Дизайн контролерів для взаємодії з API

Познайомтеся з концепцією та роллю контролерів в автоматизації API тестування. Дізнайтеся, як правильне проектування коду може значно покращити якість та ефективність вашої роботи з API.

Що таке контролери в автоматизації API?

Визначення та функції

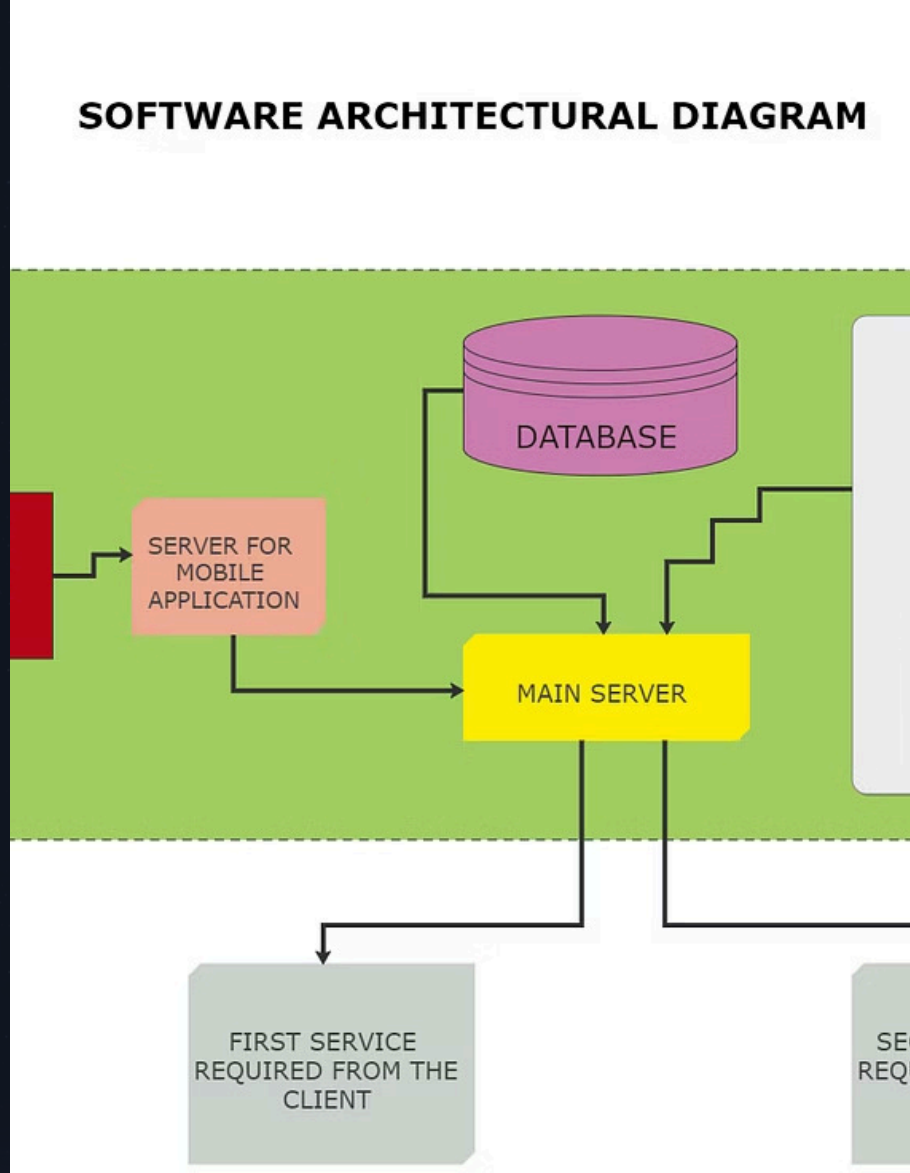
Контролер — це спеціалізований модуль коду, який інкапсулює логіку виконання запитів до API з використанням бібліотек типу Axios або Fetch. Контролери централізовано зберігаються в одному місці та використовуються там, де це необхідно, забезпечуючи єдину точку доступу до API операцій.

