



PRUEBA API CON REST- ASSURED



Devon Alvares Osorio

Pruebas de Api con Rest-Assured

Probar servicios REST es fundamental para garantizar su funcionalidad, seguridad y rendimiento. Aquí te presento algunas ventajas y mejoras que pueden surgir al probar servicios REST:

1. **Validación de funcionalidad:** Las pruebas permiten verificar que los servicios REST funcionen según lo esperado, cumpliendo con los requisitos del negocio y proporcionando las respuestas correctas a las solicitudes.
2. **Detección temprana de errores:** Las pruebas identifican errores y problemas en una etapa temprana del desarrollo, lo que facilita su corrección y evita costosos errores en producción.
3. **Mejora de la calidad del software:** Al garantizar que los servicios REST funcionen correctamente y cumplan con los estándares de calidad, se mejora la calidad general del software y se aumenta la confianza en su funcionamiento.

CASOS DE PRUEBA EJECUTADOS

CASO 1:

```
Característica: Verificar la funcionalidad de un servicio GET para
obtener un solo usuario
  Yo Como usuario
  Quiero verificar que un servicio GET devuelva un solo usuario
correctamente
  Para asegurarme de que puedo obtener la información de un usuario
específico
```

```
Escenario: Verificar obtención de un solo usuario
Dado que el servicio GET de usuarios está en línea y accesible
Cuando realiza una solicitud GET al endpoint
Entonces debería devolver un código de estado HTTP 200
Y la respuesta con información del usuario con email
```

Para la ejecución de esta prueba se hace la Assertion con el código de respuesta y el email del usuario

```
public class ListarUnUsuarioStepDefinitions {
    private Response response; 3 usages
    @Dado("que el servicio GET de usuarios está en línea y accesible")
    public void queElServicioGETDeUsuariosEstáEnLíneaYAccesible() {
        RestAssured.baseURI = REQRES_IN_SINGLE_USER;
    }
    @Cuando("realiza una solicitud GET al endpoint")
    public void realizaUnaSolicitudGETAlEndpoint() {
        response = given().log().all().get();
    }
    @Entonces("deberia devolver un código de estado HTTP {int}")
    public void deberiaDevolverUnCódigoDeEstadoHTTP(Integer int1) {
        assertEquals(response.getStatusCode(), actual: 200);
    }

    @Entonces("la respuesta con información del usuario con email")
    public void laRespuestaConInformaciónDelUsuarioConEmail() {
        Gson gson = new Gson();
        DTOUsuario dtoUsuario = gson.fromJson(response.getBody().asString(), DTOUsuario.class);
        assertEquals(dtoUsuario.getData().getEmail(), EMAIL_ESPERADO);
    }
}
```

Se observa la ejecución correcta del test y el tipo de método que le estamos mandando

The screenshot shows the output of a test runner. On the left, a list of test steps is shown with green checkmarks indicating success: 'co.com.prueba.runner.Fori 3 sec 642 ms', 'Verificar la funcionalida 3 sec 642 ms', and 'Verificar obtención c 3 sec 642 ms'. On the right, the test results summary shows 'Tests passed: 1 of 1 test - 3 sec 642 ms'. Below this, the request details are displayed: 'Request method: GET', 'Request URI: https://reqres.in/api/users/2', 'Proxy: <none>', 'Request params: <none>', and 'Query params: <none>'. The 'Request URI' is highlighted with a red box.

Luego generamos el reporte y podemos observar el escenario de escenario

The screenshot shows a test report with a 'Report' tab selected. The report is titled '@test1' and contains the following information:

- Característica:** Verificar la funcionalidad de un servicio GET para obtener un solo usuario
- Yo Como usuario** Quiero verificar que un servicio GET devuelva un solo usuario correctamente un usuario específico
- Escenario:** Verificar obtención de un solo usuario
- Steps:**
 - ✓ **Dado** que el servicio GET de usuarios está en línea y accesible
 - ✓ **Cuando** realiza una solicitud GET al endpoint
 - ✓ **Entonces** deberia devolver un código de estado HTTP 200
 - ✓ **Y** la respuesta con información del usuario con email

CASO 2:

