

Reto Práctico #1 – Test Plan en Azure DevOps

Nombre: Iván Camilo Martínez

Curso: Máster en QA y Automatización de Pruebas

Consultor: Deivy Arley Torres

Fecha de entrega: 27/04/2025

1. Introducción

Este informe presenta el desarrollo de un plan de pruebas de API basado en los servicios web del CRUD 'User' de Swagger Petstore (<https://petstore.swagger.io/>), utilizando Postman para la ejecución de pruebas y Azure DevOps para la gestión del trabajo. Se crearon seis casos de prueba: cuatro exitosos y dos que generaron bugs, documentando sus resultados mediante la extensión Test & Feedback.

2. Backlog y Tareas

Backlog creado: (inserta aquí captura del Backlog en Azure DevOps)

The screenshot shows the Azure DevOps Test Plan backlog board. The left sidebar has 'Test Plan' selected under 'Boards'. The main area displays a table of tasks:

Order	ID	Title	Assigned To	State
1	68	Swagger Petstore-QA	Ivan Camilo Ma...	To Do
	69	Crear Usuario	Ivan Camilo Ma...	To Do
	70	Crear Usuario (incorrecto)	Ivan Camilo Ma...	To Do
	71	Consultar Usuario Existente	Ivan Camilo Ma...	To Do
	72	Consultar Usuario (incorrecto)	Ivan Camilo Ma...	To Do
	73	Actualizar Usuario Correctamente	Ivan Camilo Ma...	To Do
	74	Eliminar Usuario del Sistema	Ivan Camilo Ma...	To Do
2	49	> OrangeHRM-QA		To Do
3	38	> SauceDemo-QA		To Do
4	15	> SEMO-QA	Ivan Camilo Ma...	To Do
5	1	> Proyecto		To Do

On the right, there are sections for 'Planning' (drag-and-drop work items for sprints) and 'Sprint 1' (Planned Effort: 0, 19 tasks, 23 bugs). A 'New Sprint' button is also visible.

Tareas asociadas:

ID	Tarea	Descripción
1	Crear Usuario	Validar que se puede crear

	Correctamente	un usuario
2	Consultar Usuario Existente	Obtener datos de un usuario creado
3	Actualizar Usuario Correctamente	Modificar datos de un usuario
4	Eliminar Usuario Existente	Borrar usuario del sistema
5	Consultar Usuario Inexistente	Validar error al consultar usuario inválido
6	Crear Usuario con Datos Vacíos	Probar error al crear usuario sin información

3. Test Plan

(inserta aquí captura del Test Plan y la carpeta “Casos de Prueba User” en Azure DevOps)

The screenshot shows the Azure DevOps Test Plan interface. On the left, there's a sidebar with options like Test Plan, Overview, Boards, Work items, Backlogs (which is selected), Sprints, Queries, Delivery Plans, Analytics views, Repos, Pipelines, Test Plans, Artifacts, and Project settings. The main area is titled 'Test Plan Team' and shows a 'Backlog' view. There are two main sections: 'Issues' and 'Planning'. The 'Issues' section lists user stories and bugs, each with a title, ID, assignee ('Ivan Camilo Ma...'), state ('To Do'), and a small icon. The 'Planning' section shows a 'Sprint 1' with a planned effort of 0, containing 23 tasks and 23 bugs. A 'New Sprint' button is visible at the bottom right of the planning area.

Order	ID	Title	Assigned To	State
1	68	Swagger Petstore-QA	Ivan Camilo Ma...	To Do
	69	Creación Usuario	Ivan Camilo Ma...	To Do
	70	Creación Usuario (incorrecto)	Ivan Camilo Ma...	To Do
	71	Consulta Usuario Existente	Ivan Camilo Ma...	To Do
	72	Consulta Usuario (incorrecto)	Ivan Camilo Ma...	To Do
	73	Actualización Usuario Correctamente	Ivan Camilo Ma...	To Do
	74	Eliminación Usuario del Sistema	Ivan Camilo Ma...	To Do
	89	Bug - Creación de Usuario sin datos (Es posible crear ...)	Ivan Camilo Ma...	To Do
2	49	> OrangeHRM-QA		To Do
3	38	> SauceDemo-QA		To Do
4	15	> DEMO-QA	Ivan Camilo Ma...	To Do
5	1	> Proyecto		To Do

