



ТЕМА 11

Тестування та документування API

Unit/functional тесту · Postman/Insomnia · OpenAPI/Swagger

Бекенд-розробка, сервіси та мікросервіси

Комп'ютерна інженерія · 3 курс · Лекція 11 з 18

7 питань сьогоднішньої теми

01

Піраміда тестування

02

**Unit-тестування з
PHPUnit**

03

**Функціональне
тестування API**

04

Postman та Insomnia

05

**Навіщо документувати
API**

06

**Структура специфікації
OpenAPI**

07

**Практика: опис
ендпоінту**

Три рівні тестів



Чим нижче рівень — тим більше тестів і швидше вони виконуються.

Шаблон AAA: Arrange, Act, Assert

```
use PHPUnit\Framework\TestCase;

class PriceCalculatorTest extends TestCase {
    public function testAddsTaxCorrectly(): void {
        // Arrange
        $calculator = new PriceCalculator();

        // Act
        $result = $calculator->addTax(100, 0.2);

        // Assert
        $this->assertEquals(120, $result);
    }
}
```

Основні методи перевірки

Метод	Перевіряє
<code>assertEquals(\$a, \$b)</code>	Значення рівні (нежорстке порівняння)
<code>assertSame(\$a, \$b)</code>	Значення й тип ідентичні (===)
<code>assertTrue()</code> / <code>assertFalse()</code>	Значення є true / false
<code>assertCount(\$n, \$arr)</code>	Масив містить рівно \$n елементів
<code>expectException(\$class)</code>	Код кидає вказаний виняток

Мокування залежностей

```
public function testFindByIdReturnsNullWhenNotFound(): void {  
    $pdoMock = $this->createMock(PDO::class);  
    $stmtMock = $this->createMock(PDOStatement::class);  
  
    $stmtMock->method('fetch')->willReturn(false);  
    $pdoMock->method('prepare')->willReturn($stmtMock);  
  
    $repository = new ProductRepository($pdoMock);  
    $result = $repository->findById(999);  
  
    $this->assertNull($result);  
}
```

Мокати варто межі системи — базу даних, мережеві запити, файлову систему.

Реальний HTTP-запит до застосунку

```
class ProductApiTest extends TestCase {  
    private Client $client;  
  
    protected function setUp(): void {  
        $this->client = new Client(['base_uri' => 'http://localhost:8000']);  
    }  
  
    public function testGetProductReturns200(): void {  
        $response = $this->client->get('/api/products/1');  
        $this->assertEquals(200, $response->getStatusCode());  
  
        $data = json_decode($response->getBody(), true);  
        $this->assertArrayHasKey('name', $data);  
    }  
}
```

Типовий робочий процес у Postman

- 1 Обрати HTTP-метод зі списку
- 2 Ввести URL ендпоінта
- 3 Додати заголовки (напр. Authorization)
- 4 Заповнити тіло запиту у форматі JSON
- 5 Send — переглянути статус і відповідь

Порівняння інструментів

Інструмент	Особливості
Postman	Найпопулярніший, хмарна синхронізація, тести на JavaScript
Insomnia	Легший інтерфейс, зручний для GraphQL, відкритий код

```
pm.test("Status is 200", () => pm.response.to.have.status(200));
```

Простий тестовий скрипт у вкладці Tests — проміжний крок між ручним тестуванням і PHPUnit.

Проблема ручної документації

Текстова документація (Word, вікі) швидко застаріває: код змінюється, опис — ні. Розробники фронтенду й мобільних застосунків не мають актуальної інформації про API.

Рішення: OpenAPI — машинозчитуваний опис, з якого автоматично генерується наочна, актуальна документація.

OpenAPI vs Swagger

OpenAPI

Стандарт (формат) опису API

YAML/JSON-файл, що формально описує ендпоінти, параметри, схеми даних

Swagger

Набір інструментів

Swagger UI, Swagger Editor — генерують інтерактивну документацію з файлу OpenAPI

Опис ендпоінту: paths

```
paths:
  /products/{id}:
    get:
      summary: Отримати товар за id
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: Товар знайдено
        '404':
          description: Товар не знайдено
```

Опис моделі даних: components/schemas

```
components:  
  schemas:  
    Product:  
      type: object  
      properties:  
        id:  
          type: integer  
        name:  
          type: string  
        price:  
          type: number
```

`$ref: '#/components/schemas/Product'` — посилання замість дублювання опису в кожному ендпоінті.

Основні розділи специфікації

Розділ	Призначення
info	Назва, версія та опис API загалом
paths	Перелік ендпоінтів, методів, параметрів і відповідей
components/schemas	Багаторазово використовувані описи структур даних
components/securitySchemes	Опис способу автентифікації (напр. Bearer JWT)

POST /products — повний приклад

```
/products:
  post:
    summary: Створити новий товар
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required: [name, price]
            properties:
              name: { type: string }
              price: { type: number }
    responses:
      '201': { description: Товар створено }
      '422': { description: Помилка валідації }
```

Підсумок теми 11

- Піраміда тестування: багато unit-тестів, менше функціональних, мало E2E
- Unit-тест перевіряє одну функцію ізольовано за шаблоном Arrange-Act-Assert
- Мокування підміняє зовнішні залежності (БД, мережа) тестовими дублерами
- Функціональний тест надсилає реальний HTTP-запит і перевіряє відповідь
- Postman/Insomnia — інструменти ручного тестування API без написання коду
- OpenAPI дозволяє автоматично генерувати актуальну документацію API