Reporte generado Pruebas de Herramientas de automatización Serenity VS Karate(Services Rest)

Responsable: Yolima Alejandra Guadir

Fecha: 29/03/2023

1. Resultasos de casos pruebas Serenity

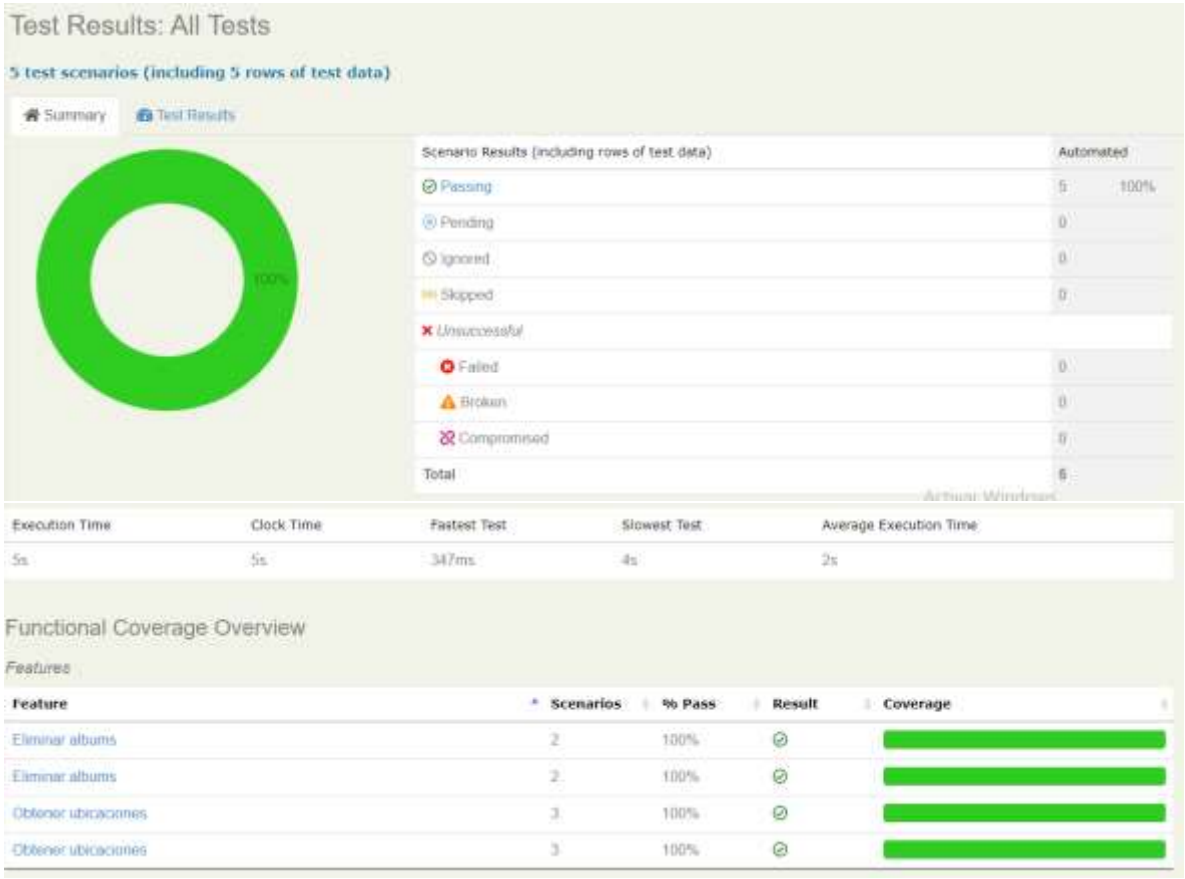


Figura 1: resultados de test de casos de prueba para servicios GET y DELETE.