4. Casos de Prueba

Caso 1

Crear Usuario Correctamente

Método: POST

URL: <https://petstore.swagger.io/v2/user>

Resultado: Pasa

The screenshot shows the Postman interface with a collection named "CRUD - SwaggerUserTest". A test case titled "POST Crear Usuario (Falla)" is selected. The request method is set to "POST" and the URL is "https://petstore.swagger.io/v2/user". The "Body" tab contains a JSON payload:

```

1 {
2   "id": 7,
3   "username": "Ivan",
4   "firstName": "Ivan",
5   "lastName": "Martinez",
6   "email": "Ivangel.com",
7   "password": "Ivan23",
8   "userStatus": 0
9 }
10

```

The response status is 200 OK, and the results show all tests passed:

- PASSED Response status code is 200
- PASSED Response time is within an acceptable range
- PASSED Response has the required fields - code, type, and message
- PASSED Code is a non-negative integer
- PASSED Type and message must be non-empty strings

Caso 2

Consultar Usuario Existente

Método: GET

URL: <https://petstore.swagger.io/v2/user/Ivan1>

Resultado: Pasa

The screenshot shows the Postman interface with the same collection and test case. The request method is now "GET" and the URL is "https://petstore.swagger.io/v2/user/Ivan". The "Body" tab shows a pre-request script:

```

pm.text("Response time is less than 200ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(200);
});

```

After the response, a post-response script runs:

```

pm.test("Response has the required fields", function () {
  const responseData = pm.response.json();
  pm.expect(responseData).to.have.property('id');
  pm.expect(responseData).to.have.property('username');
  pm.expect(responseData).to.have.property('firstName');
  pm.expect(responseData).to.have.property('lastName');
  pm.expect(responseData).to.have.property('email');
  pm.expect(responseData).to.have.property('password');
  pm.expect(responseData).to.have.property('phone');
  pm.expect(responseData).to.have.property('userStatus');
});

```

The response status is 200 OK, and the results show most tests passed except for one failure:

- PASSED Response status code is 200
- PASSED Response time is less than 200ms
- PASSED Response has the required fields
- PASSED Email is in a valid format
- FAILED Phone is in a valid format | Assertion Error: Phone should be a 10-digit number: expected 'string' to match '/^\\d{10}\\\$/'

Caso 3

Actualizar Usuario

Método: POST

URL: <https://petstore.swagger.io/v2/user/Ivan87>

Resultado: ✅ Pasa

The screenshot shows the Postman interface with a collection named "CRUD - SwaggerUserTest". A POST request is selected to update a user with ID 87. The body contains the following JSON:

```
1 {
2   "id": 87,
3   "username": "Ivan87",
4   "firstName": "Ivan",
5   "lastName": "Martínez",
6   "email": "camilogmail.com",
7   "password": "camilo123",
8   "phone": "3197487381",
9   "userStatus": 0
10 }
```

The response status is 200 OK with a response time of 149 ms. Test results show three green passes: Response status code is 200, Response time is less than 200ms, and Validate the response schema for the fields - code, type, and message.

Caso 4

Eliminar Usuario

Método: DELETE

URL: <https://petstore.swagger.io/v2/user/Ivan87>

Resultado: ✅ Pasa

The screenshot shows the Postman interface with the same collection. A DELETE request is selected to delete a user with ID 87. The body is empty.

The response status is 404 Not Found with a response time of 189 ms. Test results show one red fail: Response has the required Content-Type header | AssertionError: the given combination of arguments (undefined and string) is invalid for this assertion. You can use an array, a map, an object, a set, or a t. Other tests passed: Response status code is 404, Response time is less than 200ms, Response body is null or empty, and Validate response schema for 404.

