



Pruebas realizadas en Postman

Servicios REST
Con Framework de Serenity BDD y Framework de Karate

Responsable:
Gretty María Mosquera Taborda

Versión: 1

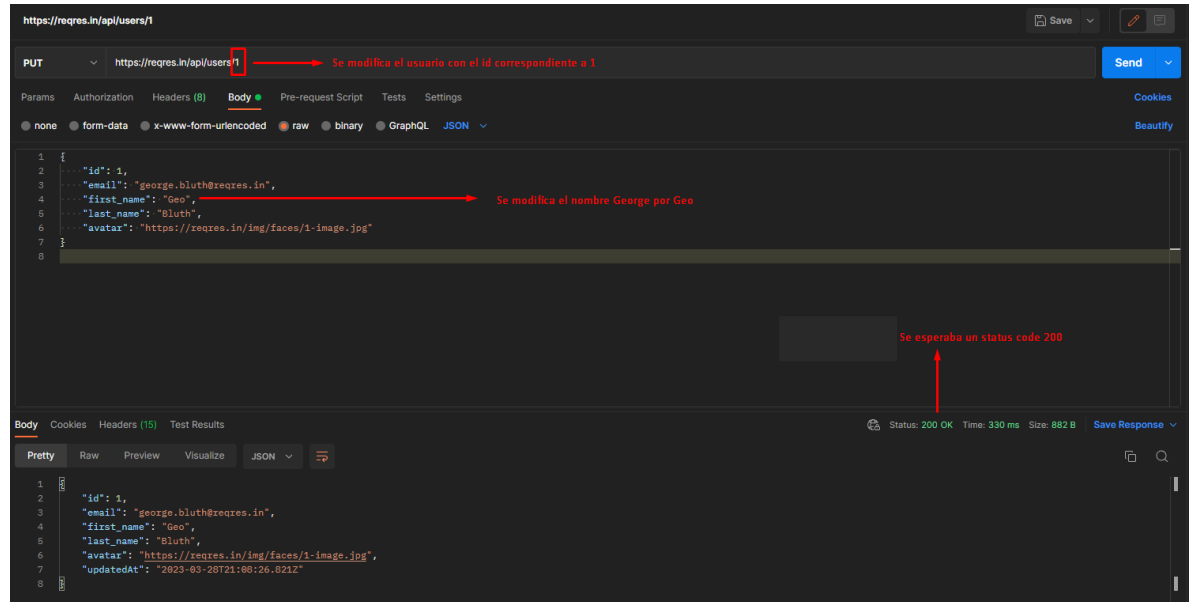
Índice

1.	Servicio PUT – Resultados en Postman	3
2.	Reporte generado con Serenity – servicio PUT	5
3.	Servicio GET – Resultados en POSTMAN	5
4.	Reporte generado con Serenity – servicio GET	6
5.	Servicio POST – Resultados en POSTMAN	6
6.	Reporte generado con Karate – Servicio POST	1
7.	Servicio DELETE – Resultados en POSTMAN	1
8.	Reporte generado con Karate – Servicio DELETE	1
9.	Conclusiones	1