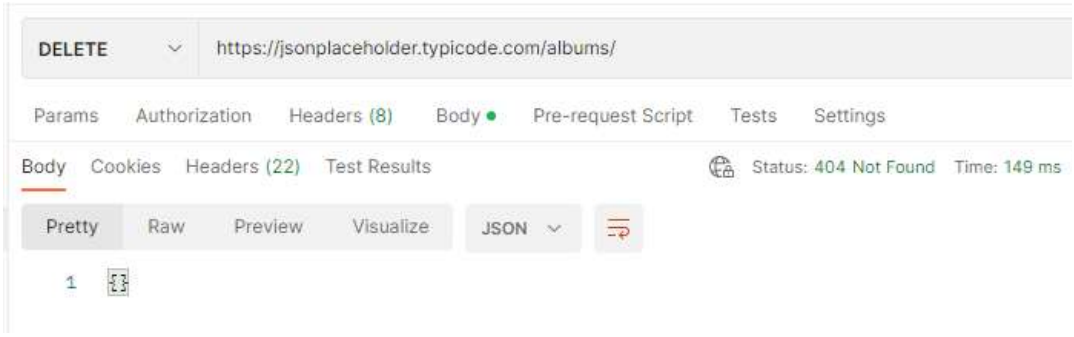
Caso de prueba para eliminar albums

```
#encoding:UTF-8
Feature: Eliminar albums
  AS  administrador de API de albums
  I WANT TO
  Eliminar albums por id
  SO THAT
  I depurar la lista de albums

  @delete
  Scenario Outline: eliminar albums
    Given el administrador esta en la pagina de albums
    When el usuario envia solicitud para eliminar albums  por id <id>
    Then la pagina retornara un estatus con codigo <codigo>

    Examples:
      | id  | codigo |
      | "1" | 200    |
      | "/" | 404    |
```

