Análisis de caso

Aserciones

Ashley Rodriguez.

Situación inicial

Una empresa de tecnología llamada ApiSecure S.A. se encuentra desarrollando una API REST pública para la gestión de clientes. El equipo de QA está encargado de garantizar que cada endpoint cumpla con los requisitos funcionales y devuelva respuestas confiables y predecibles.

Hasta el momento, han estado realizando pruebas manuales en Postman, pero debido a un error en producción que no fue detectado a tiempo (un endpoint devolvía un código 200 OK con un mensaje de error en el cuerpo), la dirección del proyecto decidió implementar pruebas automatizadas con aserciones en Java.

La empresa utiliza Spring Boot y JUnit 5 como stack principal de desarrollo y testing.

1.Identificá los aspectos clave a testear por endpoin:

Endpoint	**Código de estado esperado**	**Estructura y contenido del cuerpo**
GET /clientes/{id}	200 OK (si existe), 404 Not Found (si no existe)	{ "id": 1, "nombre": "Ashley Rodriguez", "email": "ashley.rodriguez@correo.com" }
POST /clientes	201 Created (si éxito), 400 Bad Request (si faltan campos)	{ "id": 10, "nombre": "Ana", "email": "ana.z@correo.com" }
DELETE /clientes/{id}	204 No Content (si éxito), 404 Not Found (si no existe)	Sin cuerpo o {}

2. Elegí qué tipo de aserciones se aplicarán:

assertEquals → Comparar valores exactos
contains / not contains → Verificar que un string aparezca o no en el JSON
JSONPath → Extraer valores del cuerpo JSON y validarlos.
statusCode() → Validar código HTTP de respuesta.
header() → Validar headers obligatorios.
time() → Validar tiempo de respuesta.

3. Definí una prueba de ejemplo con código pseudojava o real, aplicando mínimo 4 tipos distintos de aserciones.

```
    // 1. GET - Verificación de existencia de usuario
    @Test
    void testGetClientePorId() {
        Response response =
            given()
                .log().all()
                .when()
                .get("/users/2")
                .then()
                .log().all()
                .statusCode(200)
                .contentType(ContentType.JSON)
                .time(lessThan(2000L))
                .extract().response();

        assertThat(response.jsonPath().getInt("data.id")).isEqualTo(2);
        assertThat(response.jsonPath().getString("data.email")).contains("@r
eqres.in");
    }
```

1. Código HTTP (statusCode())
2. Header (header())
3. Contenido JSON con JSONPath + AssertJ (id positivo)
4. Contenido JSON con AssertJ (email contiene "@")

4. Reflexión final
Las aserciones permiten

- Detectar errores funcionales antes de que lleguen a producción.
- Validar que el API cumpla con contratos esperados
- Mejorar la confiabilidad y la calidad del producto.
- Reducir la dependencia de pruebas manuales, acelerando la integración continua.