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 conexion el status es "server error" y en el caso de codigo para cada uno de estos status se entrega los codigos 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	Chiloe

5. Scenario Outline: Get Client 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	Chiloe

Ejercicio 2 (QAA-02) Automatización de API

Descripción:

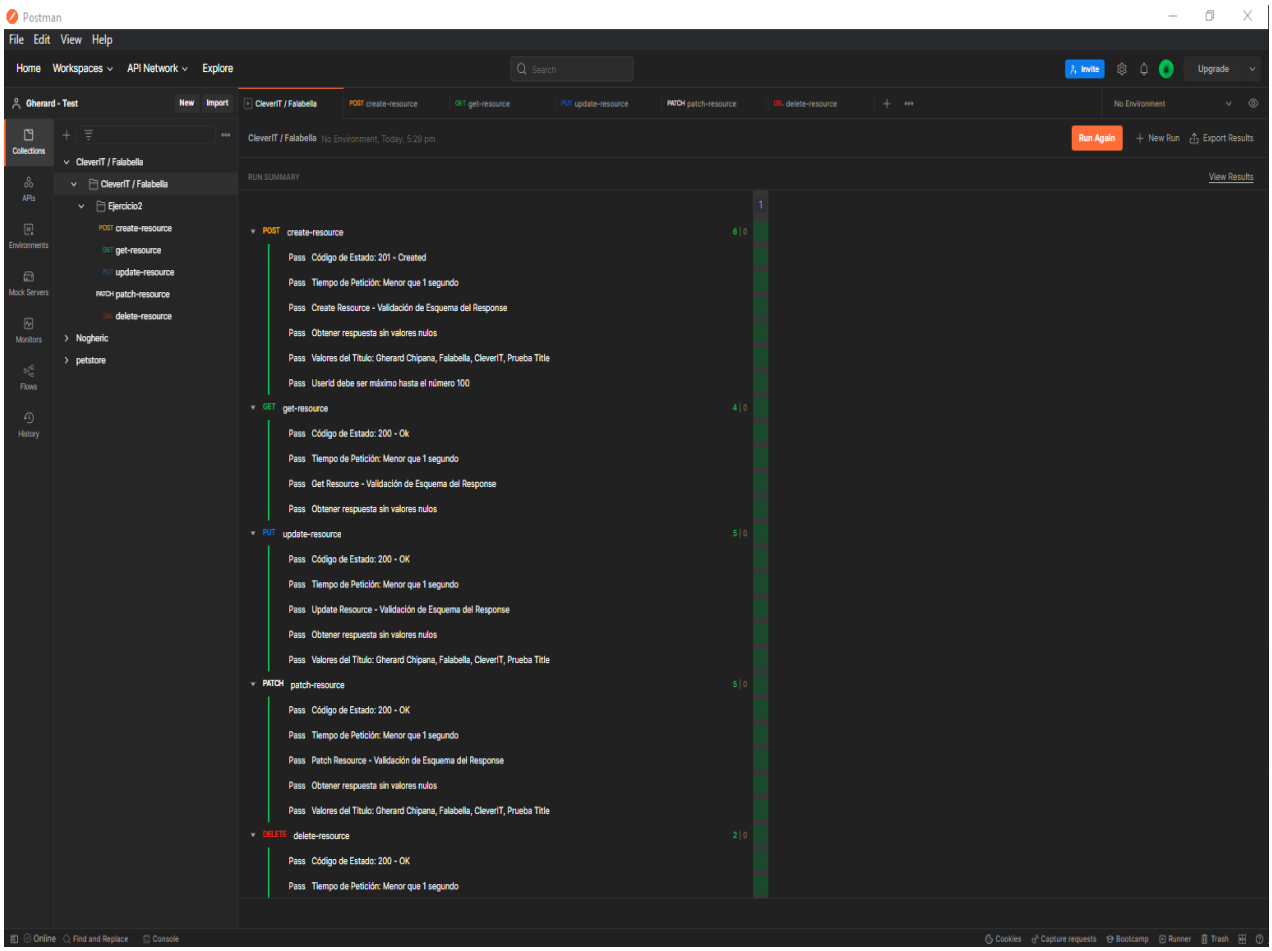
Tenemos la siguiente API https://jsonplaceholder.typicode.com/

