САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:

Габов Михаил

К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

1. Введение

Целью данной работы является тестирование разработанного API для платформы рецептов с использованием Postman. Была создана коллекция запросов для проверки основных сценариев использования API. Для автоматизации проверок были написаны тесты на JavaScript во вкладке "Tests" для каждого запроса.

Тестирование охватывает ключевые эндпоинты, включая создание пользователя, аутентификацию, создание и получение рецептов.

2. Структура коллекции и использование переменных

Для организации тестов была создана коллекция в Postman

Основные переменные:

- user_email: Сохраняет email созданного пользователя для последующего входа.
- user_password: Сохраняет пароль созданного пользователя.
- jwt_token: Сохраняет JWT, полученный после успешной аутентификации, для использования в защищенных эндпоинтах.
- last_created_recipe_id: Сохраняет ID последнего созданного рецепта для проверки его наличия в других запросах.

3. Описание реализованных тестов

Ниже представлено описание запросов и тестов, реализованных для проверки API.

3.1. Создание пользователя (POST /users)

Этот запрос регистрирует нового пользователя в системе.

Тесты:

- Проверка, что API возвращает статус 201 Created при успешном создании.
- Проверка, что тело ответа является валидным JSON.
- Сохранение email и пароля из тела запроса в переменные окружения для

использования в запросе на аутентификацию.

```
pm.test("Status code is 201 Created", function () {
    pm.response.to.have.status(201);
});

pm.test("Response is a valid JSON", function () {
    pm.response.to.be.json;
});

const requestBody = JSON.parse(pm.request.body.raw);

pm.environment.set("user_email", requestBody.email);
pm.environment.set("user_password", requestBody.password);

console.log("Сохранен email для логина:", requestBody.email);
```

3.2. Аутентификация пользователя (POST /users/login)

Этот запрос выполняет вход пользователя в систему и получает токен для доступа к защищенным ресурсам.

Тесты:

- Проверка, что API возвращает статус 200 OK.
- Проверка, что ответ содержит поле token и его значение является строкой.
- Сохранение полученного JWT в переменную окружения jwt_token.

```
pm.test("Status code is 200 OK", function () {
    pm.response.to.have.status(200);
});

const responseJson = pm.response.json();

pm.test("Response contains a token", function () {
```

```
    pm.expect(responseJson).to.have.property('token');
    pm.expect(responseJson.token).to.be.a('string');
});

if (responseJson.token) {
    pm.environment.set("jwt_token", responseJson.token);
    console.log("JWT токен успешно сохранен.");
}
```

3.3. Создание рецепта (POST /recipes)

Этот запрос создает новый рецепт. Для доступа к этому эндпоинту требуется аутентификация.

Тесты:

- Проверка, что API возвращает статус 201 Created.
- Проверка, что в ответе содержится объект созданного рецепта со свойством id.
- Сохранение id созданного рецепта в переменную last_created_recipe_id для последующих проверок.

```
pm.test("Status code is 201 Created", function () {
    pm.response.to.have.status(201);
});

const responseJson = pm.response.json();

pm.test("Response contains created recipe with an ID", function () {
    pm.expect(responseJson).to.have.property('id');
    pm.expect(responseJson.id).to.be.a('number');
});

// Сохраняем ID для следующих тестов
if (responseJson.id) {
```

```
    pm.environment.set("last_created_recipe_id", responseJson.id);
    console.log("ID созданного рецепта сохранен:", responseJson.id);
}
```

3.4. Получение списка рецептов (GET /recipes)

Этот запрос получает список всех рецептов.

Тесты:

- Проверка, что API возвращает статус 200 OK.
- Проверка, что тело ответа является массивом.
- Проверка, что массив рецептов не пустой.
- Проверка наличия в списке только что созданного рецепта (с использованием last_created_recipe_id).

```
pm.test("Status code is 200 OK", function () {
    pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
    pm.expect(pm.response.json()).to.be.an('array');
});

const responseJson = pm.response.json();

pm.test("Response array contains at least one recipe", function () {
    pm.expect(responseJson.length).to.be.at.least(1);
});

pm.test("Newly created recipe is in the list", function () {
    const lastCreatedRecipeId =
parseInt(pm.environment.get("last_created_recipe_id"));
    if (lastCreatedRecipeId) {
        const found = responseJson.some(recipe => recipe.id ===
```

```
lastCreatedRecipeId);
        pm.expect(found, "Только что созданный рецепт должен быть в списке").to.be.true;
    } else {
        console.warn("Skipping 'Newly created recipe is in the list' test: last_created_recipe_id not set.");
    }
});
```

4. Результаты тестирования

Все тесты были успешно выполнены. Результаты прогона коллекции подтверждают, что протестированные эндпоинты работают корректно в соответствии с ожиданиями:

- Статусы ответов соответствуют ожидаемым.
- Структура JSON-ответов корректна.
- Данные, созданные в одном запросе, успешно извлекаются и проверяются в последующих запросах.

5. Заключение

В ходе выполнения домашнего задания была создана и настроена коллекция Postman для тестирования API