# Análisis de caso

Librería rest assured

Ashley Rodriguez.

**Situación inicial**

Una empresa de tecnología llamada DataFlow Corp ha desarrollado una nueva API REST pública para la gestión de usuarios de su sistema interno. La API permite registrar usuarios, obtener información de ellos, actualizar datos y eliminarlos. Esta API será consumida por aplicaciones móviles y sistemas internos, por lo que es fundamental garantizar su correcto funcionamiento antes del despliegue.

El equipo de QA ha decidido implementar pruebas automatizadas usando REST Assured con Java para validar los endpoints principales. Tu rol será diseñar y ejecutar las pruebas que aseguren que el comportamiento de la API es consistente, confiable y que responde adecuadamente ante errores comunes.

**Endpoints a probar**

- GET /clientes/{id}
- POST /clientes
- DELETE /clientes/{id}
-

**Aspectos clave a testear Por cada endpoint se deben validar:**

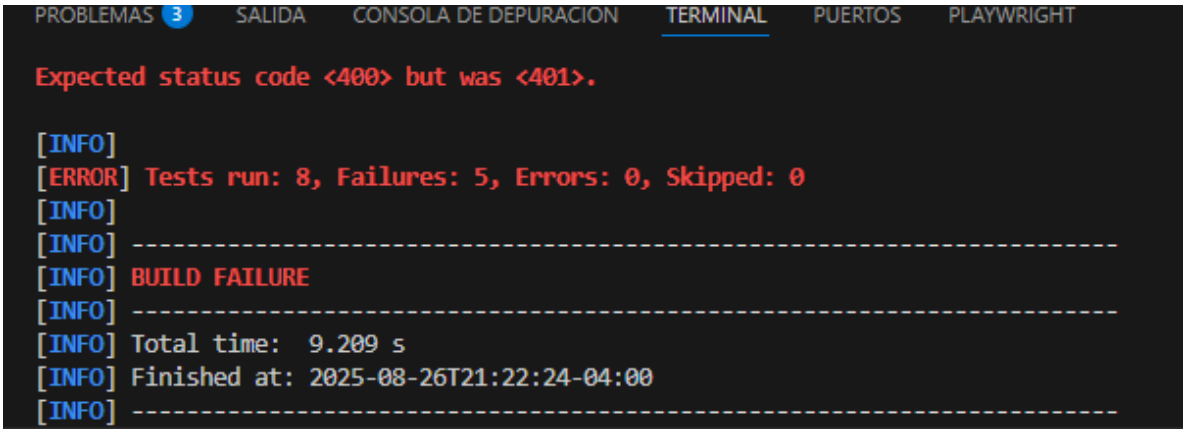
- Código de estado esperado (200, 201, 204, 404, 400).
- Encabezados obligatorios como Content-Type: application/json.
- Estructura y contenido del cuerpo de respuesta con JSONPath

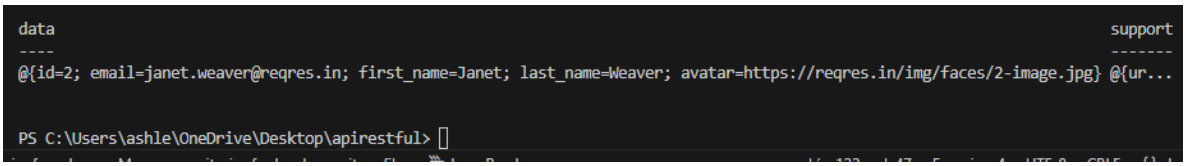
```
class ClienteApiTest {

    @BeforeAll
    static void setup() {
        // Base pública de ReqRes
        RestAssured.baseURI = "https://reqres.in/api";
        // Evita problemas de SSL en entornos corporativos
        RestAssured.useRelaxedHTTPSValidation();
    }



    // 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");
    }

    // 2. GET – Usuario inexistente (404)
    @Test
    void testGetUsuarioInexistente() {
        given()
            .log().all()
            .when()
            .get("/users/23")
            .then()
            .log().all()
            .statusCode(404)
            .time(lessThan(2000L));
    }

    // 3. POST – Registro exitoso
    @Test
    void testRegistroExitoso() {
        String body = "{ \"email\": \"ashley@reqres.in\", \"password\":
\"abcd123\" }";

        Response response =
            given()
                .log().all()
                .contentType(ContentType.JSON)
                .body(body)
                .when()
                .post("/register")
                .then()
                .log().all()
```

```
                .statusCode(200)
                .contentType(ContentType.JSON)
                .time(lessThan(2000L))
                .body("id", notNullValue())
                .body("token", notNullValue())
                .extract().response();

        assertThat(response.jsonPath().getInt("id")).isPositive();
        assertThat(response.jsonPath().getString("token")).isNotEmpty();
    }

    // 4. POST – Registro fallido (falta password)
    @Test
    void testRegistroFallido() {
        String body = "{ \"email\": \"ashleyy@fife\" }";

        given()
            .log().all()
            .contentType(ContentType.JSON)
            .body(body)
            .when()
            .post("/register")
            .then()
            .log().all()
            .statusCode(400)
            .contentType(ContentType.JSON)
            .body("error", equalTo("Missing password"))
            .time(lessThan(2000L));
    }

    // 5. PUT – Modificación de cargo de usuario
    @Test
    void testModificarCargoUsuario() {
        String body = "{ \"name\": \"ashley\", \"job\": \"trabajo\" }";

        Response response =
            given()
                .log().all()
                .contentType(ContentType.JSON)
                .body(body)
                .when()
                .put("/users/2")
                .then()
                .log().all()
```

```
                .statusCode(200)
                .contentType(ContentType.JSON)
                .time(lessThan(2000L))
                .extract().response();

        assertThat(response.jsonPath().getString("job")).isEqualTo("trabajo"
);
        assertThat(response.jsonPath().getString("name")).isEqualTo("ashley"
);
    }

    // 6. DELETE – Eliminación de usuario
    @Test
    void testEliminarUsuario() {
        given()
            .log().all()
            .when()
            .delete("/users/2")
            .then()
            .log().all()
            .statusCode(204)
            .time(lessThan(2000L));
    }

    // 7. Validación de headers y performance
    @Test
    void testValidarHeaders() {
        given()
            .log().all()
            .when()
            .get("/users/2")
            .then()
            .log().all()
            .statusCode(200)
            .header("Content-Type", containsString("application/json"))
            .header("Server", notNullValue())
            .time(lessThan(2000L));
    }
}
```


PROBLEMAS 3
SALIDA
CONSOLA DE DEPURACION
TERMINAL
PUERTOS
PLAYWRIGHT
Expected status code <400> but was <401>.
[INFO]
[ERROR] Tests run: 8, Failures: 5, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.209 s
[INFO] Finished at: 2025-08-26T21:22:24-04:00
[INFO] ------------------------------------------------------------------------


data
support
@{id=2; email=janet.weaver@reqres.in; first_name=Janet; last_name=Weaver; avatar=https://reqres.in/img/faces/2-image.jpg} @{ur...
PS C:\Users\ashle\OneDrive\Desktop\apirestful>