Resultado “Prueba QA Automation”

Ejercicio 1 (QAA-01) API Temperatura

Se documenta los escenarios para la API, casos con parámetros válidos y parámetros inválidos.

Parámetros validos			
ID	Nombre	Datos de entrada	Resultado esperado

1	Consulta temperaturas para Chile- Santiago	**request params:** Fecha
		example: 17-02-2022
		Headers:
		Country
		example: Chile
		City
		example: Santiago
			Status code: 200 OK
			example response body:
			
			{
			"temperaturaActual": "23.4",
			"temperaturaDiaSiguiente":"25"
			}

2	Consulta temperaturas para Chile- Arica	**request params:** Fecha
		example: 17-02-2022
		Headers:
		Country
		example: Chile
		City
		example: Arica
			Status code: 200 OK
			example response body:
			
			{
			"temperaturaActual": "24",
			"temperaturaDiaSiguiente":"26"
			}

3	Consulta temperaturas para Chile- Chiloe	**request params:** Fecha
		example: 17-02-2022
		Headers:
		Country
		example: Chile
		City
		example: Chiloe
			Status code: 200 OK
			example response body:
			
			{
			"temperaturaActual": "17",
			"temperaturaDiaSiguiente":"18"
			}

4	Consulta temperaturas para Argentina- BuenosAires	**request params:** Fecha
		example: 17-02-2022
		Headers:
		Country
		example: Argentina
		City
		example: BuenosAires
			Status code: 200 OK
			example response body:
			
			{
			"temperaturaActual": "19",
			"temperaturaDiaSiguiente":"17"
			}

Yeferson Barrios

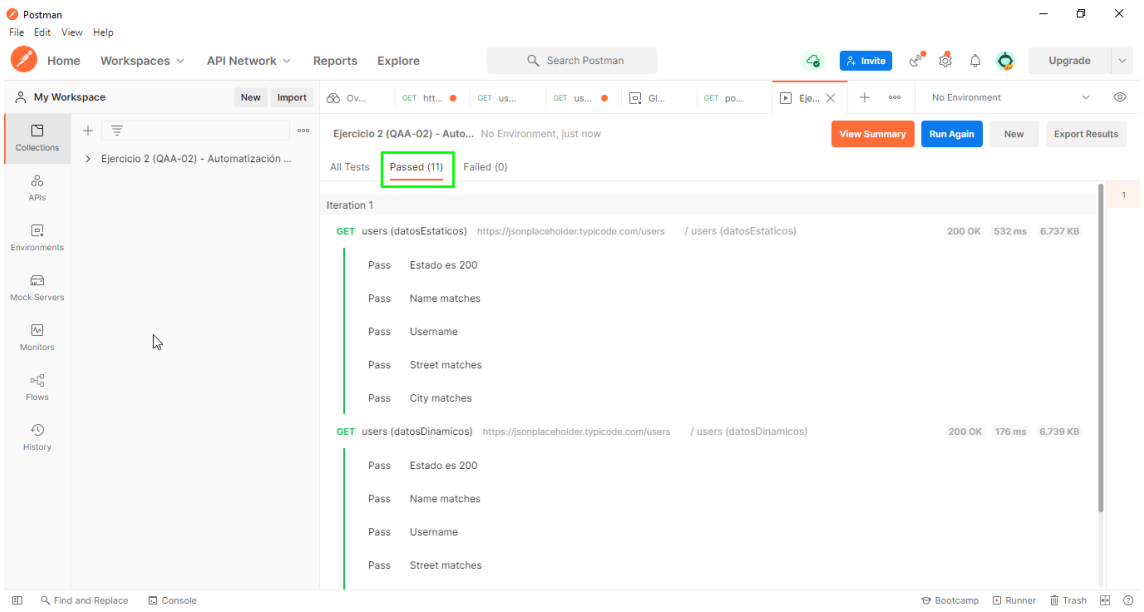
5	Consulta temperaturas para Argentina- SanJuan	**request params:** Fechaexample: 17-02-2022**Headers:**Countryexample: ArgentinaCityexample: SanJuan	Status code: 200 OKexample response body:{"temperaturaActual": "20","temperaturaDiaSiguiente":"21"}
Parámetros inválidos			
6	Consulta temperaturas con Fecha invalida	**request params:** Fecha example: 17/02/222 **Headers:** Country example: Argentina City example: SanJuan	Status code: 400 Bad Request response body: { responseresponseMessage: "Error in the request" }
7	Consulta temperaturas con Country invalido	**request params:** Fecha example: 17-02-2022 **Headers:** Country example: Bolivia City example: SanJuan	Status code: 422 response body: { responseMessage: "País y Ciudad incorrectos" }
8	Consulta temperaturas con City invalida	**request params:** Fecha example: 17-02-2022 **Headers:** Country example: Argentina City example: Cordoba	Status code: 422 response body: { responseMessage: "País y Ciudad incorrectos" }
9	Consulta temperaturas con Sensitive case ignorado	**request params:** Fechaexample: 17-02-2022**Headers:**Countryexample: argentinaCityexample: SaNjUan	Status code: 422response body:{responseMessage: "País y Ciudad incorrectos"}