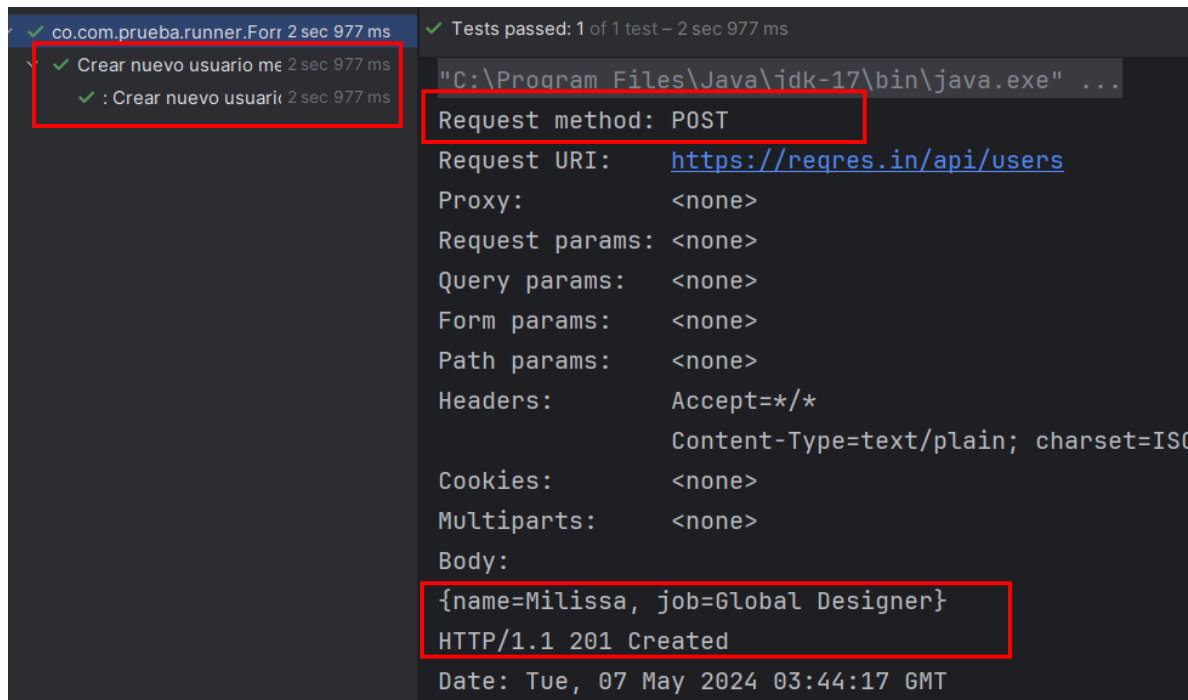
```
@test2
Característica: Crear nuevo usuario mediante un servicio POST
yo como usuario del sistema
quiero Quiero poder crear un nuevo usuario utilizando un servicio POST
Para registrar usuarios en el sistema
```

```
Escenario: Crear nuevo usuario exitosamente
Dado que el servicio POST está en línea y accesible
Cuando realiza una solicitud tipo POST
Entonces debería devolver un código 201 indicando que el usuario fue
creado correctamente
```

En el stepDefinitions se realizan los pasos para ejecutar la prueba en este caso hacemos la assertion con el código de respuesta exitoso, indicando la creacion

```
15
16 public class crearUsuarioStepDefinitions {
17     private Response response; no usages
18     @Dado("que el servicio POST está en línea y accesible")
19     public void queElServicioPOSTEstáEnLíneaYAccesible() {
20         RestAssured.baseURI = REQRES_CREATE_USER;
21     }
22     @Cuando("realiza una solicitud tipo POST")
23     public void realizaUnaSolicitudTipoPOST() {
24         crearUsuario();
25     }
26     @Entonces("debería devolver un código {int} indicando que el usuario fue creado correctamente")
27     public void deberíaDevolverUnCodigoIndicandoQueElUsuarioFueCreadoCorrectamente(Integer int1) {
28         assertEquals(EXPECTED_1, actual: 201);
29     }
30 }
31
```

En la ejecución de este test podemos observar varios aspectos como el estado, los datos que le mandamos para crear el usuario, la fecha, el método que se utiliza.

