

# PRUEBA QA AUTOMATION

QA: Gherard Chipana Quiñones

## Ejercicio 1 (QAA-01) API Temperatura

### Descripción:

Crear los casos de prueba posibles para testear el siguiente microservicio GET, que recibe:

En el request la fecha del día actual en formato DD-MM-AAAA. Dos headers, el primero es "Country", cuyo dominio de valores puede ser "Chile" o "Argentina" y el segundo es "City", cuyo dominio de valores es "Santiago", "Arica", "Chiloe" (cuando se trata de Country Chile) y "BuenosAires", "SanJuan" (cuando country es Argentina) Cuando el resultado es positivo entrega status ok y código http 200 y en caso de error el status es "client error" y en caso de error de conexión el status es "server error" y en el caso de código para cada uno de estos status se entrega los códigos http de error estándar. Considera que cuando es un País distinto a los señalados o una ciudad distinta el response "País y Ciudad incorrectos", y que todos los campos a ingresar son Sensitive Case.

El response del microservicio devuelve la temperaturaActual y temperaturaDiaSiguiente en formato json.

Feature: Obtener la Temperatura Actual y la Temperatura del Día Siguiente.

1. Scenario Outline: Get API Temperatura de Chile  
Given se configura el endpoint del microservicio GET  
When se coloca como request la fecha del día actual en formato "DD-MM-AAAA"  
And se coloca el header Country con valor <Country>  
And se coloca el header City con valor <City>  
Then se recibe un status OK  
And se recibe un código Http 200  
And se muestra la { TemperaturaActual, TemperaturaDiaSiguiente }

Examples:

Country	City
Chile	Santiago
Chile	Arica
Chile	Chiloe

2. Scenario Outline: Get API Temperatura de Argentina  
Given se configura el endpoint del microservicio GET  
When se coloca como request la fecha del día actual en formato "DD-MM-AAAA"  
And se coloca el header Country con valor <Country>  
And se coloca el header City con valor <City>  
Then se recibe un status OK  
And se recibe un código Http 200  
And se muestra la { TemperaturaActual, TemperaturaDiaSiguiente }

Examples:

Country	City
Argentina	BuenosAires
Argentina	SanJuan

3. Scenario Outline: GET País y Ciudad Incorrectos
- Given se configura el endpoint del microservicio GET
  - When se coloca como request la fecha del día actual en formato "DD-MM-AAAA"
  - And se coloca el header Country con valor <Country>
  - And se coloca el header City con valor <City>
  - Then se recibe un status client error
  - And se recibe un código Http 400
  - And se muestra { País y Ciudad incorrectos }

Examples:

| Country | City |  
| Other | Other |

4. Scenario Outline: Get Client Error
- Given se configura el endpoint del microservicio GET
  - When se coloca como request la fecha del día actual en formato "DD-MM-AAAA"
  - And se coloca el header Country con valor <Country>
  - And se coloca el header City con valor <City>
  - Then se recibe un status client error
  - And se recibe un código Http 400

Examples:

Country	City
Chile	BuenosAires
Chile	SanJuan
Argentina	Santiago
Argentina	Arica
Argentina	Chiloé

5. Scenario Outline: Get Server Error
- Given se sobrecarga el microservicio
  - And se configura el endpoint del microservicio GET
  - When se coloca como request la fecha del día actual en formato "DD-MM-AAAA"
  - And se coloca el header Country con valor <Country>
  - And se coloca el header City con valor <City>
  - Then se recibe un status server error
  - And se recibe un código Http 500

Examples:

Country	City
Chile	BuenosAires
Chile	SanJuan
Argentina	Santiago
Argentina	Arica
Argentina	Chiloé

## Ejercicio 2 (QAA-02) Automatización de API

### Descripción:

Tenemos la siguiente API <https://jsonplaceholder.typicode.com/>

Y esta API tiene los siguientes EndPoints de consulta:

posts 100 posts

comments 500 comments

albums 100 albums

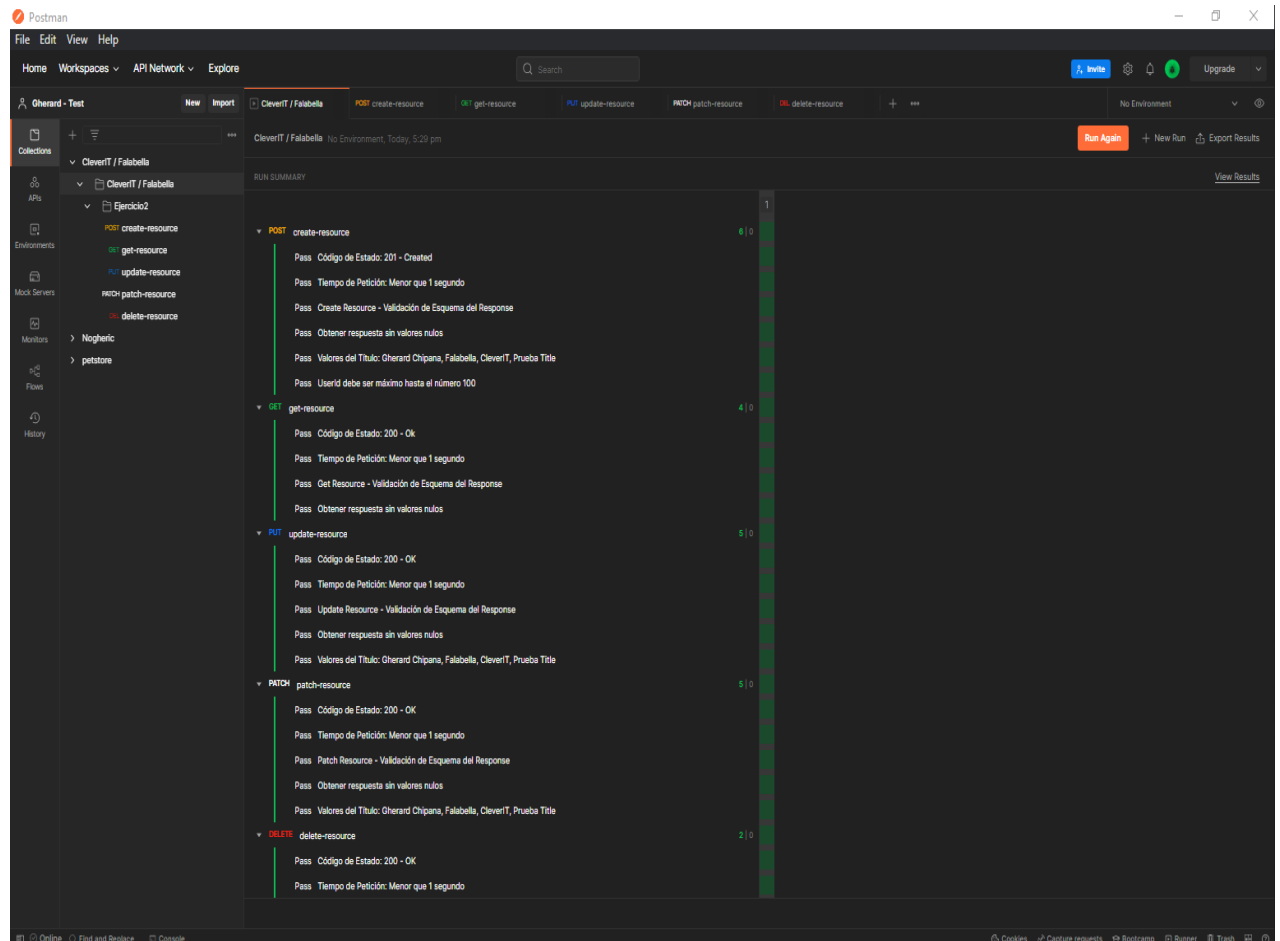
photos 5000 photos

todos 200 todos

users 10 users

Considerando esta API y sus recursos, selecciona 3 de estos recursos y diseñar los casos de prueba funcionales que veas adecuados y luego automatiza al menos 6 casos de prueba distintos (debe ser en Postman)

### POSTMAN:



NEWMAN:

EXPLORER

newman

test.json

CleverIT / Falabella / Ejercicio2

create-resource

get-resource

update-resource

patch-resource

delete-resource

POST https://jsonplaceholder.typicode.com/posts [201 Created, 1.25kB, 551ms]

Código de Estado: 201 - Created

Tiempo de Petición: Menor que 1 segundo

Create Resource - Validación de Esquema del Response

Obtener respuesta sin valores nulos

Valores del Título: Gherard Chipana, Falabella, CleverIT, Prueba Title

UserId debe ser máximo hasta el número 100

GET https://jsonplaceholder.typicode.com/posts [200 OK, 28.61kB, 42ms]

Código de Estado: 200 - OK

Tiempo de Petición: Menor que 1 segundo

Get Resource - Validación de Esquema del Response

Obtener respuesta sin valores nulos

PUT https://jsonplaceholder.typicode.com/posts/1 [200 OK, 1.14kB, 216ms]