Y esta API tiene los siguientes EndPoints de consulta:

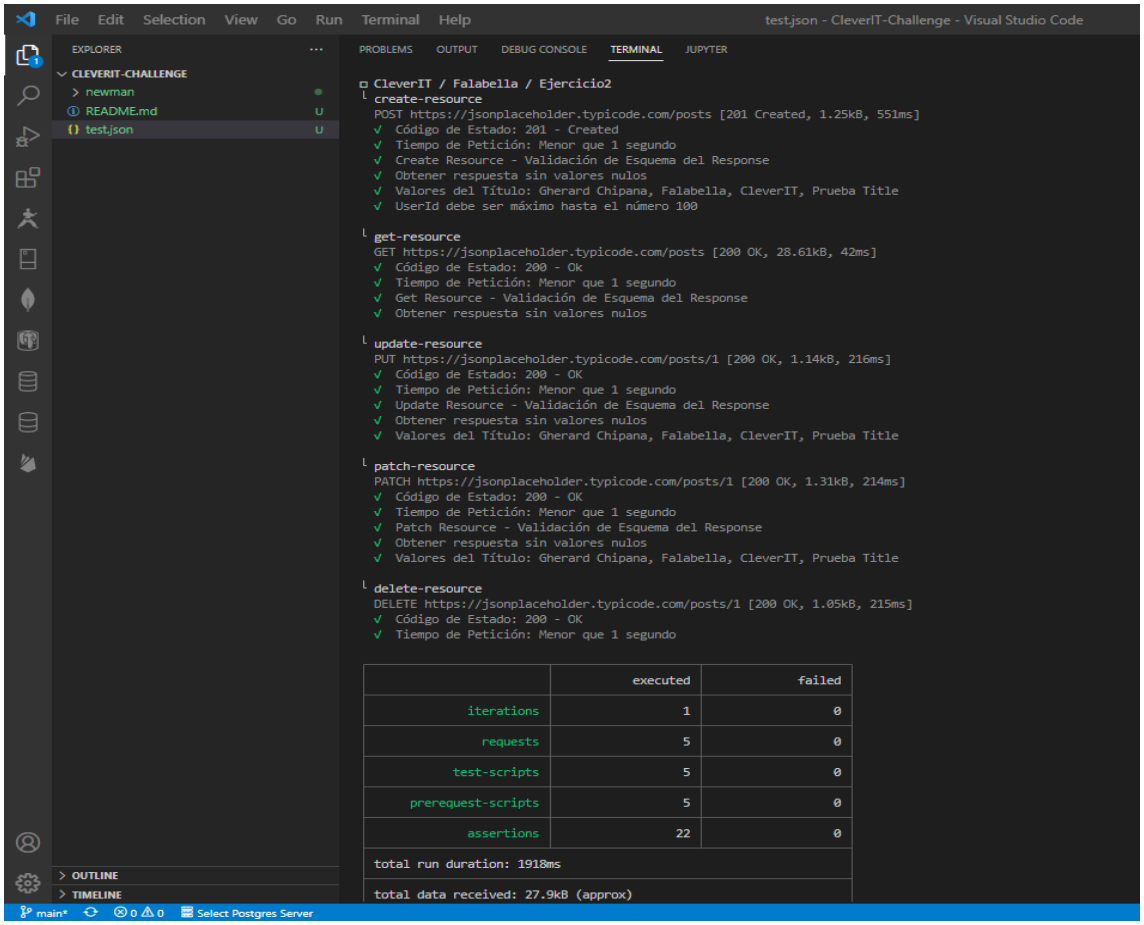
posts 100 posts
comments 500 comments
albums 100 albums
photos 5000 photos
todos 200 todos
users 10 users

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)

