САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №3

Выполнил:

Исмагилова Карина

Группа: K3344

Проверил:
Добряков Д. И.

Санкт-Петербург

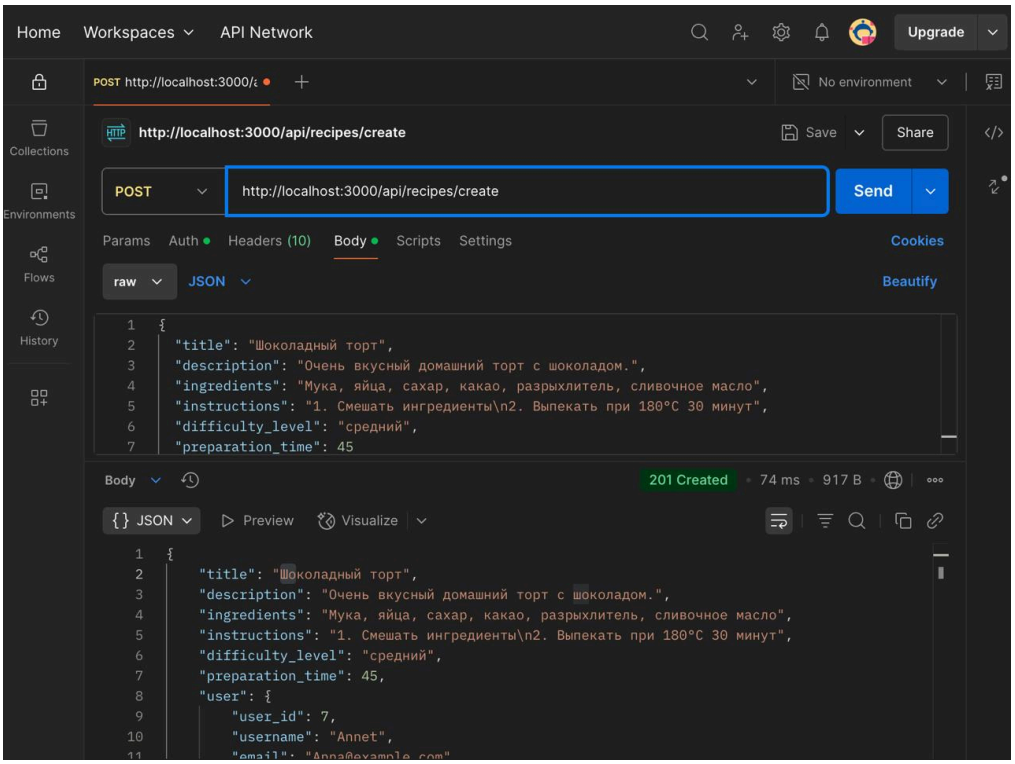
2025 г.