Це архітектурний патерн, який дозволяє відокремити бізнес-логіку тестів від технічних деталей виконання HTTP-запитів.

Типи контролерів

Контролери можна організувати за різними принципами:

- За HTTP методами (GET, POST, PUT, DELETE)
- За специфічними ендпойнтами
- За функціональними доменами (користувачі, продукти, замовлення)
- За версіями API (v1, v2)



Переваги використання контролерів

Організація коду

Контролери допомагають організовувати логіку API у модульний і структурований спосіб. Такий розподіл відповідальності призводить до більш чистого, читабельного та зручного для підтримки коду.

Масштабованість

З контролерами легше керувати та масштабувати запити API. У міру розширення проекту контролери ефективно обробляють зростаючу кількість запитів і складніших операцій.

Обслуговування

Контролери спрощують процес оновлення та підтримки API. Зміни можна вносити в централізованому місці, не впливаючи на інші частини програми.

Безпека

Контролери діють як перша лінія захисту, централізовано обробляючи автентифікацію та авторизацію, покращуючи загальну безпеку API взаємодій.

Повторне використання

Загальні функції можна абстрагувати в контролери та використовувати в різних частинах програми, значно зменшуючи дублювання коду.

Гнучкість

Контролери організовано обробляють різні типи запитів (GET, POST, PUT, DELETE), забезпечуючи гнучкість у взаємодії з API.

Практичний приклад: підхід без контролерів

Розглянемо традиційний підхід до тестування API без використання контролерів. У цьому прикладі ми виконуємо звичайний GET-запит для отримання інформації про книгу за її ID:

```
test('get book by id', async () => {
  const response = await axios.get(
    'https://demoqa.com/books?book=9781449365035'
  )
  expect(response.status).toEqual(200);
})
```

Проблема: Якщо у вас лише один запит, цей підхід може здатися простим. Однак, по мірі зростання проекту та збільшення кількості тестів, ви почнете дублювати подібні запити, які відрізняються лише ID або іншими параметрами в URL.