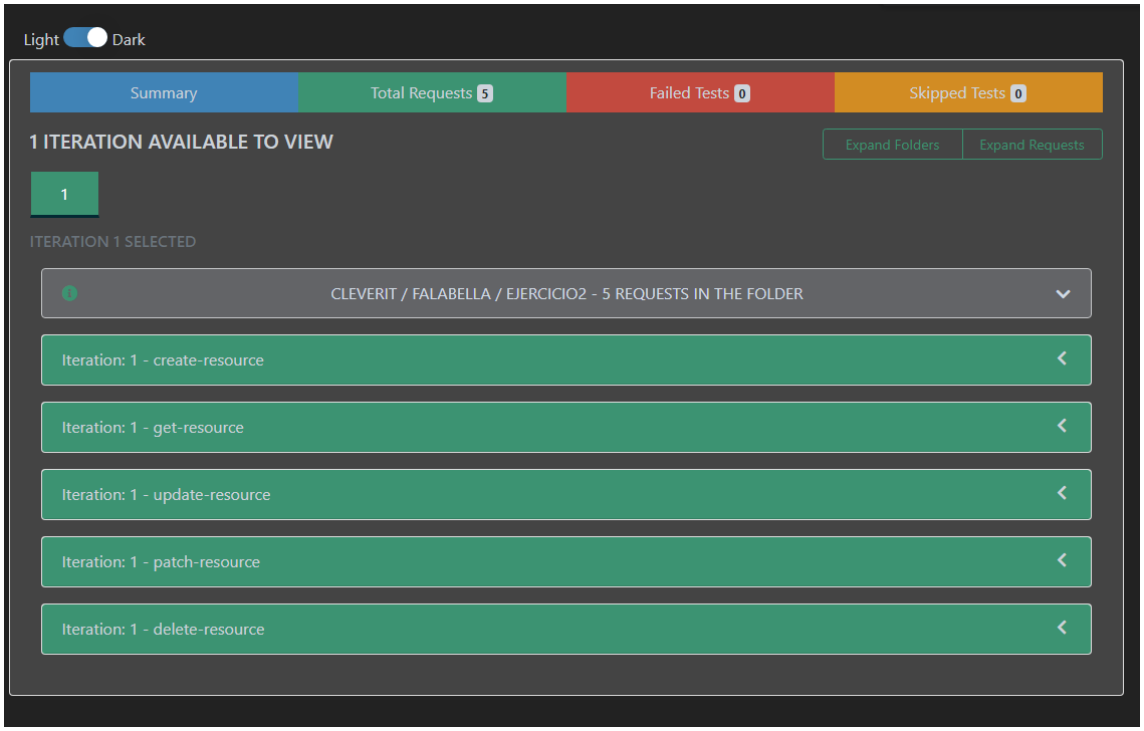
POSTMAN:



NEWMAN:



NEWMAN REPORT:



Ejercicio 3 (QAA-03) Explicación de técnicas

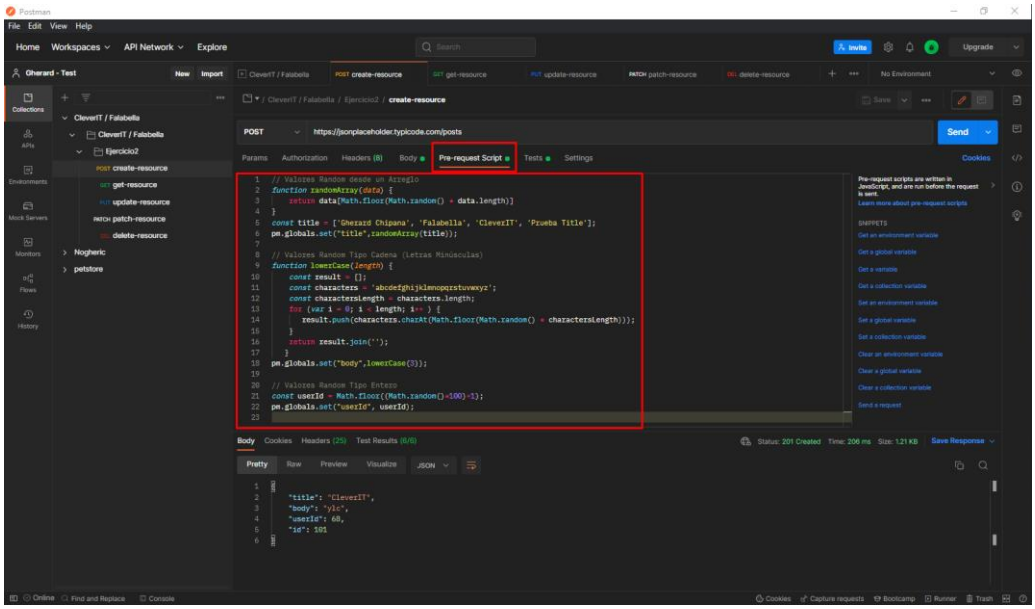
Para ambos ejercicios 1 y 2, explicar las técnicas que usaste para generar las pruebas y donde fueron aplicadas.

