САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа № 4

Выполнил:

Гуторова Инна

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- реализовать тестирование API средствами Postman;
- написать тесты внутри Postman.

Ход работы

Использовалась коллекция Travel Service API из домашней работы № 3.

Добавлены переменные окружения для удобства тестирования:

- {{baseUrl}} – базовый URL API (например, http://localhost:5000)
- {{authToken}} – JWT-токен (получаемый при логине)

Для каждого запроса реализована проверка статуса – 200 или 201.

```
pm.test("Response status code is 200", function () {
    pm.expect(pm.response.code).to.equal(200);
});
```

Также тесты проверяют время запроса и заголовки:

```
pm.test("Response time is less than 500ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});

pm.test("Content-Type is application/json", function () {
    pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});
```

Далее рассмотрим несколько конкретных примеров.

Для метода POST/…/api/auth/register также добавлены проверки, что вернулся json и он содержит сообщение успеха.

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

pm.test("Response contains success message", function () {
    const response = pm.response.json();
    pm.expect(response).to.have.property("message", "User registered successfully");
});
```

Для метода POST/…/api/auth/login добавлены проверки, что мы получаем токен и он имеет верную структуру.

```
pm.test("Response status code is 200", function () {
    pm.expect(pm.response.code).to.equal(200);
});

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

pm.test("Response should have the required field 'token'", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData).to.have.all.keys('token');
});


pm.test("Token must be a non-empty string", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData.token).to.exist.and.to.be.a('string').and.to.have.lengthOf.at.least(1, "Token should not be
        empty");
});

pm.test("JWT token has valid structure", function () {
    const token = pm.response.json().token;
    const parts = token.split(".");
    pm.expect(parts.length).to.eql(3);
});
```


Для метода POST/…/api/auth/update-password добавлены проверки ответа и передачи токена авторизации.

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

pm.test("Response contains a message field", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData).to.have.property('message');
});

pm.test("Password updated successfully", function () {
    const response = pm.response.json();
    pm.expect(response).to.have.property("message", "Password updated successfully");
});
```

```
pm.test("Authorization header is present", function () {
    pm.expect(pm.request.headers.get("Authorization")).to.exist;
    pm.expect(pm.request.headers.get("Authorization")).to.include("Bearer ");
});
```

Далее реализацию тестов для методов с сущностями рассмотрим на примере attraction. Для всех методов дублируются тесты проверки статуса, времени запроса, заголовков и авторизации, если она требуется. Ниже приведены тесты проверки структуры тела ответа.

POST/…/api/attractions

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

pm.test("Response has correct structure", function () {
    const response = pm.response.json();
    pm.expect(response).to.have.all.keys([
        "name",
        "description",
        "location",
        "route",
        "attraction_id"
    ]);
    pm.expect(response.name).to.be.a("string");
    pm.expect(response.description).to.be.a("string");
    pm.expect(response.location).to.be.a("string");
    pm.expect(response.route).to.be.a("number");
    pm.expect(response.attraction_id).to.be.a("number");
});
```

GET/…/api/attractions

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

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

pm.test("Each attraction has correct structure", function() {
    const attractions = pm.response.json();

    attractions.forEach(attraction => {
        pm.expect(attraction).to.have.all.keys(
            'attraction_id',
            'name',
            'description',
            'location',
            'route'
        );

        pm.expect(attraction.attraction_id).to.be.a('number');
        pm.expect(attraction.name).to.be.a('string');
        pm.expect(attraction.description).to.be.a('string');
        pm.expect(attraction.location).to.be.a('string');
```

```
        pm.expect(attraction.route).to.be.an('object');
        pm.expect(attraction.route).to.have.all.keys(
            'route_id',
            'title',
            'description',
            'price',
            'duration'
        );

        pm.expect(attraction.route.route_id).to.be.a('number');
        pm.expect(attraction.route.title).to.be.a('string');
        pm.expect(attraction.route.description).to.be.a('string');
        pm.expect(attraction.route.price).to.be.a('string');
        pm.expect(attraction.route.duration).to.be.a('string');
    });
});
```

GET/…/api/attractions/:id

Аналогичная проверка структуры, как в запросе выше, плюс проверка соответствия id

```
pm.test("Returned attraction_id matches requested ID", function() {
    const attraction = pm.response.json();
    const fullUrl = pm.request.url.toString();
    const urlParts = fullUrl.split('/');
    const requestedId = urlParts[urlParts.length - 1];
    pm.expect(attraction.attraction_id.toString()).to.equal(requestedId);
});
```

PUT/…/api/attractions/:id

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

pm.test("Response has correct structure", function() {
    const response = pm.response.json();
    pm.expect(response).to.have.all.keys(
        'attraction_id',
        'name',
        'description',
        'location',
        'route'
    );
});
```

```
pm.test("Data was updated correctly", function() {
    const response = pm.response.json();
    const requestData = JSON.parse(pm.request.body.raw);

    pm.expect(response.name).to.eql(requestData.name);
    pm.expect(response.description).to.eql(requestData.description);
    pm.expect(response.location).to.eql(requestData.location);
    pm.expect(response.route).to.eql(requestData.route);
});

pm.test("Attraction ID matches URL parameter", function() {
    const response = pm.response.json();
    const fullUrl = pm.request.url.toString();
    const idMatch = fullUrl.match(/\/(\d+)(?:\?|$)/);

    pm.expect(idMatch, "ID not found in URL").to.exist;
    pm.expect(response.attraction_id.toString()).to.equal(idMatch[1]);
});
```

DELETE/…/api/attractions/:id

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

pm.test("Success message exists", function() {
    const response = pm.response.json();
    pm.expect(response).to.have.property('message', 'Attraction deleted');
});
```

Аналогичным образом реализовано тестирование для всех сущностей.

Вывод

Проведенное тестирование REST API для сервиса организации путешествий подтвердило работоспособность базовой функциональности, включая авторизацию и CRUD-операции для всех сущностей.