Peteción en postmat para eliminar



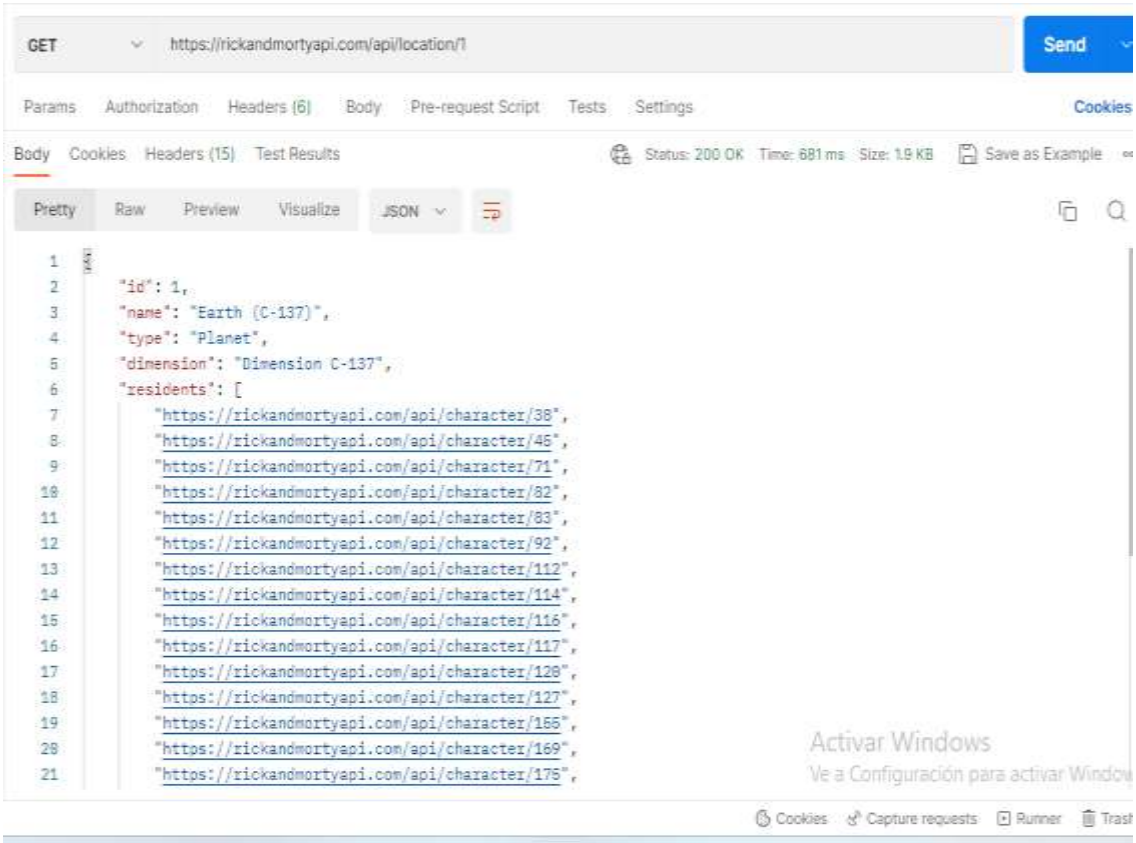
Caso de prueba para obtener ubicaciones

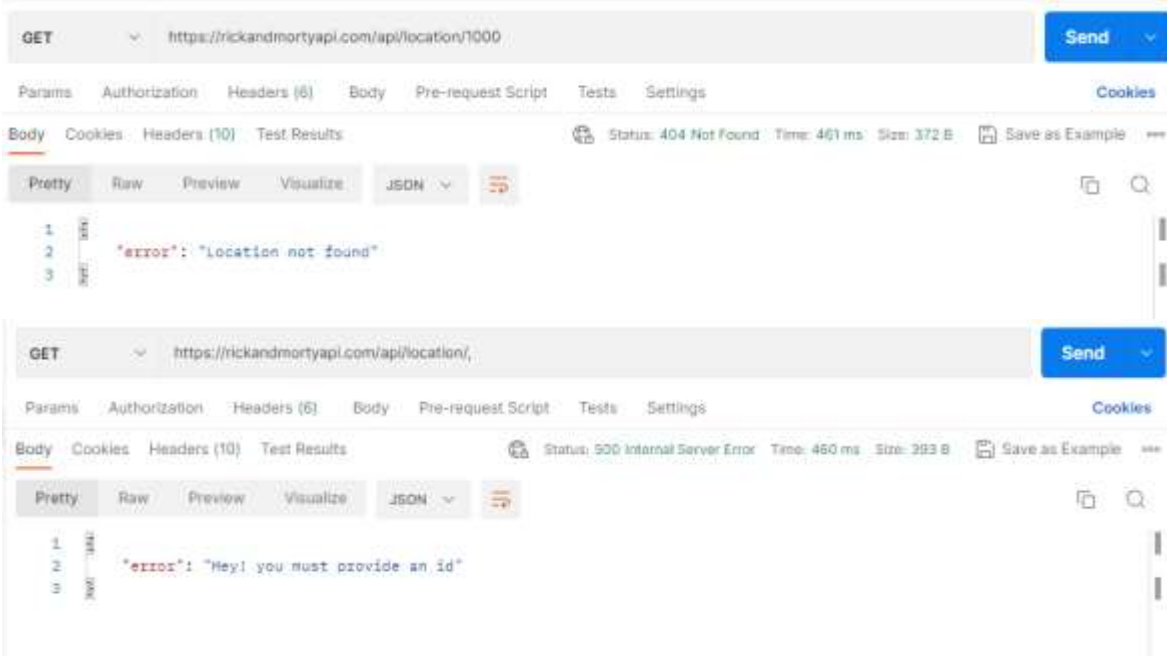
```
#encoding:UTF-8
Feature: Obtener ubicaciones
  AS  usuario de rickandmorty
  I WANT TO
  obtener ubicaciones por id
  SO THAT
  I ver datos de la ubicacion

  @Get
  Scenario Outline: buscar ubicaciones
    Given el usuario esta en la pagina
    When el usuario envia solicitud de busqueda  por id <id>
    Then la pagina retornara un estatus con codigo <codigo> nombre <name> tipo <type>

    Examples:
      | id     | codigo | name            | type     |
      | "1"    | 200    | "Earth (C-137)" | "Planet" |
      | "1000" | 404    | ""              | ""       |
      | ","    | 500    | ""              | ""       |
```

Peticiones en postman




GET
https://rickandmortyapi.com/api/location/1000
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (10)
Test Results
Status: 404 Not Found
Time: 461 ms
Size: 372 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
1
2
3
"error": "Location not found"

GET
https://rickandmortyapi.com/api/location/,
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (10)
Test Results
Status: 500 Internal Server Error
Time: 460 ms
Size: 393 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
1
2
3
"error": "Hey! you must provide an id"

2. Resultasos de casos pruebas en Karate

Tags | Timeline

Feature	Title	Passed	Failed	Scenarios	Time (ms)
src/test/java/features/comments.feature	crear comentarios en JSONPlaceholder	1	0	1	1328
src/test/java/features/user.feature	Actualizar datos de usuario en reqres	2	0	2	1123