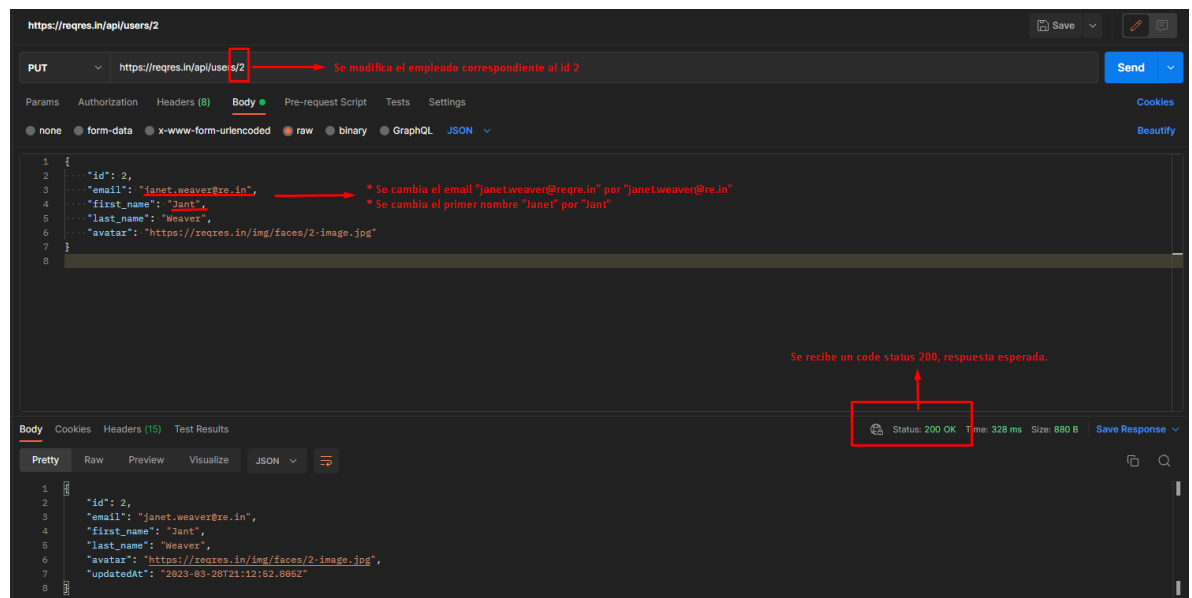
1. Servicio PUT – Resultados en Postman

Al ingresar a POSTMAN se elige el tipo de método de petición HTTP, en este caso se eligió *PUT* para modificar la información de un empleado y se ingresa la URL con el endpoint que contiene el id de dicho empleado al cual se le modificarán los datos de *email* o *first_name*. Finalmente, se presiona el botón “Send” para poder visualizar el body que contiene las propiedades del JSON modificadas.

Usuario con id 1



Usuario con id 2



Usuario con id 3

https://reqres.in/api/users/3

PUT https://reqres.in/api/users/3 Se modifica el empleado correspondiente al id 3

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 3,
3   "email": "emma.wong@gmail.com",
4   "first_name": "Emma",
5   "last_name": "Wong",
6   "avatar": "https://reqres.in/img/faces/3-image.jpg"
7 }
```

Se recibe un code status 200, respuesta esperada

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "email": "emma.wong@gmail.com",
4   "first_name": "Emma",
5   "last_name": "Wong",
6   "avatar": "https://reqres.in/img/faces/3-image.jpg",
7   "updatedAt": "2023-03-28T21:26:57.481Z"
8 }
```

Status: 200 OK Time: 328 ms Size: 877 B Save Response

Usuario con id 4

https://reqres.in/api/users/4

PUT https://reqres.in/api/users/4 Se modifica el empleado correspondiente al id 4