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

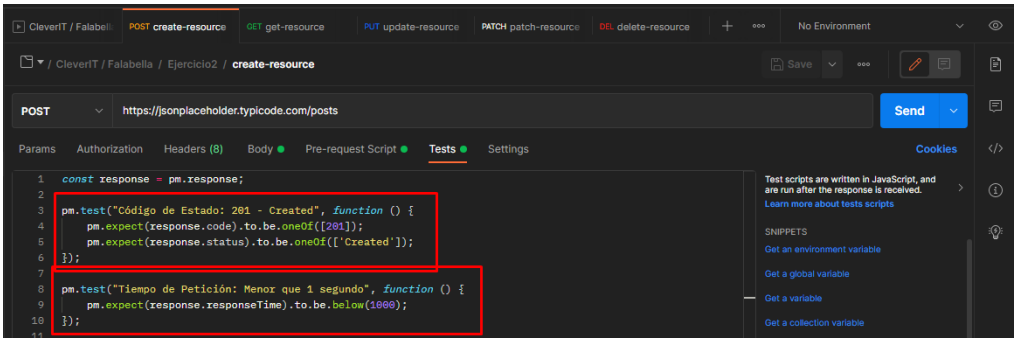
En el EJERCICIO 1 se utilizó el lenguaje **Gherkin**, el cual es utilizado hoy en día para poder optimizar la comunicación entre el perfil de negocio y el perfil técnico, este lenguaje es utilizado sobre todo cuando se realiza automatización de pruebas, para que todos los involucrados desde negocio hasta 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"**

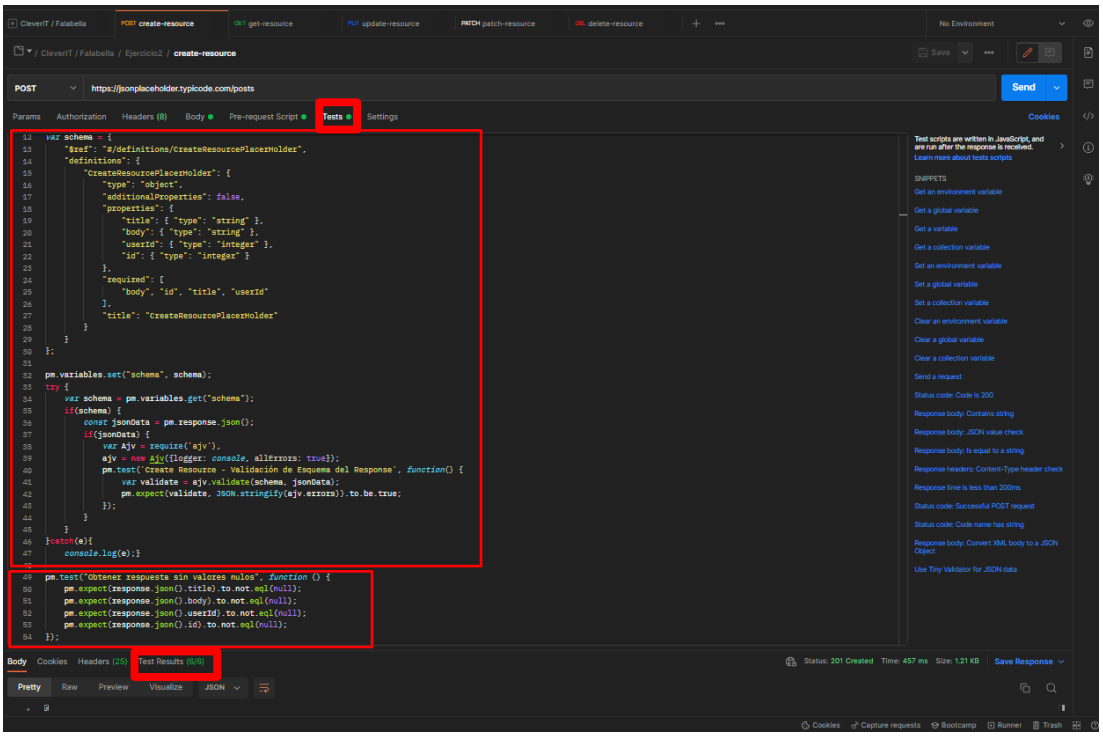


2. **Ahora realizamos nuestros test-case en la pestaña "Tests"**
    a. Code – Status: 201 – Created
    b. Tiempo de Petición: Menor que 1 Segundo.



