# 🧪 Guía Práctica – Testing de APIs con `requests`

> **Objetivo:** aprender a probar APIs REST usando Python + `requests`, validar respuestas, manejar errores y automatizar pruebas.

---

## 🏁 Estructura básica para probar un endpoint

```
import requests

url = "https://api.example.com/items"
response = requests.get(url)

print("Status:", response.status_code)
print("Body:", response.json())
```

---

## ✔ Validar código de estado (lo más básico)

```
assert response.status_code == 200, "La API no devolvió 200 OK"
```

---

## ✔ Validar contenido JSON

```
data = response.json()

assert "id" in data, "La respuesta no tiene 'id'"
assert isinstance(data["id"], int), "'id' no es un número"
```

---

## ✔ Validar lista de resultados

```
assert isinstance(data, list), "La API no devolvió una lista"
assert len(data) > 0, "La lista vino vacía"
```

---

## 🚀 Probar un POST que crea un recurso

```
payload = {
    "title": "Test desde Python",
```

```
    "body": "Hola Axel probando API",
    "userId": 1
}

response = requests.post(
    "https://jsonplaceholder.typicode.com/posts",
    json=payload
)

assert response.status_code == 201, "No se creó el recurso"
data = response.json()
assert data["title"] == payload["title"]
```

---

## 🧹 Probar un DELETE

```
response = requests.delete("https://jsonplaceholder.typicode.com/posts/1")
assert response.status_code in (200, 204), "El delete falló"
```

---

## 🧩 Probar errores esperados

Cuando la API debería devolver error:

```
response = requests.get("https://api.example.com/noexiste")

assert response.status_code == 404
```

---

## ⏱️ Configurar timeouts (muy usado en producción)

```
response = requests.get("https://api.example.com", timeout=3)
```

---

# 🔐 Testing con autenticación

## Token Bearer

```
headers = {"Authorization": "Bearer MI_TOKEN"}

response = requests.get(
    "https://api.example.com/secure",
    headers=headers
```

```
)

assert response.status_code == 200
```

---

### Basic Auth

```
from requests.auth import HTTPBasicAuth

response = requests.get(
    "https://api.example.com/auth",
    auth=HTTPBasicAuth("user", "pass")
)
```

---

## 🗀 Testing con archivos

### Subir archivo

```
files = {"file": open("ejemplo.pdf", "rb")}

response = requests.post(
    "https://api.example.com/upload",
    files=files
)

assert response.status_code == 200
```

---

### Descargar y verificar tamaño

```
response = requests.get("https://example.com/image.jpg")

assert len(response.content) > 1000, "La imagen es sospechosamente pequeña"
```

---

## 🧪 Organización recomendada para pruebas

```
import requests

BASE_URL = "https://jsonplaceholder.typicode.com"
```

```python
def test_get_post():
    r = requests.get(f"{BASE_URL}/posts/1")
    assert r.status_code == 200
    assert "title" in r.json()

def test_create_post():
    payload = {"title": "Testing", "body": "Demo", "userId": 1}
    r = requests.post(f"{BASE_URL}/posts", json=payload)
    assert r.status_code == 201
```

## 📦 Errores comunes y cómo resolverlos

| Error | Significa | Solución |
|---|---|---|
| `ConnectionError` | No conecta | Revisar URL o VPN |
| `Timeout` | El servidor tarda mucho | Aumentar timeout |
| `JSONDecodeError` | No vino JSON | Ver `response.text` |
| `403 / 401` | No autorizado | Revisar token / headers |
| `500` | Error del servidor | No es tu culpa: loguear y reportar |

## 🎯 Ejemplo de prueba completa (realista)

```python
import requests

BASE = "https://jsonplaceholder.typicode.com"

def test_api():
    print("Probando GET...")
    r = requests.get(f"{BASE}/posts/1")
    r.raise_for_status()  # falla si no es 200
    data = r.json()
    assert "id" in data

    print("Probando POST...")
    payload = {"title": "AxelTest", "body": "Hola", "userId": 1}
    r2 = requests.post(f"{BASE}/posts", json=payload)
    assert r2.status_code == 201

    print("🔥 Todo OK!")

test_api()
```