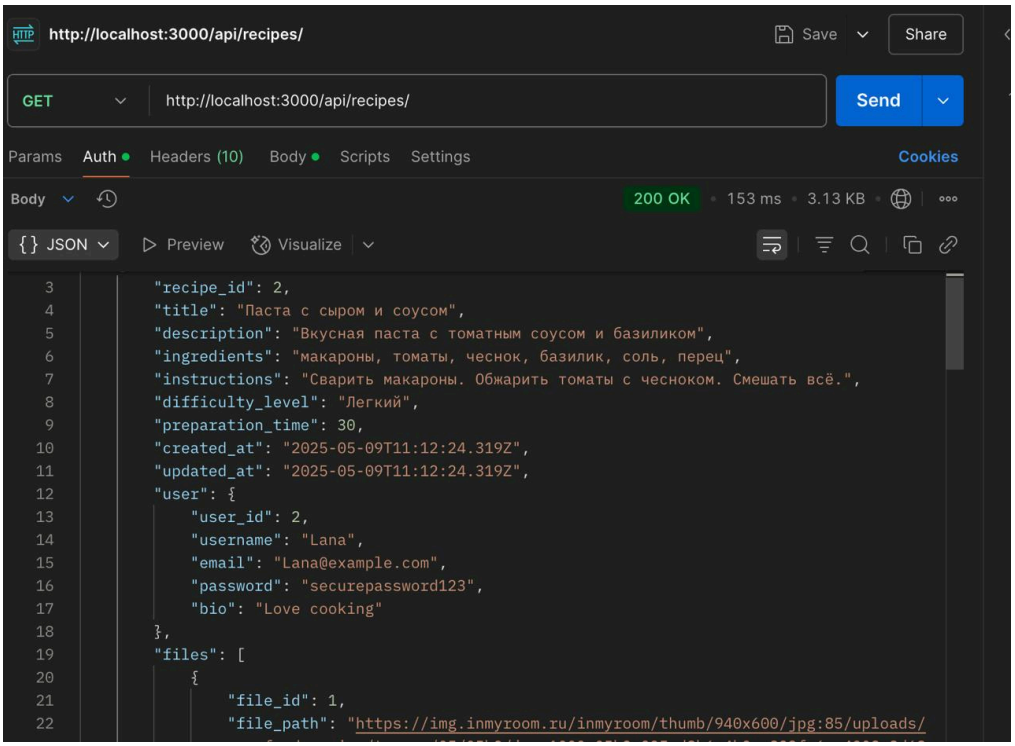
Задача

Реализовать автодокументирование средствами swagger, реализовать документацию API средствами Postman.