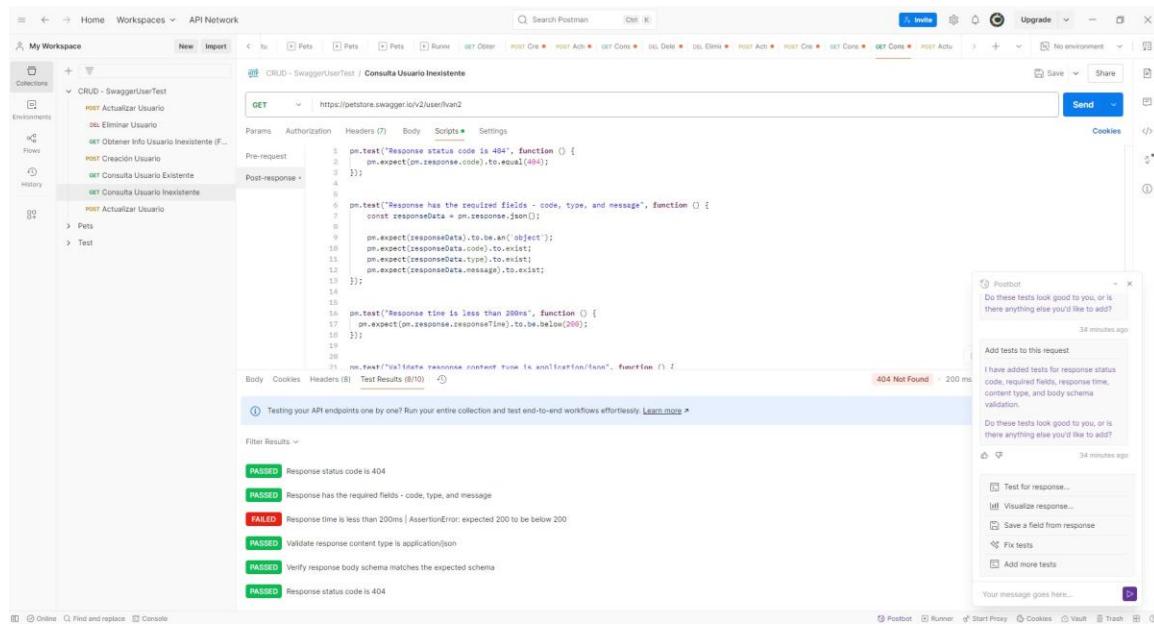
Caso 5

Consultar Usuario Inexistente

Método: GET

URL: <https://petstore.swagger.io/v2/user/Ivan2>

Resultado: ✅ Pasa



The screenshot shows the Postman interface with a collection named "CRUD - SwaggerUserTest". A test case titled "GET Consulta Usuario Inexistente" is selected. The request URL is set to <https://petstore.swagger.io/v2/user/Ivan2>. The "Post-response" script contains the following code:

```
1  pm.test("Response status is 404", function () {
2      pm.expect(pm.response.code).to.equal(404);
3  });
4
5  pm.test("Response has the serialized fixture - code, type, and message", function () {
6      const responseData = pm.response.json();
7
8      pm.expect(responseData).to.be.an("object");
9      pm.expect(responseData.code).to.exist;
10     pm.expect(responseData.type).to.exist;
11     pm.expect(responseData.message).to.exist;
12 });
13
14
15  pm.test("Response time is less than 200ms", function () {
16      pm.expect(pm.response.getTime()).to.be.below(200);
17  });
18
19
20  pm.test("Validate response content type is application/json", function () {
21      pm.expect(pm.response.contentType).to.equal("application/json");
22  });
23
24
25  pm.test("Verify response body schema matches the expected schema", function () {
26      pm.response.validate();
27  });
28
29
30  pm.test("Response status code is 404", function () {
31      pm.expect(pm.response.code).to.equal(404);
32  });
33});
```

The "Test Results" section shows 6 tests, all of which passed. The status bar at the bottom indicates "404 Not Found" and "200 ms". A tooltip at the bottom left says: "Testing your API endpoints one by one? Run your entire collection and test end-to-end workflows effortlessly. [Learn more](#)". A context menu is open on the right side of the interface.

Caso 6

Crear Usuario con Datos Vacíos

Método: POST

URL: <https://petstore.swagger.io/v2/user>

Resultado: ❌ Falla (Bug)

Bug - Creación de Usuario sin datos (Es posible crear Usuario sin datos)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'CRUD - SwaggerUserTest' and 'Petstore API'. The main area displays a collection named 'CRUD - SwaggerUserTest' with a single test case: 'POST Creación Usuario Vacío'. The request URL is 'https://petstore.swagger.io/v2/user'. The body is set to 'raw' JSON:

```
1 {  
2   "id": 0,  
3   "username": "",  
4   "firstName": "",  
5   "lastName": "",  
6   "email": "",  
7   "password": "",  
8   "phone": "",  
9   "userStatus": 0  
10 }
```

Below the request, the 'Test Results' tab is selected, showing two test cases:

- PASSED: Response status code is 200
- FAILED: Response time is within an acceptable range | AssertionError: expected 589 to be below 200
- PASSED: Response has the required fields - code, type, and message
- FAILED: Id should be a non-negative integer | AssertionError: expected undefined to be a number
- FAILED: Username, firstName, lastName, and email are non-empty strings | AssertionError: expected undefined to be a string

A modal window titled 'Postbot' is open on the right, providing options to add tests to the request.

5. Conclusiones

- Se completaron exitosamente los 5 casos de prueba solicitados.
- Los 5 casos válidos cumplieron con los resultados esperados.
- El caso erróneo permitió identificar falencias en la validación del servicio.
- Se gestionó correctamente el ciclo de pruebas desde el Backlog hasta la ejecución y reporte de bugs.