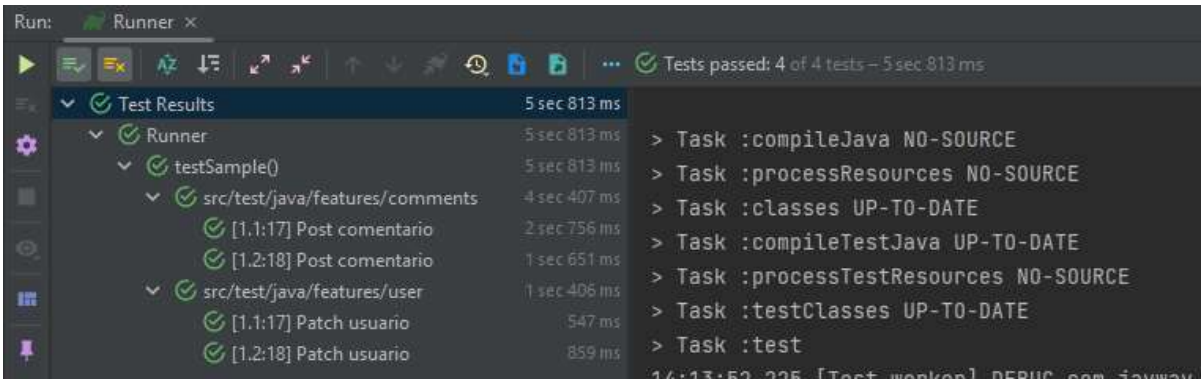
Scenario: [1.1:17] **Post comentario**	ms: 1278
9 Given url "https://jsonplaceholder.typicode.com/comments"	88
10 And request { "postId": "1", "name": "mary", "email": "john.doe123@example.com", "body":"body1"}	50
11 When method post	1094
12 Then status 201	0
13 And match response.name != null	45
14 And match response.body != null	1

Scenario: [1.2:18] **Post comentario**	ms: 314
9 Given url "https://jsonplaceholder.typicode.com/comments"	1
10 And request { "postId": "3", "name": "juan", "email": "a@gmail.com", "body":"body2"}	0
11 When method post	309
12 Then status 201	0
13 And match response.name != null	2
14 And match response.body != null	2

Scenario: [1.1:17] **Patch usuario**	ms: 538
8 Given url "https://reqres.in"+"/api/users/" + 1	1
9 And request { "name": "mary", "job": "dev" }	0
10 When method patch	516
11 Then status 200	0
12 And response.name != null	12
13 And response.job != null	1
14 And assert response.name != response.job	8

Scenario: [1.2:18] **Patch usuario**	ms: 487
8 Given url "https://reqres.in"+"/api/users/" + 3	1
9 And request { "name": "juan", "job": "dev" }	1
10 When method patch	484
11 Then status 200	0
12 And response.name != null	0
13 And response.job != null	0
14 And assert response.name != response.job	0

Figura 2 resultados de test de casos de prueba para servicios POST Y PATCH

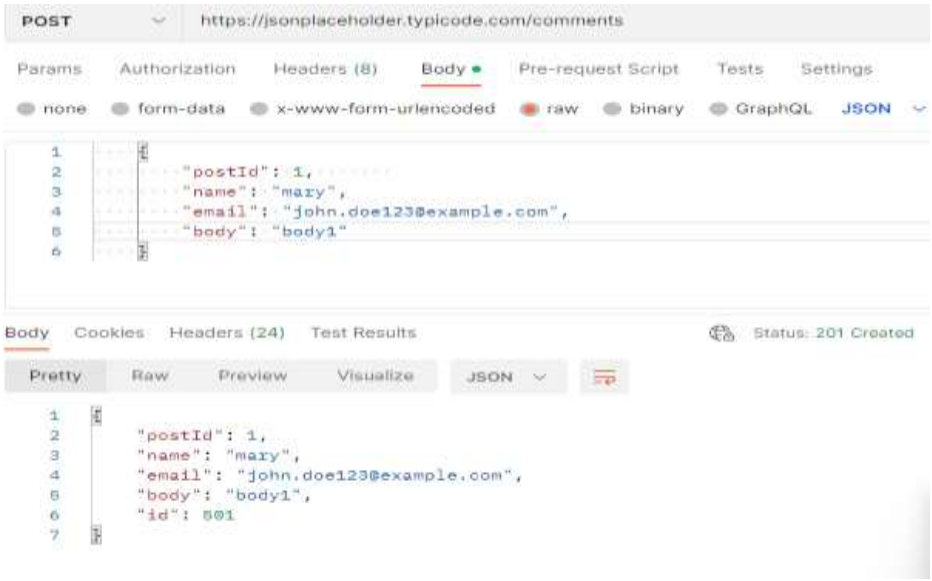


Caso de prueba para crear comentarios

```
Feature: Post comentario en JSONPlaceholder
  AS  usuario de API JSONPlaceholder
  I WANT TO
  crear comentarios
  SO THAT
  I ver comentarios

  Scenario Outline: Post comentario
    Given   url "https://jsonplaceholder.typicode.com/comments"
    And request   {  "postId": <id>,  "name": <name>, "email": <email>, "body":<body>}
    When method post
    Then status <code>
    And match response.name != null
    And match response.body != null
    Examples:
      | id  | code | name   | email                  | body    |
      | "1" | 201  | "mary" | "john.doe123@example.com" | "body1" |
      | "1" | 201  | "juan" | "ajjegmail.com"           | "body2" |
```