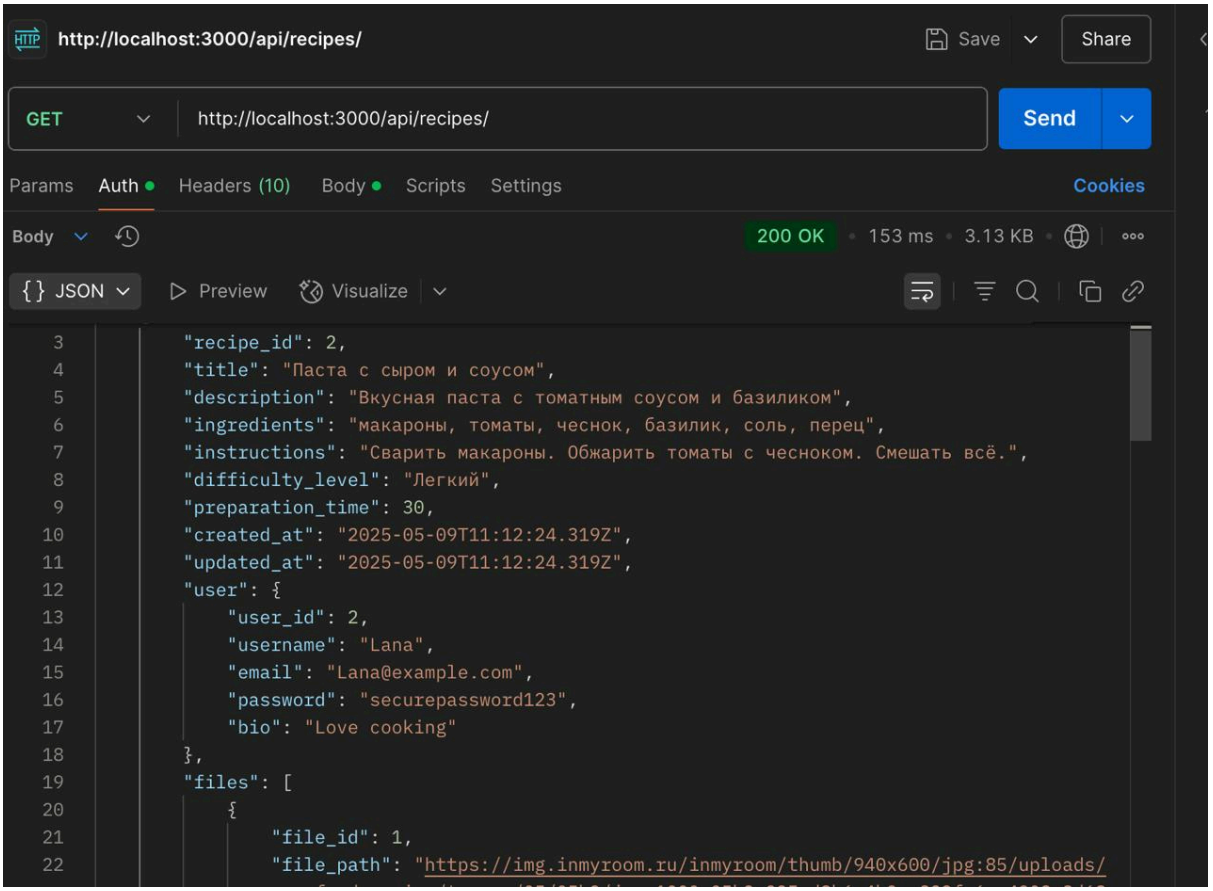
Ход работы

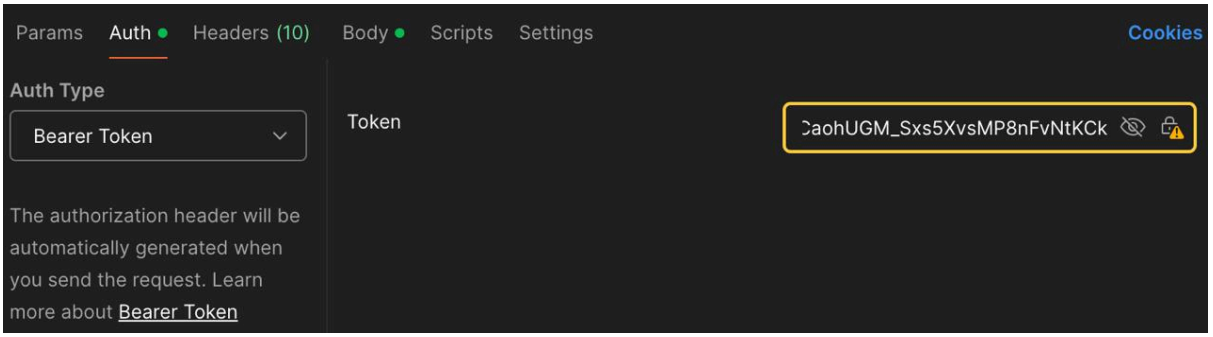
Были протестированы все crud-запросы с помощью Postman.





Все запросы кроме get доступны благодаря вход и получению токена. Все операции с favourite доступны только после входа.





Для реализации документирования средствами Swagger был создан файл для настройки и интеграции Swagger-документации в Express-приложение.

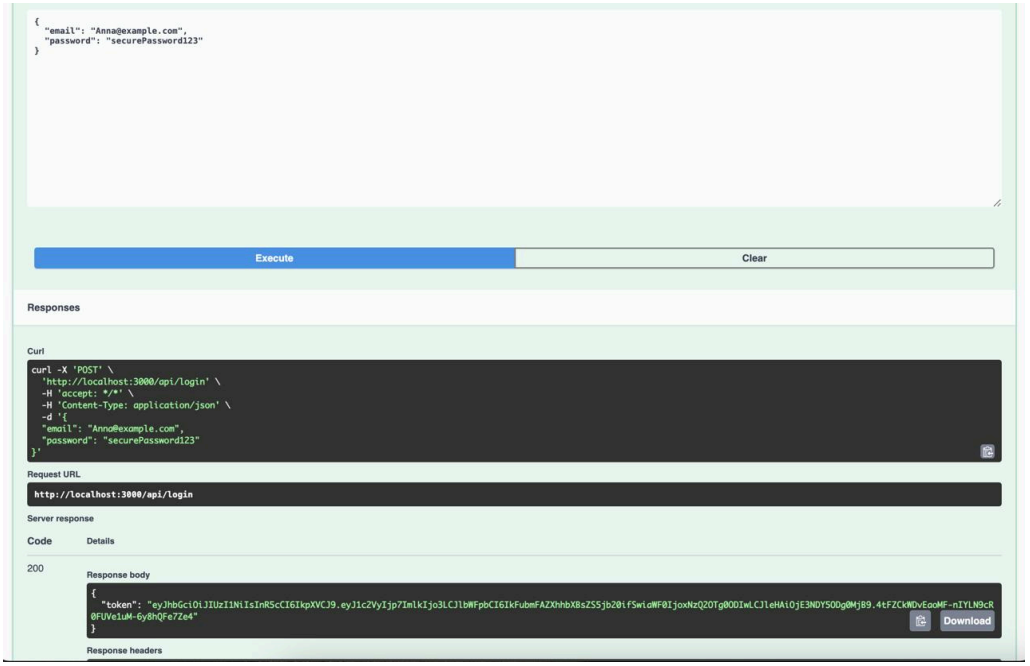
```
import swaggerJsdoc from 'swagger-jsdoc';
import swaggerUi from 'swagger-ui-express';
import e from 'express';

const options: swaggerJsdoc.Options = {
 definition: {
   openapi: '3.0.0',
   info: {
     version: '1.0.0',
     description: 'API documentation',
   },
   components: {
     securitySchemes: {
       bearerAuth: {
         type: 'http',
         scheme: 'bearer',
```

```
          bearerFormat: 'JWT',
        },
      },
    },
    security: [{ bearerAuth: [] }],
  },
  apis: ['./src/routes/*.ts'],
};

const swaggerSpec = swaggerJsdoc(options);

export default function setupSwagger(app: e.Application): void
{
                app.use('/docs',                swaggerUi.serve,
swaggerUi.setup(swaggerSpec));
}
```

