

**Autor:** Liseth Troncoso

## **Reto técnico pruebas de servicio con Postman**

### **Primera parte:**

Valentin Despa es un desarrollador de software que está trabajando en un proyecto, en el cual hace un servicio Rest, para poder ver libros disponibles, así como poder hacer ordenes de libros, así como actualización de estas órdenes, y hacer un borrado de una orden en caso de ser necesario. Por este motivo después de terminado el desarrollo del servicio el desarrollador quiere probar el correcto funcionamiento de este y para esto contrata a la empresa Choucair Testing.

Por este motivo el desarrollador le proporciona a Choucair Testing la documentación de la Api en el siguiente link:

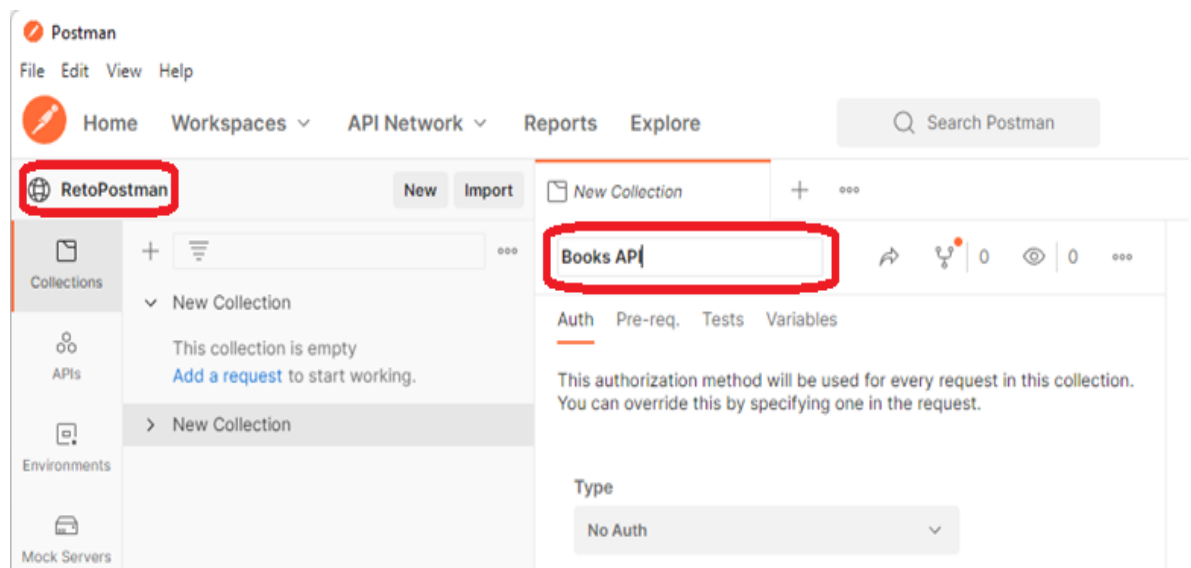
<https://github.com/vdespa/introduction-to-postman-course/blob/main/simple-books-api.md>.

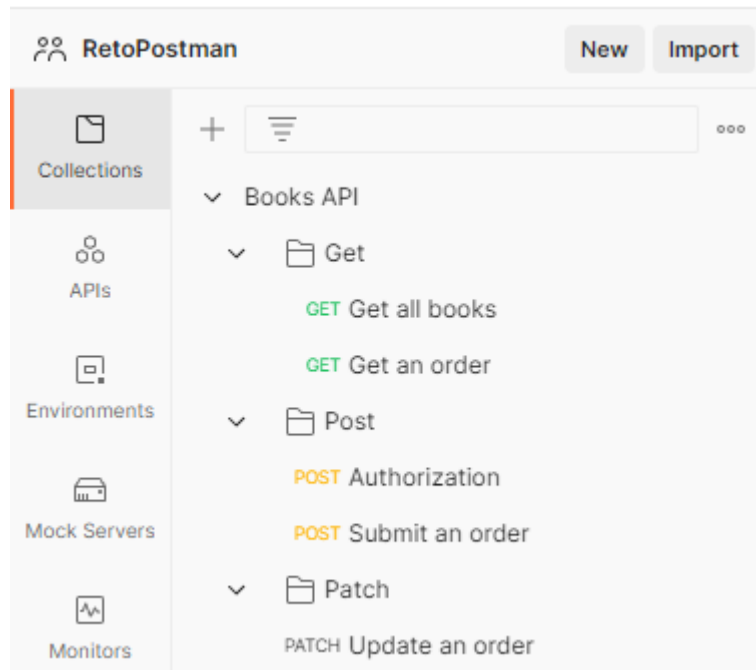
Ya después de haber leído la documentación, con el cliente se llega a un acuerdo de que las pruebas realizadas se harán en Postman, y que deben estar organizadas en una colección que se divida en subcarpetas en donde nos permitan ver de mejor manera que endpoint se está probando. Por otra parte lo que el cliente espera que tenga nuestras pruebas en la herramienta de Postman es lo siguiente:

