

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет
нафти і газу

Кафедра інженерії програмного забезпечення

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни

Основи автоматизованого тестування

Виконав: ст. гр. ІІ-23-1К

Гнатюк Д. М.

Івано-Франківськ

2025

Тема: Тестування REST API за допомогою Jest та Axios

Мета: Ознайомитися з клієнтами для виконання HTTP-запитів, обрати один з них, реалізувати тестування API-запитів з використанням Jest та Axios, написати по 5 тестів до кожного з основних методів (GET, POST, PUT, DELETE) і виконати інтеграційне тестування з використанням публічного API.

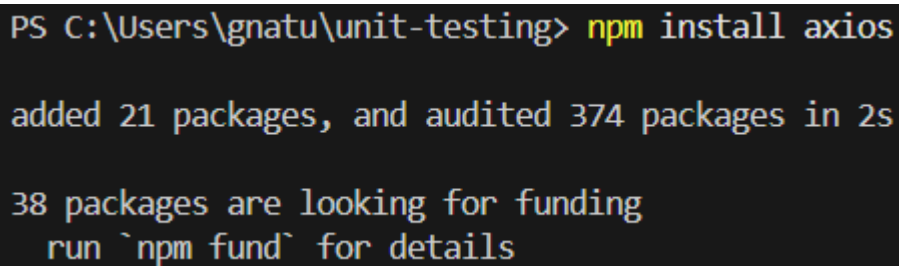
Посилання на Github: <https://github.com/DementialDima/Lab1-AutomatedTesting>

Обрані інструменти

- Клієнт для HTTP-запитів: Axios
- Тестовий фреймворк: Jest
- API: <https://jsonplaceholder.typicode.com>

Було виконано ініціалізацію проєкту, встановлено необхідні залежності:

`npm install axios`



```
PS C:\Users\gnatu\unit-testing> npm install axios
added 21 packages, and audited 374 packages in 2s
38 packages are looking for funding
run `npm fund` for details
```

Простий GET-запит для перевірки:

```
import axios from 'axios';

axios.get('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });
```

Тестування API

Було написано по 5 тестів до кожного з методів API:

```
import axios from 'axios';

const baseUrl = 'https://jsonplaceholder.typicode.com';

describe('API Integration Tests', () => {
  let postId;

  test('GET all posts', async () => {
    const response = await axios.get(`${baseUrl}/posts`);
    expect(response.status).toBe(200);
    expect(Array.isArray(response.data)).toBe(true);
  });

  test('GET specific post', async () => {
```

```

    const response = await axios.get(`${baseUrl}/posts/1`);
    expect(response.status).toBe(200);
    expect(response.data.id).toBe(1);
  });

  test('POST create new post', async () => {
    const newPost = { title: 'Test', body: 'Body', userId: 1 };
    const response = await axios.post(`${baseUrl}/posts`, newPost);
    expect(response.status).toBe(201);
    expect(response.data).toMatchObject(newPost);
    postId = response.data.id;
  });

  test('PUT update post', async () => {
    const update = { title: 'Updated Title', body: 'Updated Body', userId: 1 };
    const response = await axios.put(`${baseUrl}/posts/1`, update);
    expect(response.status).toBe(200);
    expect(response.data).toMatchObject(update);
  });

  test('DELETE post', async () => {
    const response = await axios.delete(`${baseUrl}/posts/1`);
    expect(response.status).toBe(200);
  });
});

```

Результат виконання:

```

PS C:\Users\gnatu\unit-testing> npm test

> integration-testing@1.0.0 test
> jest

PASS ./api.test.js (5.002 s)
  API Integration Tests - JSONPlaceholder
    ✓ GET all posts - returns array (141 ms)
    ✓ GET all users - returns array (30 ms)
    ✓ GET all comments - returns array (38 ms)
    ✓ GET all albums - returns array (29 ms)
    ✓ GET all todos - returns array (28 ms)
    ✓ GET post with ID 1 (32 ms)
    ✓ GET user with ID 1 (30 ms)
    ✓ GET comment with ID 1 (367 ms)
    ✓ GET album with ID 1 (28 ms)
    ✓ GET todo with ID 1 (27 ms)
    ✓ POST new post (140 ms)
    ✓ POST new comment (138 ms)
    ✓ POST new album (179 ms)
    ✓ POST new photo (359 ms)
    ✓ POST new todo (139 ms)
    ✓ PUT update post (139 ms)
    ✓ PUT update comment (140 ms)
    ✓ PUT update album (361 ms)
    ✓ PUT update todo (362 ms)
    ✓ PUT update photo (363 ms)
    ✓ DELETE post (359 ms)
    ✓ DELETE comment (141 ms)
    ✓ DELETE album (139 ms)
    ✓ DELETE todo (140 ms)
    ✓ DELETE photo (137 ms)

Test Suites: 1 passed, 1 total
Tests: 25 passed, 25 total
Snapshots: 0 total
Time: 5.059 s
Ran all test suites.
PS C:\Users\gnatu\unit-testing>

```

Висновок: У ході лабораторної роботи було отримано практичні навички роботи з Axios та Jest, а також з REST API. Було реалізовано повний цикл інтеграційного тестування з використанням публічного API сервісу JSONPlaceholder. Тестування продемонструвало здатність інструментів ефективно обробляти HTTP-запити та перевіряти коректність відповідей.

Відповіді на запитання

1. Що таке Арі?

API (Application Programming Interface) — це набір правил, який дозволяє різним програмам обмінюватися даними між собою. Наприклад, фронтенд сайту звертається до бекенду через API.

2. Що таке Арі тестування?

API тестування — це перевірка роботи API-запитів (GET, POST, PUT, DELETE тощо) без використання графічного інтерфейсу. Мета — переконатися, що API працює коректно і повертає правильні відповіді.

3. Які Арі клієнти вам відомі?

- Axios
- fetch (вбудований у браузер)
- Postman (графічний клієнт)
- cURL
- Insomnia
- Supertest (для Node.js)

4. Які методи для формування запитів вам відомі?

- GET — отримати дані
- POST — створити новий запис
- PUT — оновити існуючий запис
- DELETE — видалити запис
- PATCH — часткове оновлення
- HEAD, OPTIONS — службові запити

5. Які статус коди вам відомі?

- 200 OK — запит виконано успішно

- 201 Created — створено новий ресурс
- 400 Bad Request — помилка клієнта
- 401 Unauthorized — потрібна авторизація
- 403 Forbidden — доступ заборонено
- 404 Not Found — ресурс не знайдено
- 500 Internal Server Error — помилка сервера

6. Які позитивні сторони тестування інтеграції ви можете назвати?

- Перевірка взаємодії між модулями
- Раннє виявлення помилок у логіці
- Підвищення надійності системи
- Не вимагає UI

7. Які негативні сторони тестування інтеграції вам відомо?

- Може бути складним у налаштуванні
- Залежить від стабільності зовнішніх API
- Важко тестувати, якщо API ще не завершено
- Тести можуть бути повільнішими за юніт-тести

8. З яких елементів складатиметься простий API тест?

- URL запити
- Метод (GET, POST тощо)
- Тіло запити (для POST/PUT)
- Очікуваний статус-код
- Очікувані дані у відповіді

9. Що таке hook в розрізі тестів?

Hook — це функція, яка виконується до або після тестів. Використовується для підготовки або очищення середовища.

10. Які hook и Вам відомі?

beforeAll() — виконується один раз перед усіма тестами

afterAll() — після всіх тестів

beforeEach() — перед кожним тестом

afterEach() — після кожного тесту