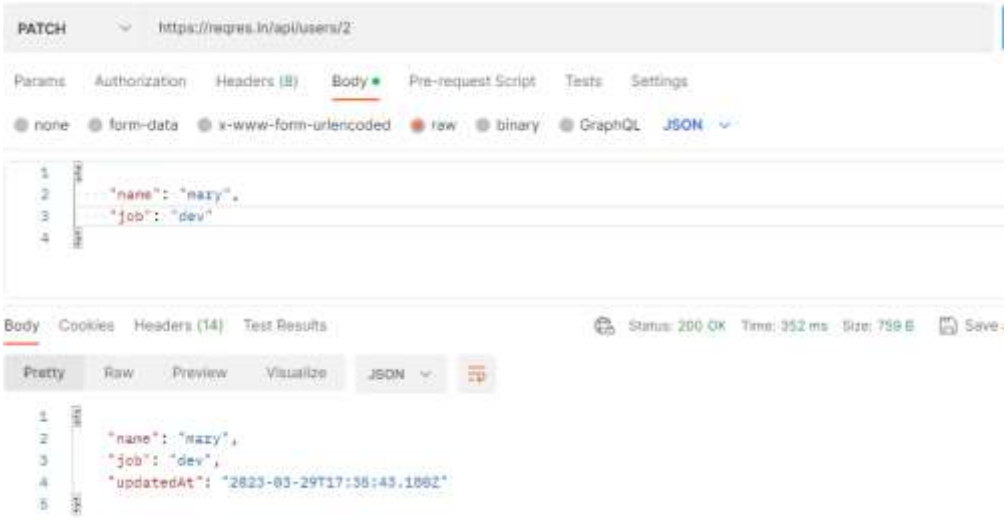
Petición en postmant



Caso de prueba para actualizar datos de usuario

```
Feature: Actualizar datos de usuario en reqres
  AS  usuario de API de reqres
  I WANT TO
  actualizar datos de usuario
  SO THAT
  I actualizar datos
  Scenario Outline: Patch usuario
    Given   url "https://reqres.in"+"/api/users/" + <id>
    And request   {  "name": <name>,  "job": <job>  }
    When method patch
    Then status <code>
    And response.name != null
    And response.job != null
    And assert response.name != response.job
    Examples:
      | id | code | name   | job   |
      | 1  | 200  | "mary" | "dev" |
      | 3  | 200  | "juan" | "dev" |
```

Petición en postmat



Conclusiones:

El escenario para obtener locations de la Api de rock and se implementó la petición GET para enviar una solicitud para obtener locations, el escenario verifica que la respuesta tenga un estado de código HTTP y que no sea nula la respuesta, si el status es 200 verifica que los campos name y type, en el caso de ser diferente retorna un mensaje con respecto al mensaje generado al realizar la petición.

El escenario para eliminar álbum, se implementó la petición DELETE para enviar una petición de eliminar álbum por Id, el escenario verifica que la respuesta tenga un estado de codigo que coincide con el código proporcionada en las tablas de ejemplos.

El escenario para actualizar crear comentarios se implementó la petición POST para enviar una solicitud para crear un comentario en la URL proporcionada, el escenario verifica que la respuesta tenga un estado de código HTTP que coincide con el código proporcionado en la tabla de ejemplos, también verifica que los campos "name" y "body" de la respuesta no sean nulos.

El escenario para actualizar datos utiliza la petición PATCH para enviar una solicitud para actualizar los datos de usuario en la URL proporcionada, la solicitud incluye un objeto JSON que contiene el nombre y el trabajo del usuario. El escenario verifica que la respuesta tenga un estado de código HTTP que coincida con el código proporcionado en la tabla de ejemplos, también verifica que los campos "name" y "job" de la respuesta no sean nulos y valida que el campo "name" sea diferente del campo "job".

Las herramientas implementadas generan informes que permiten visualizar informe de los resultados de la prueba de manera detallada, por otra parte, la implementación del framework de Kate es intuitivo ya que conserva el formato de gherkin para realizar las peticiones y validaciones.