1. Se debe crear un ambiente de pruebas para hacer la ejecución de nuestras pruebas.
2. También se hace necesario el uso de variables desde el ambiente creado.
3. Poder hacer un crud con cada uno de los endpoints del servicio (incluir por lo menos un camino feliz y uno no feliz).
4. Cada una de las validaciones que se hacen deben estar respaldadas con aserciones que lo justifiquen, no solo validaciones de código de respuesta.

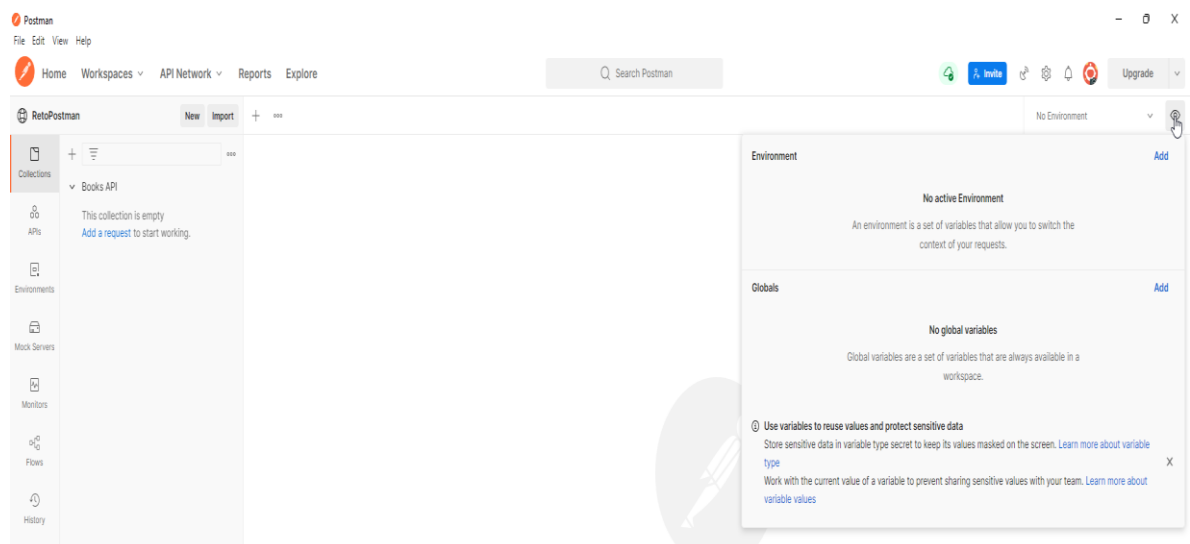
## Solución:

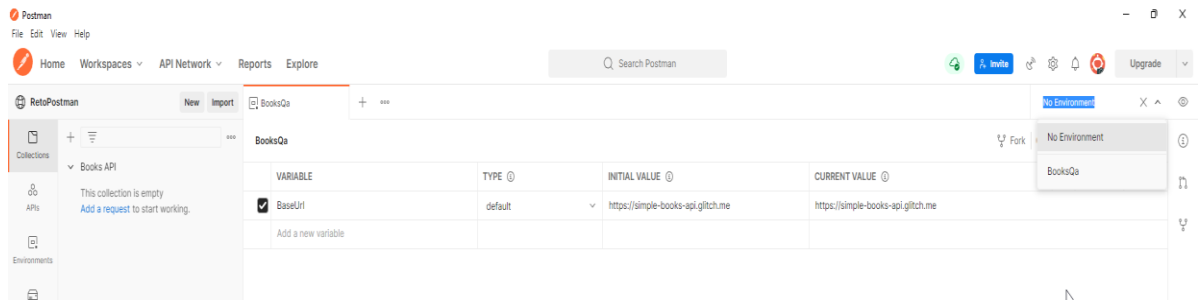
- Iniciaremos con la creación de nuestro espacio de trabajo, el cual se llama **RetoPostman**, Postman nos ofrece tres tipos diferentes de espacios de trabajo: espacios de trabajo personales, espacios de trabajo de equipo y espacios de trabajo públicos.
- Creamos la colección **Books API**, al dar clic en New Collection y escribir el nombre, esta colección se dividirá en subcarpetas que nos permitan ver de mejor manera que endpoint se está probando.



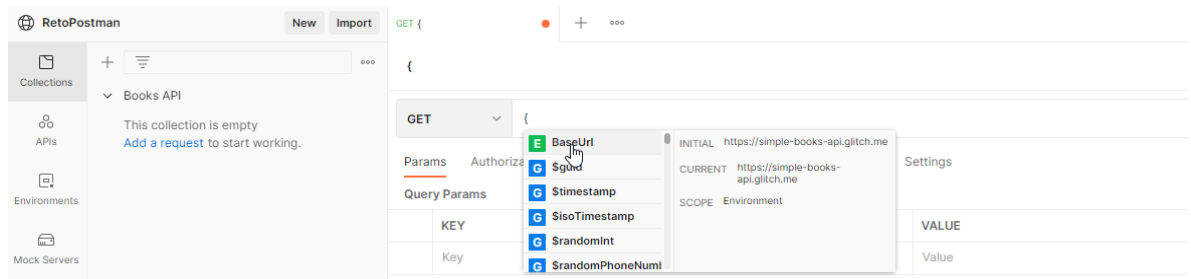


- Creación de ambiente de pruebas para hacer la ejecución de nuestras pruebas y uso de variables desde el ambiente creado:
  1. Da clic en el ícono de la esquina superior derecha para mostrar los environment o da clic en la pestaña Environment
  2. Da clic en el botón Add
  3. Agrega un nombre al environment. En este caso es **BookQa**.
  4. Agrega una nueva variable con los siguientes datos:  
 Variable: **BaseUrl**  
 Initial value: **https://simple-books-api.glitch.me**
  5. Da clic en Save, si todo es correcto se muestra el Environment.