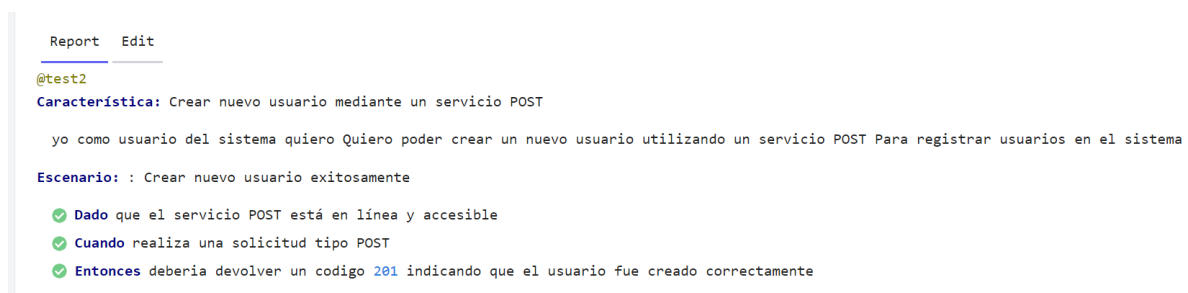


The screenshot displays a test runner interface with a dark theme. On the left, a list of test steps is shown, with the first two steps highlighted by a red box: "co.com.prueba.runner.Forr 2 sec 977 ms" and "Crear nuevo usuario me 2 sec 977 ms". The main panel on the right shows the details of the selected test step, which is a POST request to "https://regres.in/api/users". The request details are as follows:

- Request method: POST
- Request URI: <https://regres.in/api/users>
- Proxy: <none>
- Request params: <none>
- Query params: <none>
- Form params: <none>
- Path params: <none>
- Headers: Accept=*/*, Content-Type=text/plain; charset=ISO
- Cookies: <none>
- Multiparts: <none>
- Body: {name=Milissa, job=Global Designer}

The response status is "HTTP/1.1 201 Created" and the date is "Tue, 07 May 2024 03:44:17 GMT".

Observamos el reporte que nos genera para este caso de prueba



The screenshot shows a test report for the "Crear nuevo usuario" test case. The report is titled "Report" and "Edit". The test case is labeled "@test2". The report content is as follows:

Característica: Crear nuevo usuario mediante un servicio POST

yo como usuario del sistema quiero Quiero poder crear un nuevo usuario utilizando un servicio POST Para registrar usuarios en el sistema

Escenario: : Crear nuevo usuario exitosamente

- ✓ Dado que el servicio POST está en línea y accesible
- ✓ Cuando realiza una solicitud tipo POST
- ✓ Entonces debería devolver un código 201 indicando que el usuario fue creado correctamente

CASO 3:

Característica: Verificar la funcionalidad de un servicio GET de la pagina jsonplaceholder.typicode
Yo Como usuario
Quiero verificar que un servicio GET devuelva todos los usuario
Para obtener una lista con la información de todos los usuarios

Escenario: : Verificar obtención de todos los usuarios
Dado que el servicio GET en la pagina jsonplaceholder.typicode está en línea y accesible
Cuando realiza una solicitud GET al endpoint en la pagina
Entonces debería mostrar un código de estado HTTP 200

En el step de este caso de prueba se ejecutan los métodos necesarios y se realiza la assertion para comparar el código de respuesta y de esta forma sea exitosa la validación

```
public class ListarUsuariosStepDefinitions {
    private Response response; no usages
    @Dado("que el servicio GET en la pagina jsonplaceholder.typicode está en línea y accesible")
    public void que_el_servicio_get_en_la_pagina_jsonplaceholder_typicode_está_en_línea_y_accesible() {
        RestAssured.baseURI = USER_TYPE_CODE;
    }
    @Cuando("realiza una solicitud GET al endpoint en la pagina")
    public void realiza_una_solicitud_get_al_endpoint_en_la_pagina() {
        realizarSolicitud();
    }
    @Entonces("deberia mostrar un código de estado HTTP {int}")
    public void deberia_mostrar_un_código_de_estado_http(Integer int1) {
        assertEquals(EXPECTED_2, actual: 200);
    }
}
```

En este test podemos observar cómo nos responde con un código de estado 200

The screenshot displays the results of a test execution. On the left, a tree view shows the test hierarchy: 'co.com.prueba.runner.For' (2 sec 693 ms), 'Verificar la funcionalida' (2 sec 693 ms), and ': Verificar obtención' (2 sec 693 ms). The main panel shows the test output, including the command 'C:\Program Files\Java\jdk-17\bin\java.exe' and the response 'HTTP/1.1 200 OK'. The status bar at the bottom indicates 'Tests passed: 1 of 1 test - 2 sec 693 ms'.

```
co.com.prueba.runner.For: 2 sec 693 ms
Verificar la funcionalida: 2 sec 693 ms
: Verificar obtención: 2 sec 693 ms
Tests passed: 1 of 1 test - 2 sec 693 ms
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
HTTP/1.1 200 OK
Date: Tue, 07 May 2024 03:49:56 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
```