10	Consulta temperaturas con Error de conexion	**request params:** Fecha
		example: 17-02-2022
		Headers:
		Country
		example: Argentina
		City
		example: SanJuan
			Status code: 500
			response body:
			{
			responseMessage: Internal Server Error
			}

Ejercicio 2 (QAA-02) - Automatización de API

Se deja un link de la collection de Postman:
https://www.getpostman.com/collections/eea7f9e2333657246c96

Los casos planteados fueron positivos



Ejercicio 3 (QAA-03) Explicación de técnicas

Para el ejercicio 1: Se realizo una tabla que contiene los posibles casos de pruebas a realizar a la API que se usaba como ejemplo, esta dividida en dos secciones (con parámetros validos e “invalidos”) y consta de cuatro columnas, en las que se identifica el numero de caso, un nombre que identifica el caso, los parámetros de invocación y los resultados esperados.

Para el ejercicio 2:

En los siguientes ejemplos, crearemos solicitudes reales con datos y desarrollaremos pruebas que cumplan con nuestros requisitos.

Hay dos tipos de datos de prueba que configuraremos, ambos con diferentes propósitos:

Ejemplo de datos estáticos

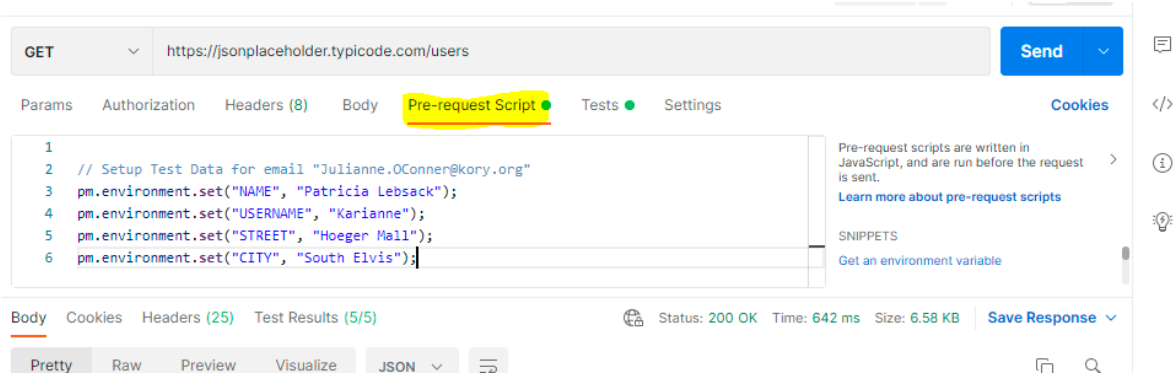
Estos son datos de prueba que conocemos de antemano y deben devolverse al final de nuestra prueba; esto incluye valores de configuración o datos que no deben cambiar entre implementaciones. No es necesario configurar estos valores en la sección **Prescript** y se pueden almacenar en la sección **test**.

Método: GET

Punto final: https://jsonplaceholder.typicode.com/users

Prescript Section:

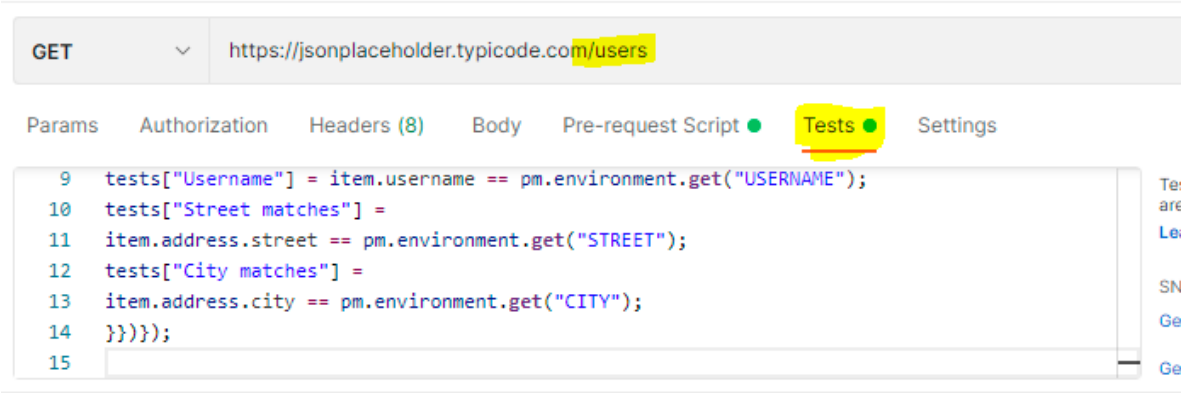
```
// seteamos un lote de prueba para Julianne.OConner@kory.org
pm.environment.set("NAME", "Patricia Lebsack");
pm.environment.set("USERNAME", "Karianne");
pm.environment.set("STREET", "Hoeger Mall");
pm.environment.set("CITY", "South Elvis");
```