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 4,
3   "email": "evelin.holt@reqres.in",
4   "first_name": "Evelin",
5   "last_name": "Holt",
6   "avatar": "https://reqres.in/img/faces/4-image.jpg"
7 }
```

Tanto en el email como en el primer nombre se cambia "Eve" por "Evelin"

Se recibe un code status 200, respuesta esperada.

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON

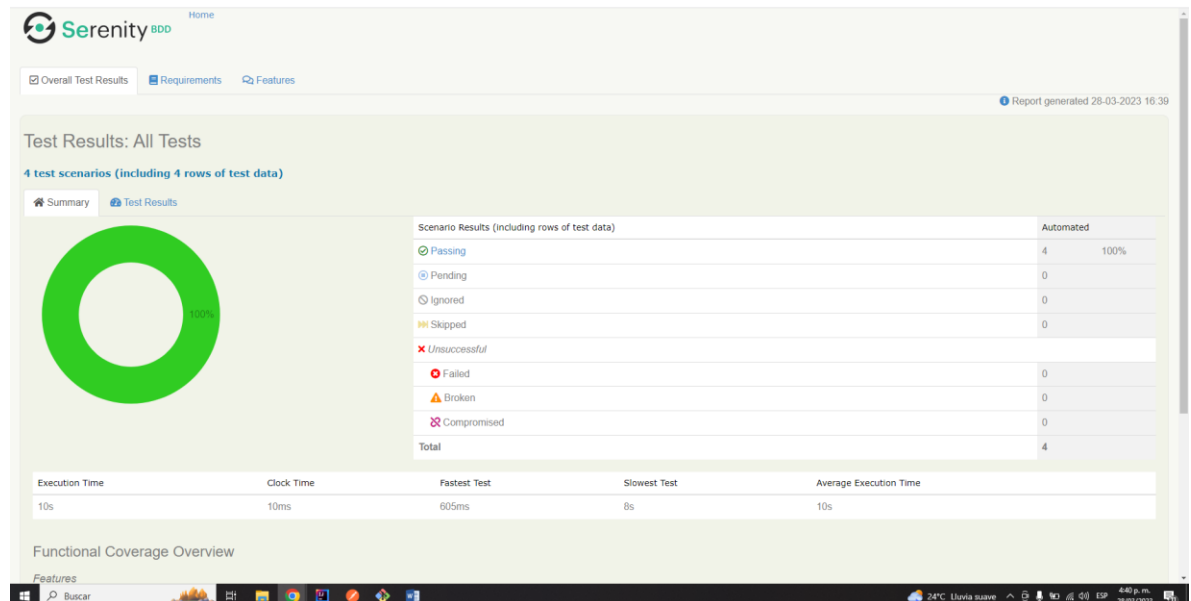
```
1 {
2   "id": 4,
3   "email": "evelin.holt@reqres.in",
4   "first_name": "Evelin",
5   "last_name": "Holt",
6   "avatar": "https://reqres.in/img/faces/4-image.jpg",
7   "updatedAt": "2023-03-28T21:31:48.727Z"
8 }
```

Status: 200 OK Time: 318 ms Size: 881 B Save Response

2. Reporte generado con Serenity – servicio PUT

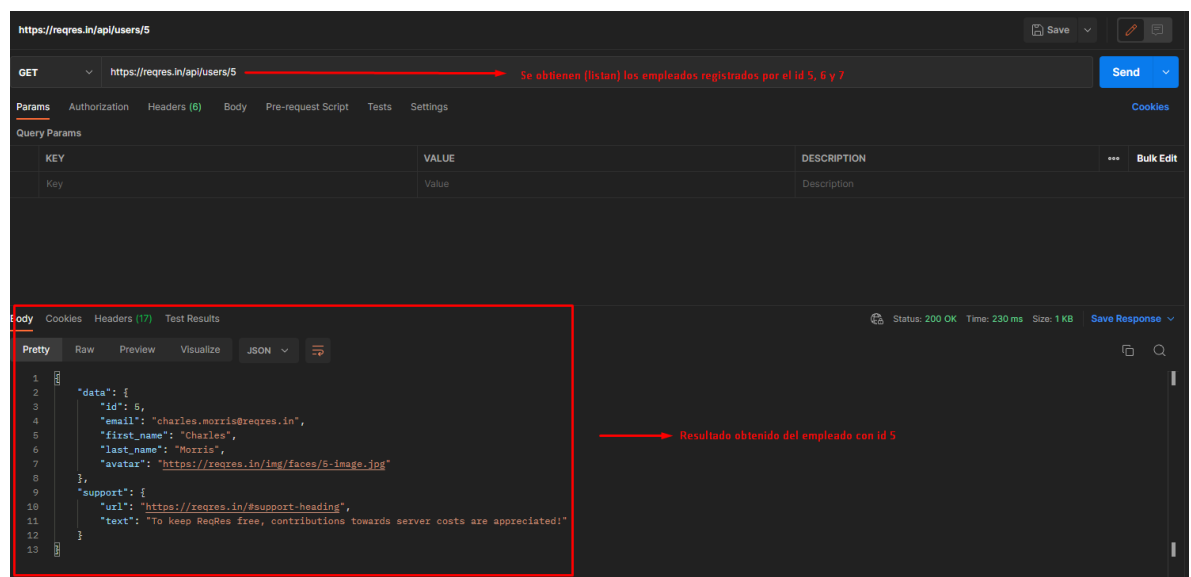
Estando en el IDE IntelliJ en la carpeta *Target* se busca un archivo *index.html* el cual se abre con el navegador de preferencia y de esta manera se obtiene el siguiente reporte.

(En el reporte se evidencia los 4 escenarios que cumplieron).



3. Servicio GET – Resultados en POSTMAN

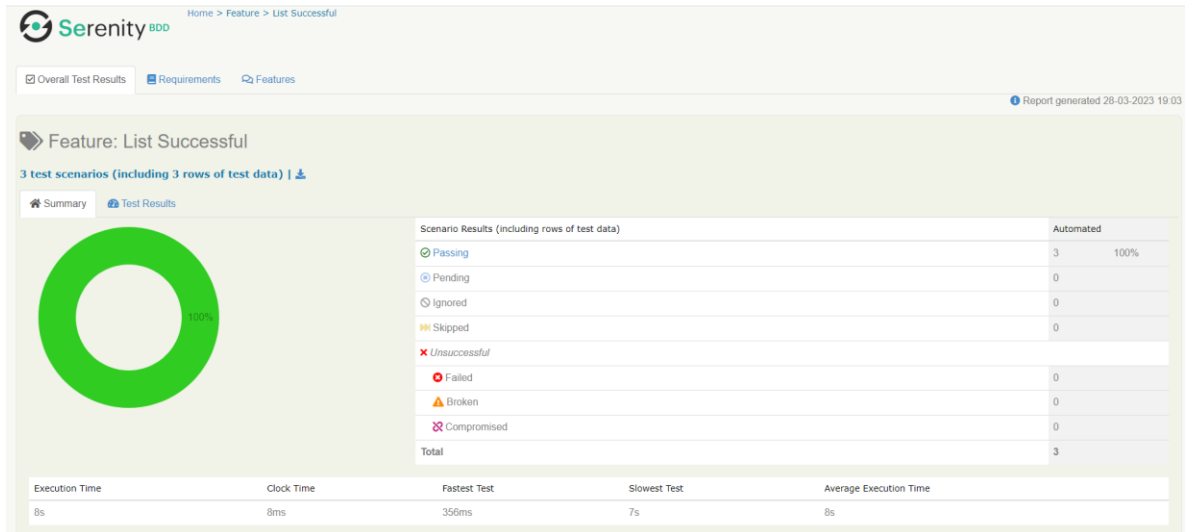