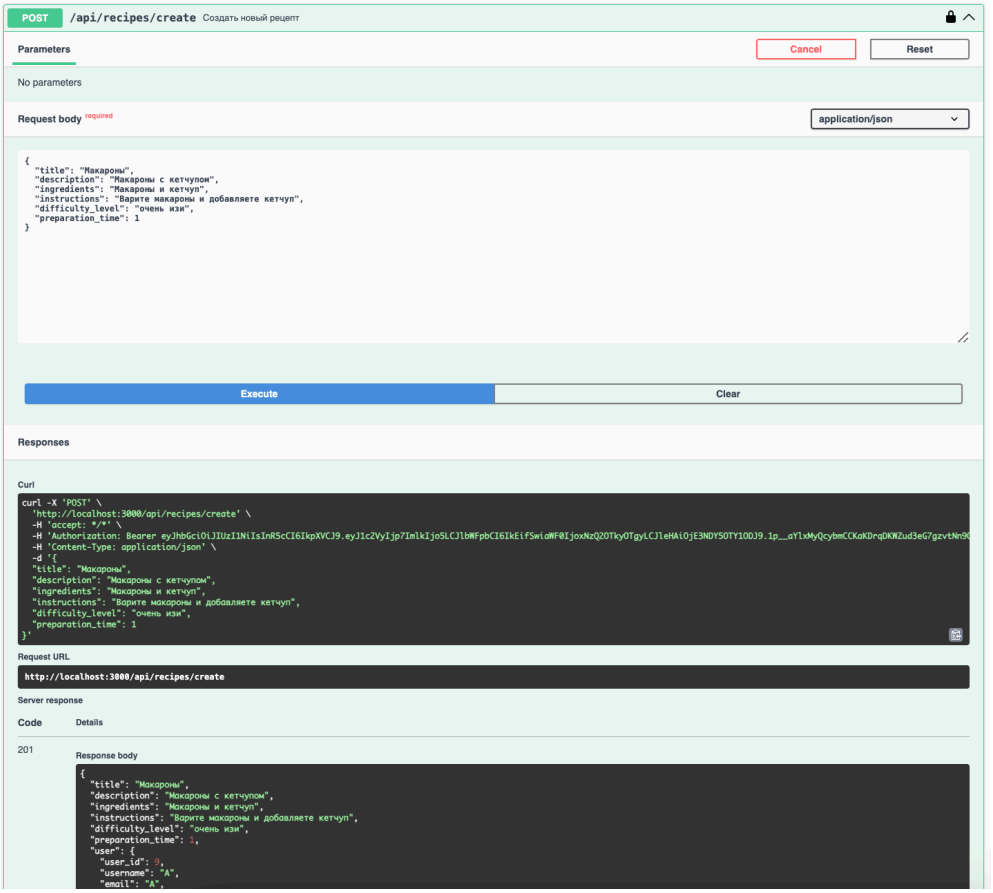
В папке routes хранятся аннотации OpenAPI для каждой сущности. Над каждым маршрутом были прописаны JSDoc-комментарии с тегом @openapi, чтобы Swagger мог автоматически прочитать и построить документацию по этим маршрутам.