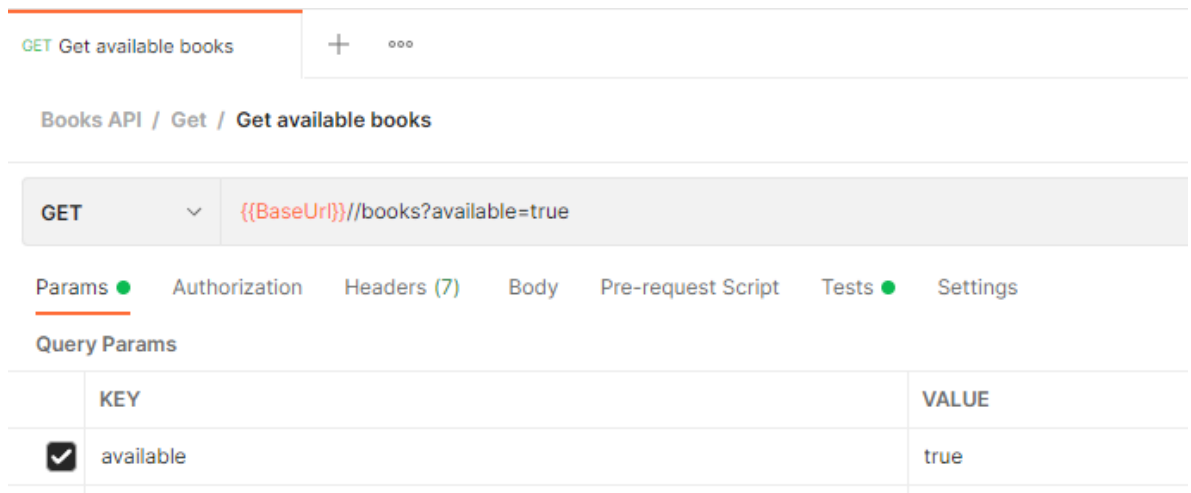


- Validaremos la variable creada realizando nuestra primera solicitud:

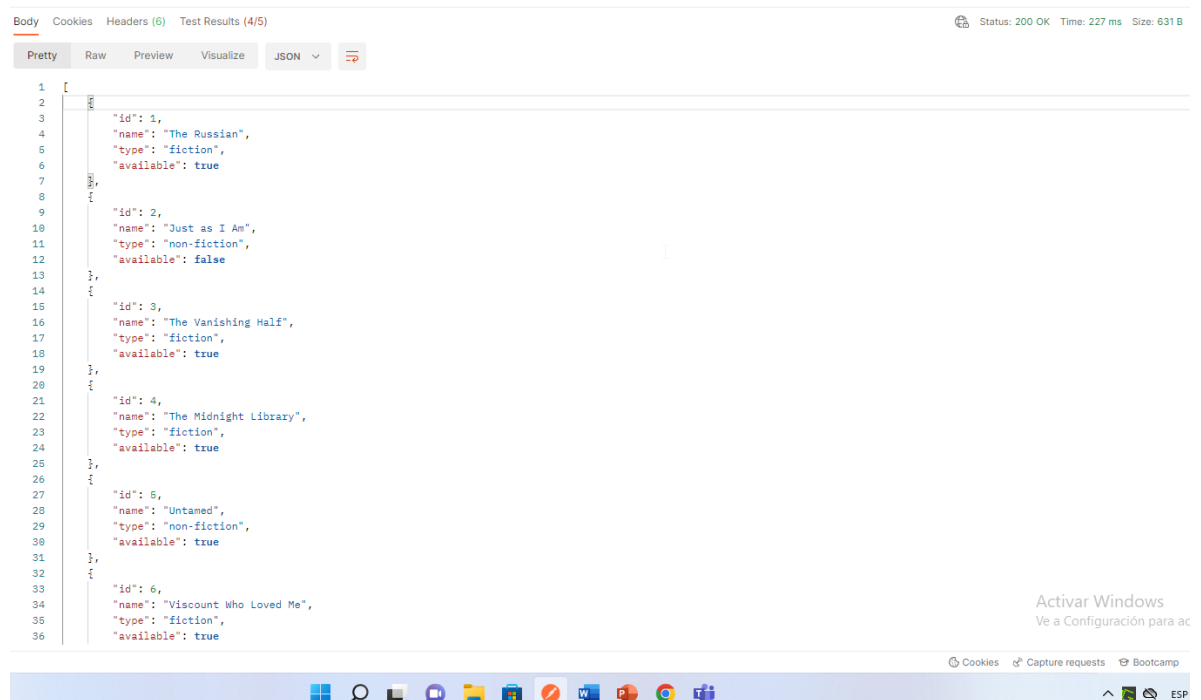


## 1) Obtener libros disponibles:

Request:



Response:



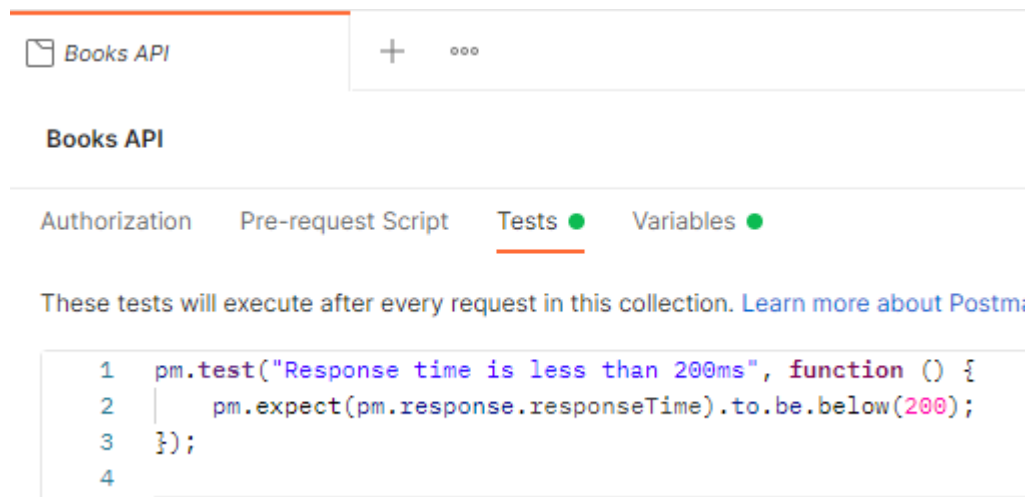
```
1 [{"id": 1,
2   "name": "The Russian",
3   "type": "fiction",
4   "available": true
5 },
6 {"id": 2,
7   "name": "Just as I Am",
8   "type": "non-fiction",
9   "available": false
10 },
11 {"id": 3,
12   "name": "The Vanishing Half",
13   "type": "fiction",
14   "available": true
15 },
16 {"id": 4,
17   "name": "The Midnight Library",
18   "type": "fiction",
19   "available": true
20 },
21 {"id": 5,
22   "name": "Untamed",
23   "type": "non-fiction",
24   "available": true
25 },
26 {"id": 6,
27   "name": "Viscount Who Loved Me",
28   "type": "fiction",
29   "available": true
30 }]
```

Como vemos la API no realiza esa función, es un error, pero se puede considerar como una sugerencia ya que en la documentación no recomienda filtrar libros por disponibles sino por:

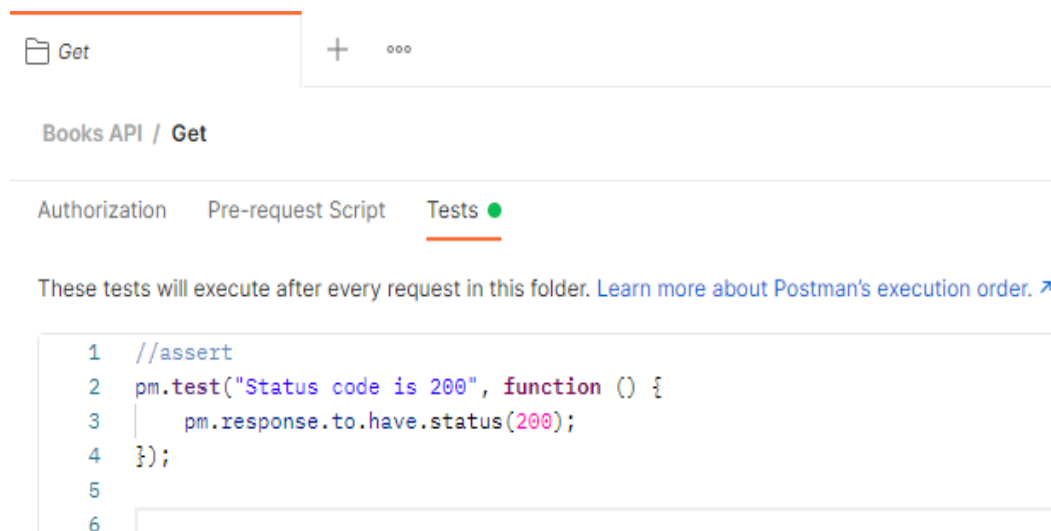
- type: fiction or non-fiction
- limit: a number between 1 and 20.