Código de Estado: 200 - OK

Tiempo de Petición: Menor que 1 segundo

Update Resource - Validación de Esquema del Response

Obtener respuesta sin valores nulos

Valores del Título: Gherard Chipana, Falabella, CleverIT, Prueba Title

PATCH https://jsonplaceholder.typicode.com/posts/1 [200 OK, 1.31kB, 214ms]

Código de Estado: 200 - OK

Tiempo de Petición: Menor que 1 segundo

Patch Resource - Validación de Esquema del Response

Obtener respuesta sin valores nulos

Valores del Título: Gherard Chipana, Falabella, CleverIT, Prueba Title

DELETE https://jsonplaceholder.typicode.com/posts/1 [200 OK, 1.05kB, 215ms]

Código de Estado: 200 - OK

Tiempo de Petición: Menor que 1 segundo

	executed	failed
iterations	1	0
requests	5	0
test-scripts	5	0
prerequisite-scripts	5	0
assertions	22	0
total run duration: 1918ms		
total data received: 27.9kB (approx)		

NEWMAN REPORT:

LightDark

Summary

Total Requests5

Failed Tests0

Skipped Tests0

1 ITERATION AVAILABLE TO VIEW

Expand FoldersExpand Requests

1

ITERATION 1 SELECTED

CLEVERIT / FALABELLA / EJERCICIO2 - 5 REQUESTS IN THE FOLDER

Iteration: 1 - create-resource

Iteration: 1 - get-resource

Iteration: 1 - update-resource

Iteration: 1 - patch-resource

Iteration: 1 - delete-resource

### Ejercicio 3 (QAA-03) Explicación de técnicas

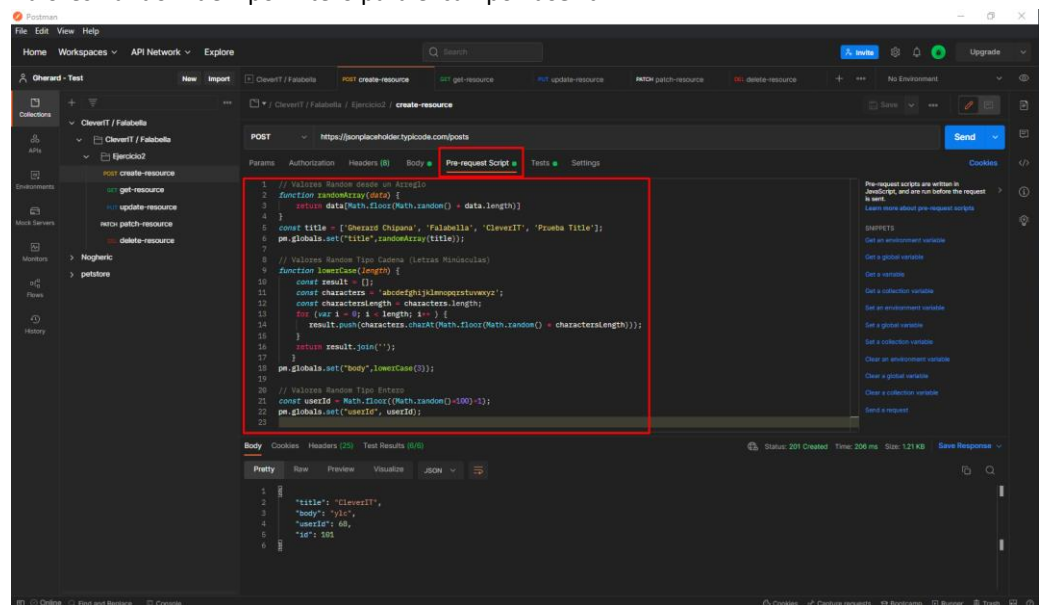
Para ambos ejercicios 1 y 2, explicar las técnicas que usaste para generar las pruebas y donde fueron aplicadas.

Además de cuáles son los puntos a tener en cuenta cuando probamos una API

En el EJERCICIO 1 se utilizó el lenguaje **Gherkin**, el cual es utilizado hoy en día para poder optimizar la comunicación entre el perfil de negocio y el perfil técnico, este lenguaje es utilizado sobre todo cuando se realiza automatización de pruebas, para que todos los involucrados desde negocio hasta el usuario final puedan comprender el flujo que se viene realizando, así mismo, permite crear escenarios de pruebas, que vendrían a ser un conjunto de casos de prueba, pero optimizados, los cuales, son pasados a código cuando se deben automatizar.