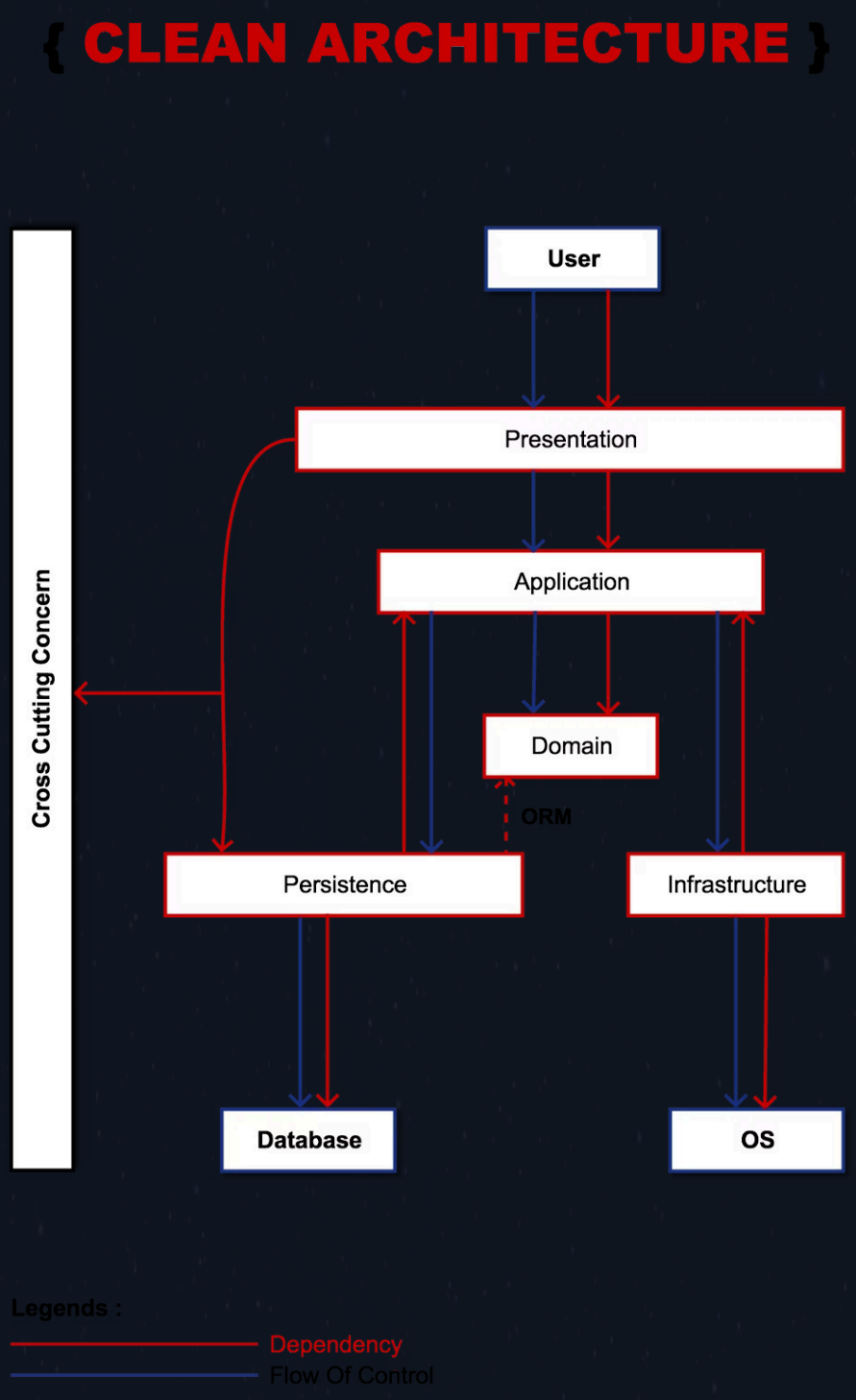
Такий підхід призводить до:

- Дублювання коду в різних тестах
- Складності в підтримці при зміні базового URL або структури запиту
- Відсутності централізованої валідації відповідей
- Ускладнення рефакторингу при оновленні API

Рішення: впровадження контролерів

Створення контролера

Для вирішення проблеми дублювання коду створюємо окремий файл з реалізацією запитів і валідацією відповідей:



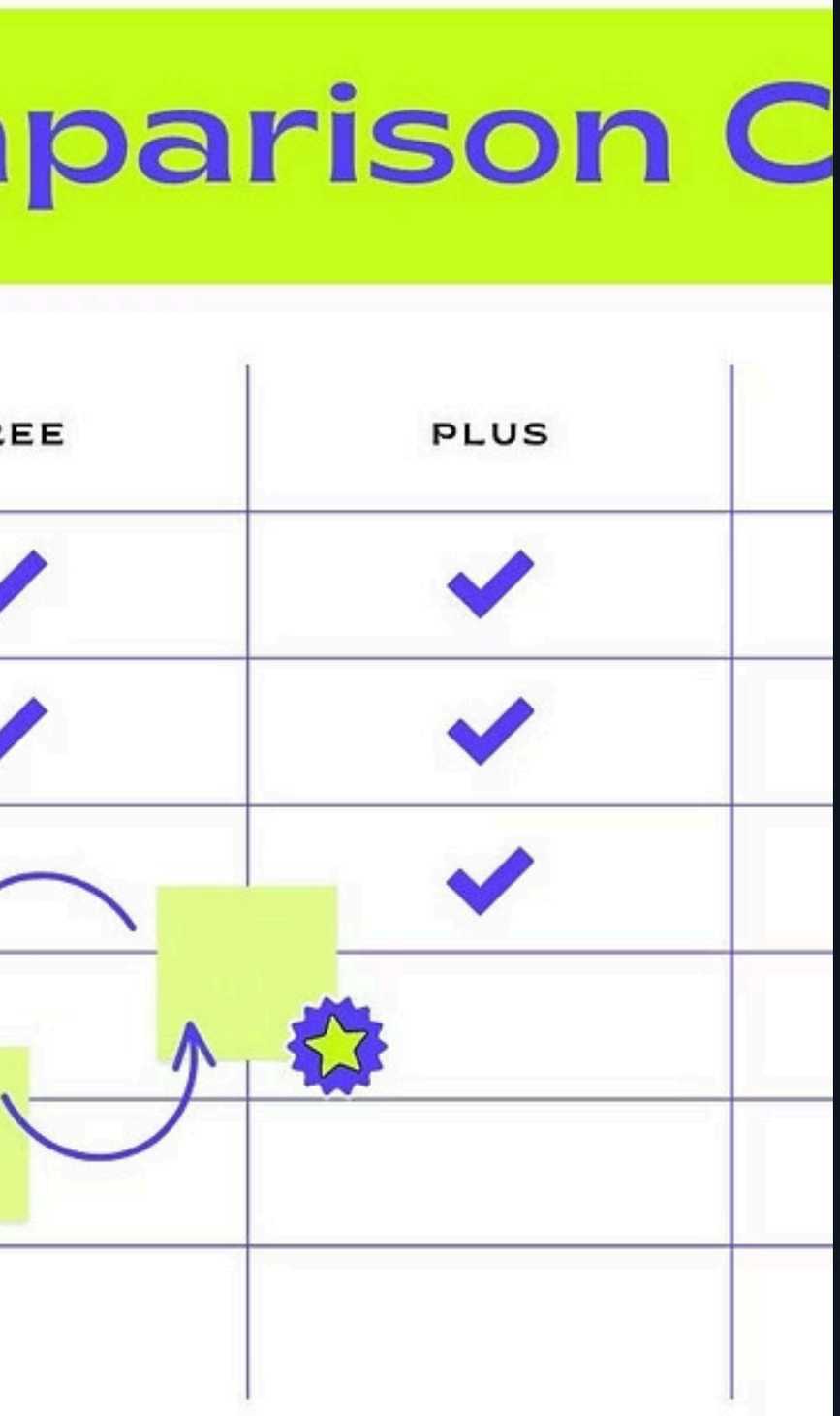
controllers.js

```
const axios = require('axios')

module.exports = class CustomControllers {
  async findBookById(bookId) {
    const params = new URLSearchParams([
      ['key', 'book'],
      ['value', bookId]
    ])
    const response = await axios.get(
      'https://demoqa.com/books',
      { params }
    )
    expect(response.status).toEqual(200);
    return response;
  }
}
```

Використання в тестах

```
const CustomControllers = require('./controllers.js')
const controllers = new CustomControllers()

test('get book by id', async () => {
  const response = await controllers
    .findBookById('9781449365035')
})
```

Тепер запит інкапсульований у методі, який приймає ID книги як параметр. Це дозволяє легко використовувати той самий запит з різними параметрами в багатьох тестах.



Порівняльний аналіз підходів

Без контролерів

- Підходить для малих проектів
- Швидкий старт розробки
- Дублювання коду
- Складність масштабування

З контролерами

- Централізоване управління
- Легка підтримка коду
- Повторне використання
- Професійна архітектура

Підтримка

Контролери спрощують підтримку коду — всі зміни вносяться в одному центральному місці, автоматично застосовуючись до всіх тестів.

Лаконічність

Код виглядає більш лаконічним і професійним, що полегшує роботу в команді та онбординг нових розробників.

Гнучкість

Підхід без контролерів добре підходить лише для одного-двох запитів, де не потрібно створювати схожих варіацій.

> **Висновок:** Використання контролерів — це не просто технічне рішення, а стратегічний підхід до побудови надійної, масштабованої та підтримуваної системи автоматизації API тестування.