *               - name
 *               - description
 *               - location
 *               - route
 *             properties:
 *               name:
 *                 type: string
 *               description:
 *                 type: string
 *               location:
 *                 type: string
 *               route:
 *                 type: integer
 *     responses:
 *       201:
 *         description: Attraction created successfully
 *       400:
 *         description: Bad Request
 *       401:
 *         description: Unauthorized
 *       403:
 *         description: Forbidden (Admin only)
 */
attractionRouter.post('/', authenticate, authorizeAdmin, createAttraction);
```

3. Создание коллекции в Postman

Вместо ручного создания запросов в Postman, исполльзовалась генерация коллекции из OpenAPI-спецификации (swagger.json). Этот файл импортируется в Postman и получаем автоматически созданную коллекцию со всеми описанными эндпоинтами:







Вывод

В ходе лабораторной работы было успешно выполнено:

- Подключение Swagger к Express-приложению с использованием swagger-jsdoc и swagger-ui-express.
- Документирование всех маршрутов REST API с помощью JSDoc-аннотаций.
- Настройка Swagger UI для визуального тестирования и навигации по API.
- Создание коллекции Postman, готовой для проверки в автоматизированной системе.