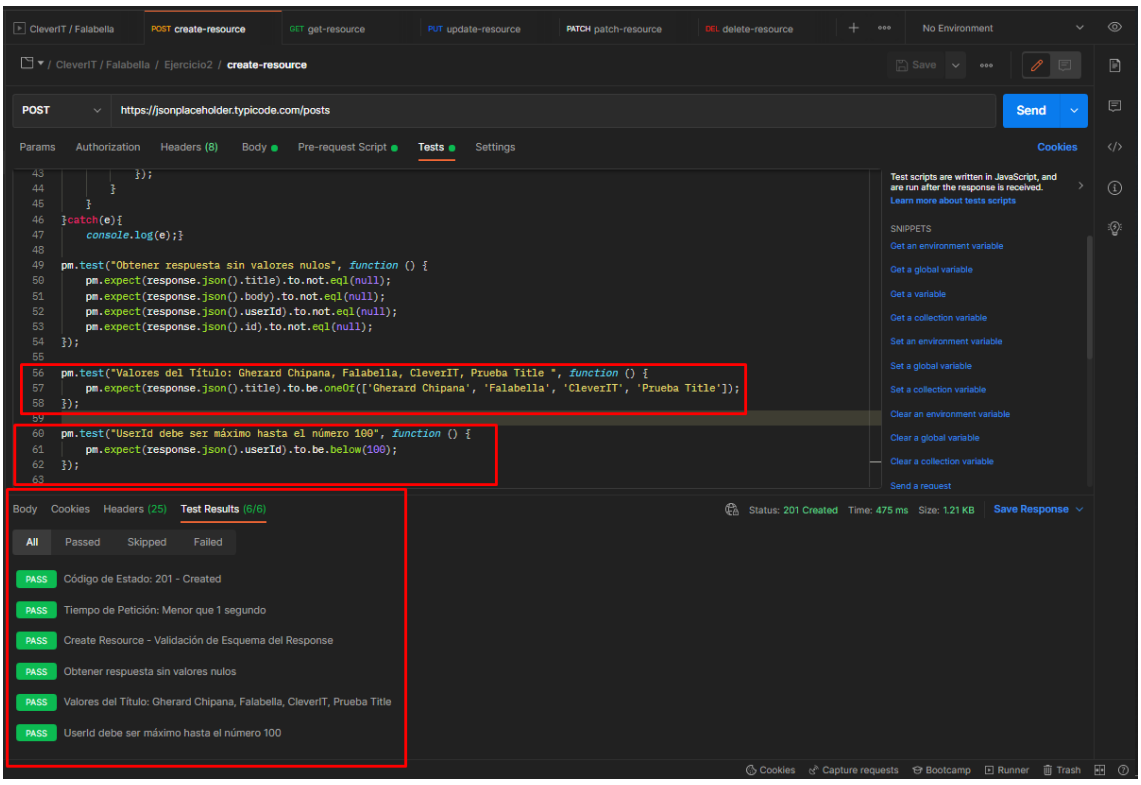
c. Validar que el esquema del response sea como el resultado esperado.

d. Obtener response sin valores null.