Пример с получением всех статей:

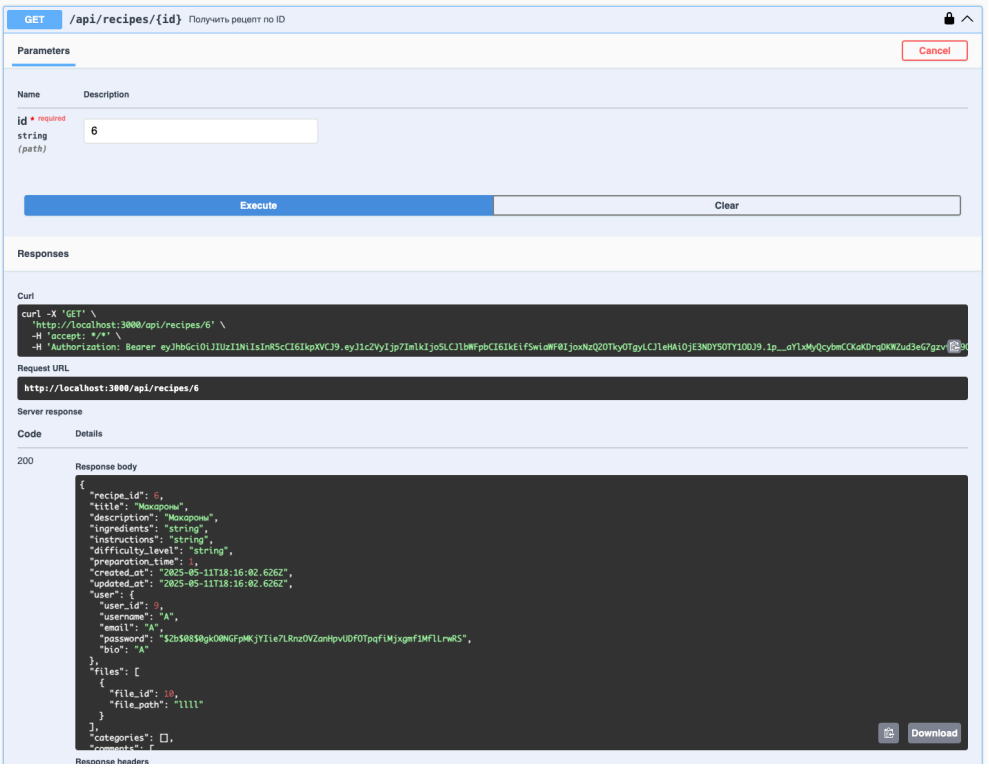
```
/**
* @openapi
* /api/articles:
*   get:
*     summary: Получить все статьи
*     tags:
*       - Articles
*     responses:
*       200:
*         description: Список статей
*/
router.get("/articles", articleController.getAllArticles);
```


{
"email": "Anna@example.com",
"password": "securePassword123"
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:3000/api/login' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"email": "Anna@example.com",
"password": "securePassword123"
}'
Request URL
http://localhost:3000/api/login
Server response
Code
Details
200
Response body
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7ImlkIjo3LCJlbWFpbCI6IkFubmFAZXhhbXBsZS5jb20ifSwiaWF0IjoxNzQ2OTg0ODIwLCJleHAiOjE3NDY5ODg0MjB9.4tFZCkWDvEaoWF-nIYLN9cR0FUVe1uM-6y8hQFe7Ze4"
}
Download
Response headers