Ademas nos muestra el cuerpo del json que devuelve

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas ut"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro quibusdam"
  },

```

Con un reporte del caso de prueba

Report Edit

@test3

Característica: Verificar la funcionalidad de un servicio GET de la pagina jsonplaceholder.typicode

Yo Como usuario Quiero verificar que un servicio GET devuelva todos los usuario Para obtener una lista con la información de todos los usuarios

Escenario: : Verificar obtención de todos los usuarios

- ✓ Dado que el servicio GET en la pagina jsonplaceholder.typicode está en línea y accesible
- ✓ Cuando realiza una solicitud GET al endpoint en la pagina
- ✓ Entonces debería mostrar un código de estado HTTP 200

CASO 4

@test4

Característica: Verificar respuesta exitosa con código 200 en servicios REST

yo como usuario del sistema
quiero verificar que un servicio rest responda correctamente con un código 200
para asegurar su disponibilidad y funcionamiento adecuado

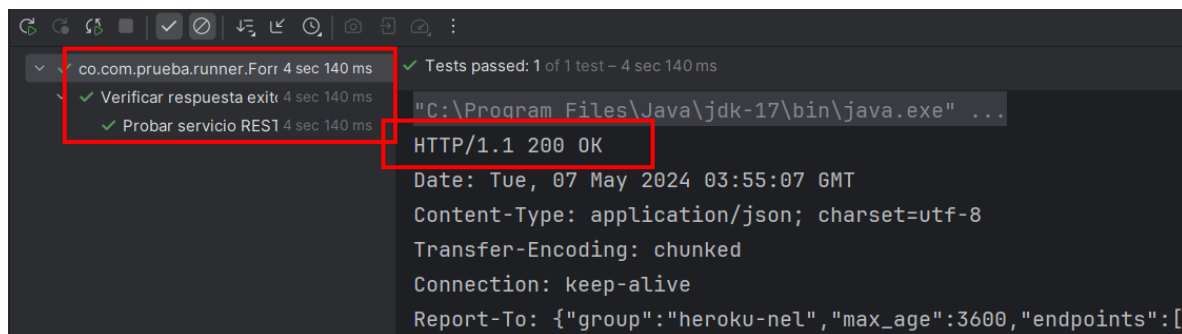
Escenario: Probar servicio REST con respuesta exitosa

- Dado que el servicio REST está en línea y accesible
- Cuando se realiza una solicitud tipo GET
- Entonces debería ver un código de estado HTTP 200

Para la ejecución de esta prueba se implementa el `setUpDefinitions`

```
1
2 public class AlbumListStrpDefinitions {
3     @Dado("que el servicio REST está en línea y accesible")
4     public void que_el_servicio_rest_está_en_línea_y_accesible() {
5         RestAssured.baseURI = ALBUM;
6     }
7     @Cuando("se realiza una solicitud tipo GET")
8     public void se_realiza_una_solicitud_tipo_get() {
9         listarAlbum();
10    }
11    @Entonces("deberia ver un código de estado HTTP {int}")
12    public void deberia_ver_un_código_de_estado_http(Integer int1) {
13        assertEquals(EXPECTED_2, actual: 200);
14    }
15 }
```

Obtenemos en la ejecución de la prueba número 4 un código de estado exitoso



The screenshot shows an IDE interface with a test runner on the left and an HTTP response on the right. The test runner shows three tests passing: "co.com.prueba.runner.Forri 4 sec 140 ms", "Verificar respuesta exito 4 sec 140 ms", and "Probar servicio REST 4 sec 140 ms". The HTTP response is "HTTP/1.1 200 OK" with headers: "Date: Tue, 07 May 2024 03:55:07 GMT", "Content-Type: application/json; charset=utf-8", "Transfer-Encoding: chunked", "Connection: keep-alive", and "Report-To: {\"group\": \"heroku-nel\", \"max_age\": 3600, \"endpoints\": [...]\"}\"".

Al igual que el cuerpo del json que genera con sus respectivos valores

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "quidem molestiae enim"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "sunt qui excepturi placeat culpa"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "omnis laborum odio"
  }
]
```