Test Section:

```
pm.test("Estado es 200", function () {
pm.response.to.have.status(200);
// parsear respuesta a JSON
var jsonData = pm.response.json();
// iterar a traves de una list de objetos JSON
jsonData.each(function(item){
if (item.email === "Julianne.OConner@kory.org") {
tests["Name matches"] = item.name == pm.environment.get("NAME");
tests["Username"] = item.username == pm.environment.get("USERNAME");
tests["Street matches"] =
item.address.street == pm.environment.get("STREET");
tests["City matches"] =
item.address.city == pm.environment.get("CITY");
```

}}}});



Es decir: hicimos un REQUEST a **/usuarios** y validamos (ASSERT) los datos en la respuesta JSON; en este caso:
Iterar a través de la respuesta JSON (lista de mapas JSON)
Ubicar el mapa con el correo electrónico Julianne.OConner@kory.org
Los valores relacionados de afirmación (NOMBRE, NOMBRE DE USUARIO, CALLE, CIUDAD) coincidieron con nuestros valores esperados establecidos en el paso Prescript.

Esto fue una prueba simple para confirmar que un solo punto final nos devuelve los valores esperados.

Ejemplo de datos dinámicos:

Estos son datos de prueba que no conocemos de antemano, pero sabemos dónde recuperarlos: datos almacenados en una base de datos o punto final de API que devolverá los datos necesarios.
En este ejemplo, usaremos dos solicitudes: una para obtener nuestros datos de prueba y otra para afirmar.

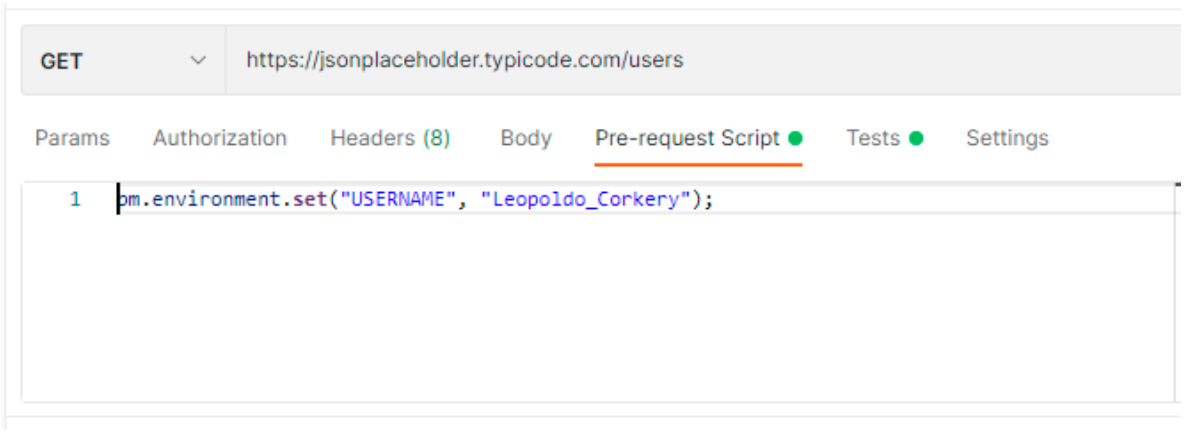
Paso #1: Obtener USER_ID dado NOMBRE DE USUARIO

Método: GET

ENDPOINT: https://jsonplaceholder.typicode.com/users

Prescript Section:
// seteamos un lote de prueba: SOLICITAREMOS EL USER_ID dado el USERNAME

pm.environment.set("USERNAME", "Leopoldo_Corkery");



Test Section

```
pm.test("estado es 200", function () {
pm.response.to.have.status(200);
// parseo a JSON
var jsonData = pm.response.json();
// ITERAMOS LISTA DE OBJETOS JSON
jsonData.each(function(item){
if (item.username == pm.environment.get("USERNAME")) {
// SETEAMOS EL user_id
pm.environment.set("USER_ID", item.id)
}
else {
// acá haríamos algo por si no apareciera el user_id
}
})});
```

Ahora tenemos el USER_ID que necesitamos para nuestra próxima solicitud.

Paso #2: Obtener una lista de POST_ID para un USER_ID dado

A continuación, solicitamos /posts con parámetro de URL {{USER_ID}}

Método: GET

endpoint: https://jsonplaceholder.typicode.com/posts?userId={{USER_ID}}

Prescript Section: ninguna

TEST Section:

```
var list_of_post_ids = [];

pm.test("Main Test", function () {
```

```
    // Parse response to JSON

    var jsonData = pm.response.json();
    jsonData.each(function(item){

    list_of_post_ids.push(item.id);


    // Condition: Status code is 200
    if(pm.response.code === 200) {
       console.log("It's 200")
       tests["200 specific test of post_ids"] = 9
}


    // Condition: Status code is 403
    if(pm.response.code === 403) {
       console.log("It's 403")
       tests["403 specific test of post_ids"] = 2
       tests["Another 403 specific test of post_ids"] = 5
})})
```

En nuestra TEST SECTION, ahora armamos una lista de post_ids para el USER_ID dado.

Ahora podemos validar fácilmente cualquier respuesta de la API JSON y adaptar los casos de prueba.