Al ingresar a POSTMAN se elige el tipo de método de petición HTTP, en este caso se eligió *GET* para listar o mostrar la información de un empleado en específico y se ingresa la URL con el endpoint que contiene el id de dicho empleado. Finalmente, se presiona el botón “Send” para poder visualizar el body que contiene las propiedades del JSON que se consultaron.



4. Reporte generado con Serenity – servicio GET

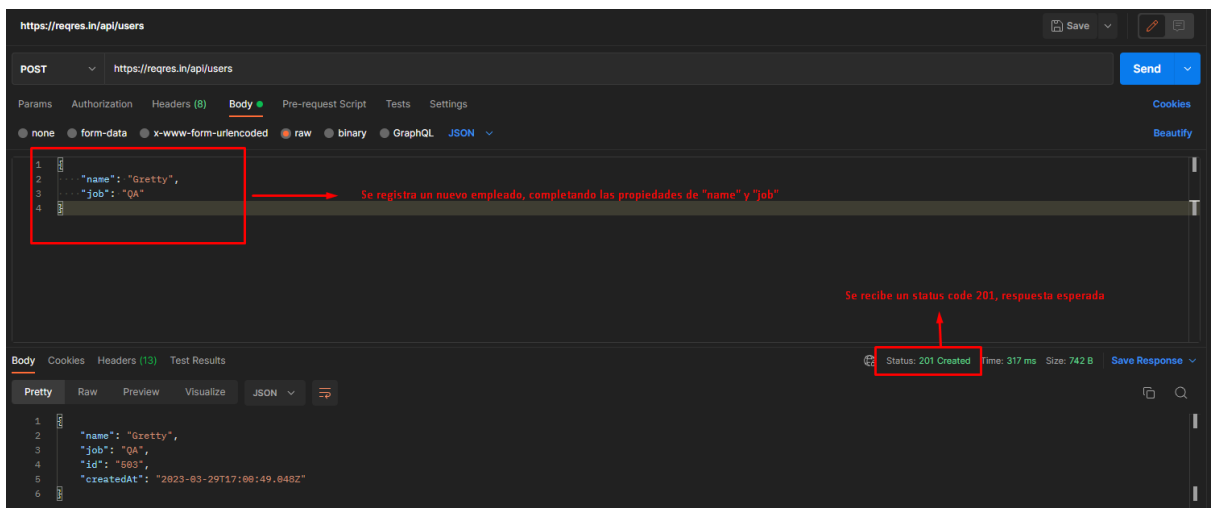
Estando en el IDE IntelliJ en la carpeta *Target* se busca un archivo *index.html* el cual se abre con el navegador de preferencia y de esta manera se obtiene el siguiente reporte.

(En el reporte se evidencia los 3 escenarios que cumplieron).



5. Servicio POST – Resultados en POSTMAN

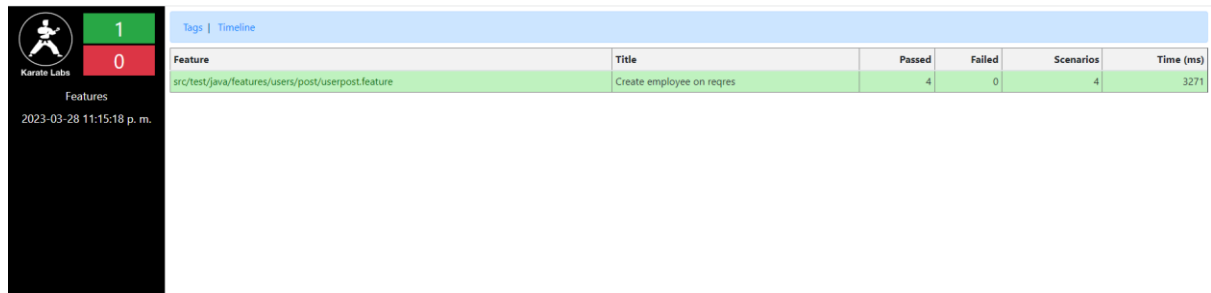
Al ingresar a POSTMAN se elige el tipo de método de petición HTTP, en este caso se eligió *POST* para agregar o registrar un empleado y se ingresa la URL con el endpoint. Finalmente, se presiona el botón “Send” para poder visualizar el body que contiene las propiedades del JSON que se añadieron.



6. Reporte generado con Karate – Servicio POST