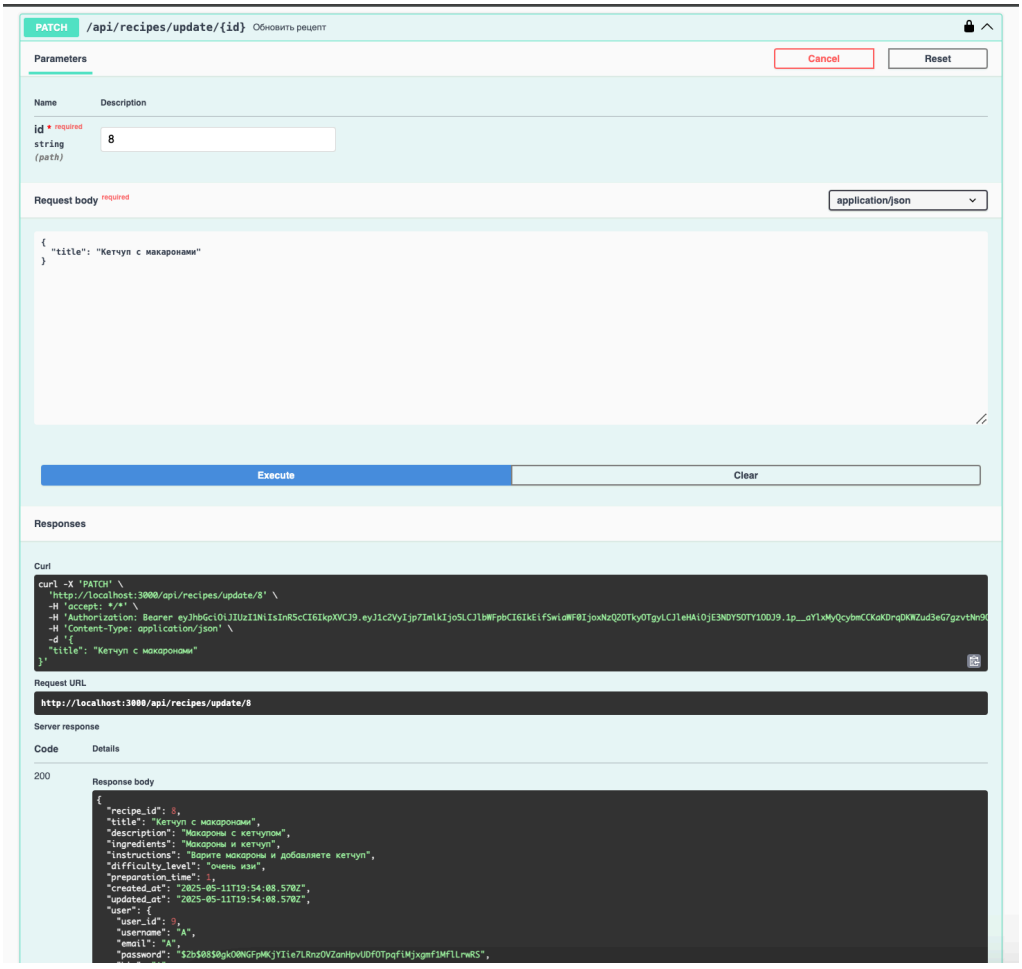
CREATE


POST
/api/recipes/create
Создать новый рецепт
Parameters
Cancel
Reset
No parameters
Request body
required
application/json
{
"title": "Макароны",
"description": "Макароны с кетчупом",
"ingredients": "Макароны и кетчуп",
"instructions": "Варите макароны и добавляете кетчуп",
"difficulty_level": "очень изи",
"preparation_time": 1
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:3000/api/recipes/create' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7ImlkIjo5LCJlbWFpbCI6IkEifSwiaWF0IjoxNzQ2OTkyOTgyLCJleHAiOjE3NDY5OTY10DJ9.1p__aYlxMyQcybmCCKaKDrqDKWZud3eG7gzvtNn9C
-H 'Content-Type: application/json' \
-d '{
"title": "Макароны",
"description": "Макароны с кетчупом",
"ingredients": "Макароны и кетчуп",
"instructions": "Варите макароны и добавляете кетчуп",
"difficulty_level": "очень изи",
"preparation_time": 1
}'
Request URL
http://localhost:3000/api/recipes/create
Server response
Code
Details
201
Response body
{
"title": "Макароны",
"description": "Макароны с кетчупом",
"ingredients": "Макароны и кетчуп",
"instructions": "Варите макароны и добавляете кетчуп",
"difficulty_level": "очень изи",
"preparation_time": 1,
"user": {
"user_id": 9,
"username": "A",
"email": "A",