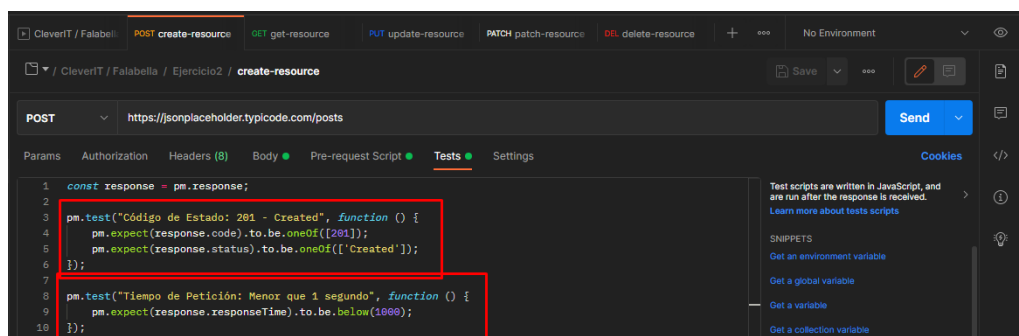
En el EJERCICIO 2 se usó la herramienta **Postman, Newman y JavaScript**, en el cual, se puede generar a través de código datos dinámicos para las pruebas y también realizar test automatizados, a continuación, se mostrará los 6 casos de prueba realizados en el api **create-resource**:

1. **Primero creamos data dinámica para poder hacer nuestras pruebas con JavaScript.**
  - a. Valores Random de un arreglo para el campo **"title"**
  - b. Valores Random de Tipo Cadena (Solo letras minúsculas) para el campo **"Body"**
  - c. Valores Random de Tipo Entero para el campo **"userId"**

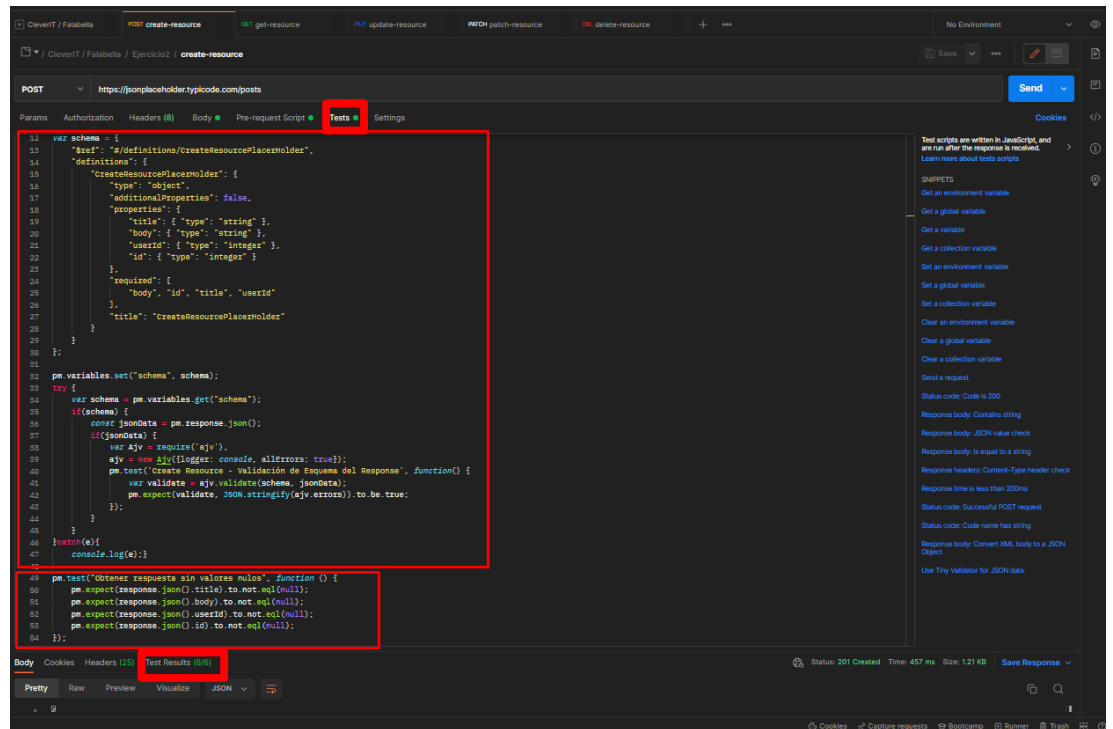


2. **Ahora realizamos nuestros test-case en la pestaña "Tests"**

- a. Code – Status: 201 – Created
- b. Tiempo de Petición: Menor que 1 Segundo.



- c. Validar que el esquema del response sea como el resultado esperado.
- d. Obtener response sin valores null.



- e. Que el campo “title” solo tenga los valores “Gherard Chipana, Falabella, CleverIT, Prueba Title”
- f. Que el campo “userId” tenga un valor máximo de 100