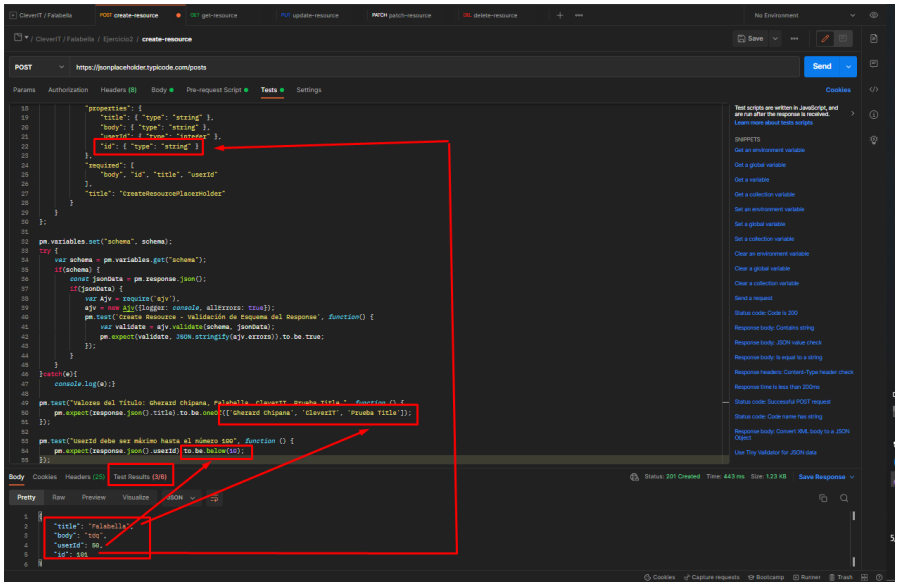


e. Que el campo **“title”** solo tenga los valores “Gherard Chipana, Falabella, CleverIT, Prueba Title”

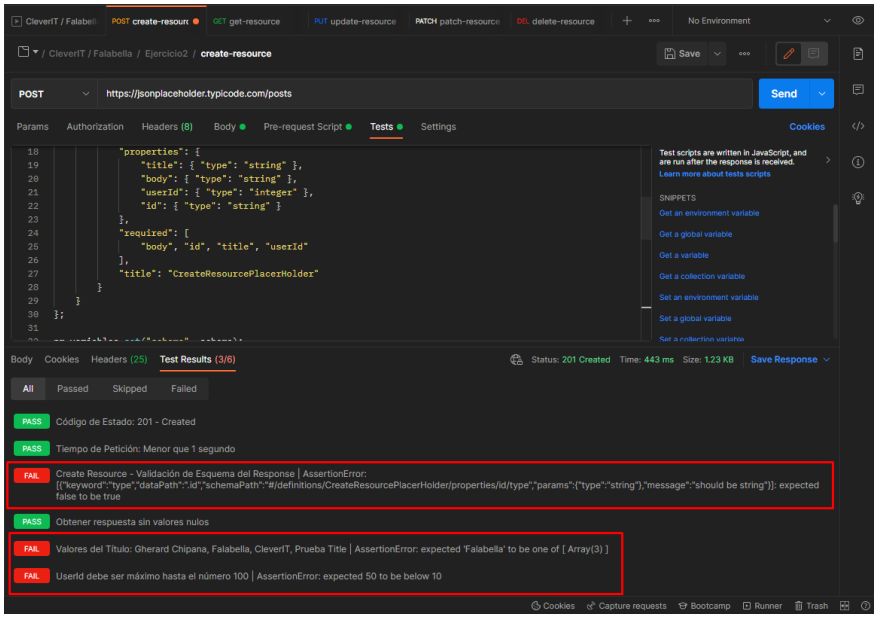
f. Que el campo **“userId”** tenga un valor máximo de 100



3. **Pruebas de Error**
    a. En las pruebas, siempre es importante no solo validar que todo responda bien, sino, qué pasa si enviamos valores erróneos, aquí la prueba de ello.



Se hizo caer tres casos de prueba, en el primero, el campo “id” que en el caso de prueba se indica debe ser string, falla, porque el response nos devuelve un integer; el segundo, en el campo “title” no se está indicando el valor Falabella, y como en el response devuelve Falabella, por eso falla; por último, el campo “userId” debe ser menor a 10, y como el response devuelve un 50, que es mayor a 10, nos arroja error.



En el Test Result nos indica el detalle de los errores ocurridos, lo cual, se muestra a los desarrolladores como bug del servicio en revisión.

DATOS ADICIONALES: Estas pruebas también pueden ser sincronizadas y desplegadas en Jenkins, en un archivo groovy se realiza la configuración del Pipeline y se puede subir a Github, GitLab o Bitbucket para luego realizar el despliegue en Jenkins, y, a través de un plugin llamada “Blue Ocean” se pueden generar reportes como en Visual Code, o, se puede realizar la impresión del mismo Jenkins.