Estando en el IDE IntelliJ en la carpeta *Target*, posterior a la carpeta *karate-reports* se busca un archivo *karate-summary.html* el cual se abre con el navegador de preferencia y de esta manera se obtiene el siguiente reporte.

(En el reporte se evidencia los 4 escenarios que cumplieron).

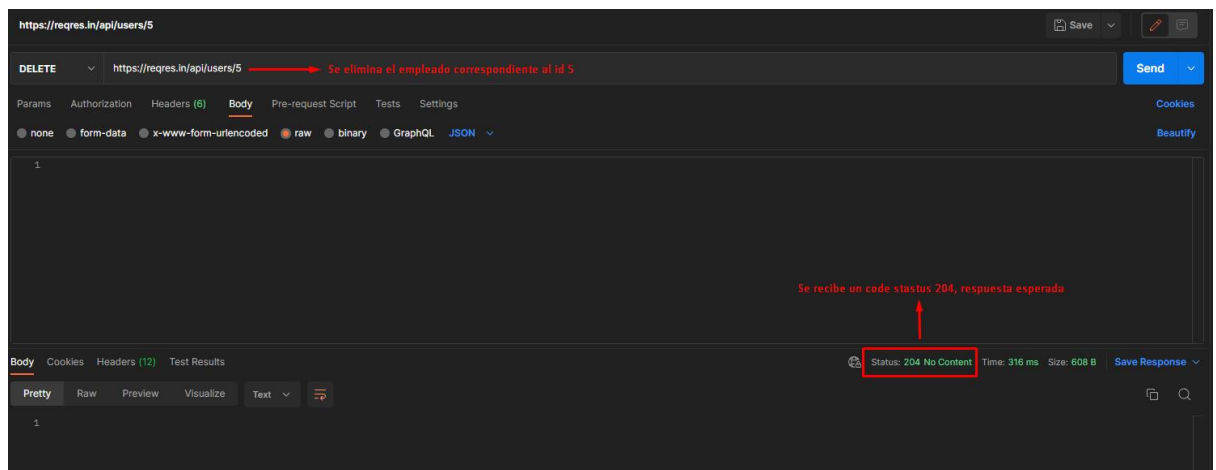


The image shows a Karate summary report. On the left, there is a sidebar with the Karate Labs logo, a green box with the number '1', a red box with the number '0', and the text 'Features' and '2023-03-28 11:15:18 p. m.'. The main area has a table with columns: Feature, Title, Passed, Failed, Scenarios, and Time (ms). The table contains one row with the following data:

Feature	Title	Passed	Failed	Scenarios	Time (ms)
src/test/java/features/users/post/userpost.feature	Create employee on regres	4	0	4	3271

7. Servicio DELETE – Resultados en POSTMAN


Al ingresar a POSTMAN se elige el tipo de método de petición HTTP, en este caso se eligió *DELETE* para eliminar un empleado en específico y se ingresa la URL con el edpoint que contiene el id de dicho empleado. Finalmente, se presiona el botón “Send”.



8. Reporte generado con Karate – Servicio DELETE

Estando en el IDE IntelliJ en la carpeta *Target*, posterior a la carpeta *karate-reports* se busca un archivo *karate-summary.html* el cual se abre con el navegador de preferencia y de esta manera se obtiene el siguiente reporte.

(En el reporte se evidencia los 4 escenarios que cumplieron).



1

0

Features

2023-03-28 11:18:55 p. m.

Tags Timeline					
Feature	Title	Passed	Failed	Scenarios	Time (ms)
src/test/java/features/users/delete/userdelete.feature	Delete employee on reqres	3	0	3	2578

9. Conclusiones

Karate, Cucumber y Rest Assured son herramientas de automatización de pruebas populares para realizar pruebas de API y aunque las tres herramientas comparten características similares, Karate ofrece algunas ventajas únicas, por ejemplo: Karate nos permite realizar pruebas completas a través de la escritura de un solo escenario gracias a su propio lenguaje de dominio específico, también facilita la ejecución de múltiples escenarios en paralelo y su uso resulta ser más sencillo en comparación a Cucumber y Rest Assured, los cuales requieren la implementación de pasos y lógica de negocio, sin embargo, la elección entre estas herramientas dependerá del proyecto y las necesidades que se tengan a nivel del equipo.