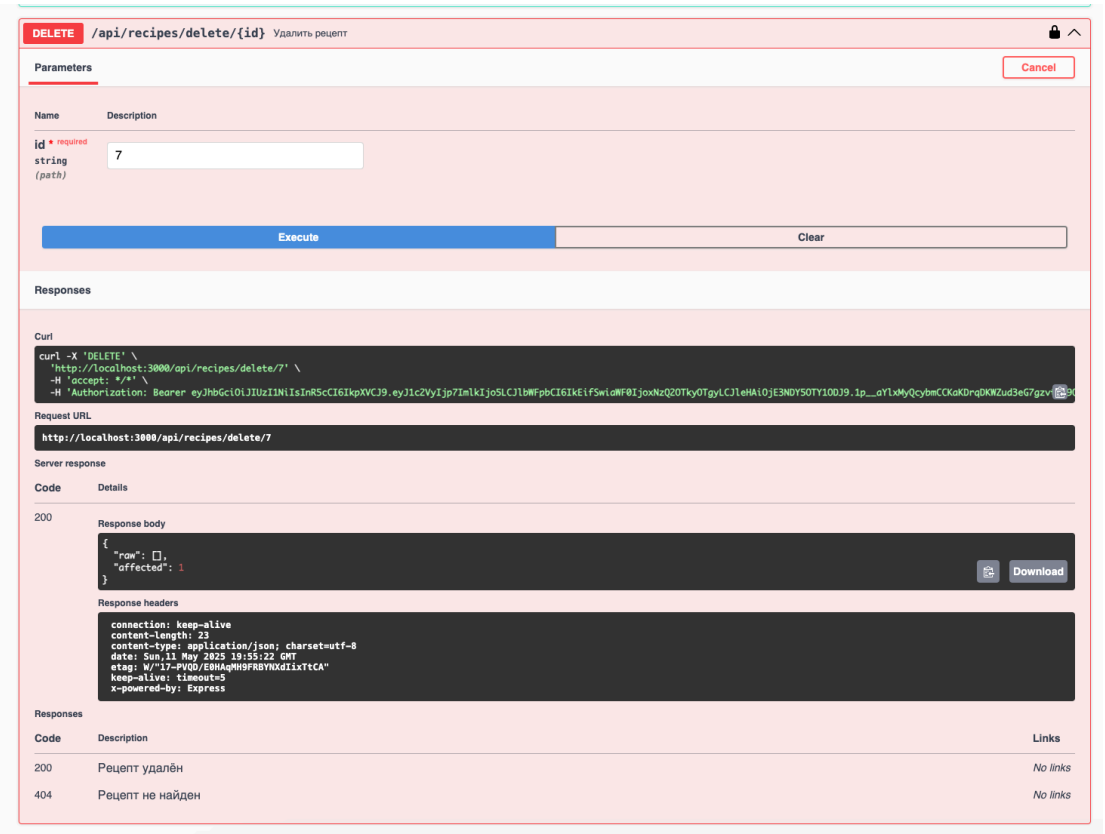
READ



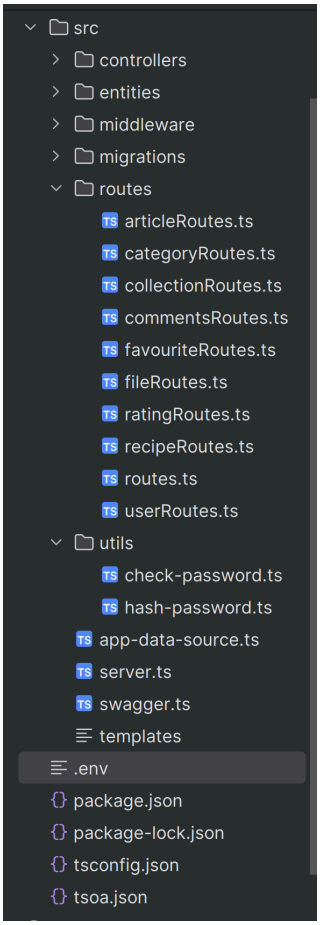
UPDATE



DELETE



Как выглядит структура на данном этапе:



Вывод: в рамках данной работы были протестированы все CRUD-операции с помощью Postman. Для API была создана документация с использованием Swagger. Также разобралась, как мы отображаем сгенерированную документацию средствами swaggerJsdoc и swaggerUi.