### Validaciones:

- En la ejecución de todas las pruebas se validará que el tiempo de respuesta de las solicitudes sea inferior a los 200 ms, este test de prueba se aplica a nivel de colección:



- En la ejecución las pruebas con el método Get se validará que el código de respuesta devuelto por la API, si es 200 la prueba pasa, de lo contrario falla, esta prueba se aplica a nivel de carpeta:



- Se valido que el que el tiempo de respuesta de la solicitud fuera inferior a 200ms y falla porque el tiempo fue de 504ms.  
Se valido que el código de respuesta devuelto por la API fuera de 200 y pasa la prueba.  
Se valido que la API tenga un encabezado de respuesta tipo contenido y que tenga el valor de application/json; charset=utf-8.  
Se validan los datos y sus tipos que debe contener el documento Json  
Y por ultimo se comprueba que se hace el filtro por libros disponibles, essta prueba falla porque la API no realiza esa función.

GET Get available books

Books API / Get / Get available books

GET{{BaseUrl}}/books?available=true

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

```
1 pm.test("Content-Type is present", function () {
2   pm.response.to.have.header("Content-Type");
3 });
4
5 pm.test("Content-Type header is application/json", () => {
6   pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json; charset=utf-8');
7 });
8
9 var jsonData = pm.response.json();
10 var schema = {
11   "items": {
12     "type": "object",
13     "properties": {
14       {
15         "id":{"type":"number"},
16         "name":{"type":"string"},
17         "type":{"type":"string"},
18         "available":{"type":"boolean"}
19       }
20     }
21   };
22
23 pm.test("Validate schema", () => {
24   pm.response.to.have.jsonSchema(schema);
25 });
26
27 const response = pm.response.json();
28 const availableBooks = response.filter((book) => book.available === 'true');
29
30 const book = availableBooks[0];
31 pm.test('book found by available', ()=> {
32   pm.expect(book.available).to.eql('true')
33 });
```

BodyCookiesHeaders (6)Test Results (4/6)

AllPassedSkippedFailed

FAIL

Response time is less than 200ms | AssertionError: expected 529 to be below 200

PASS

Status code is 200

PASS

Content-Type is present

PASS

Content-Type header is application/json

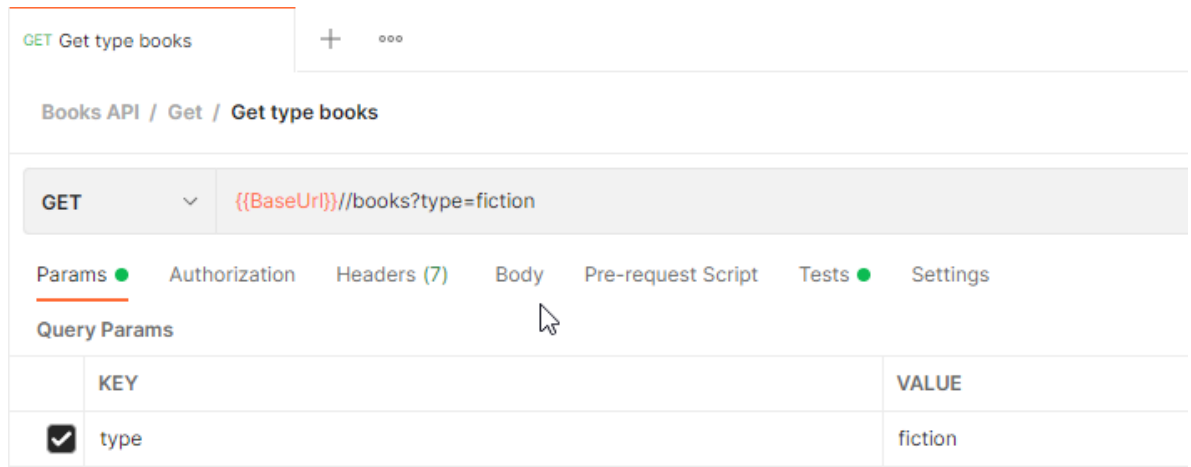
PASS

Validate schema

FAIL

book found by available | TypeError: Cannot read properties of undefined (reading 'available')

Obtener libros de ficción:

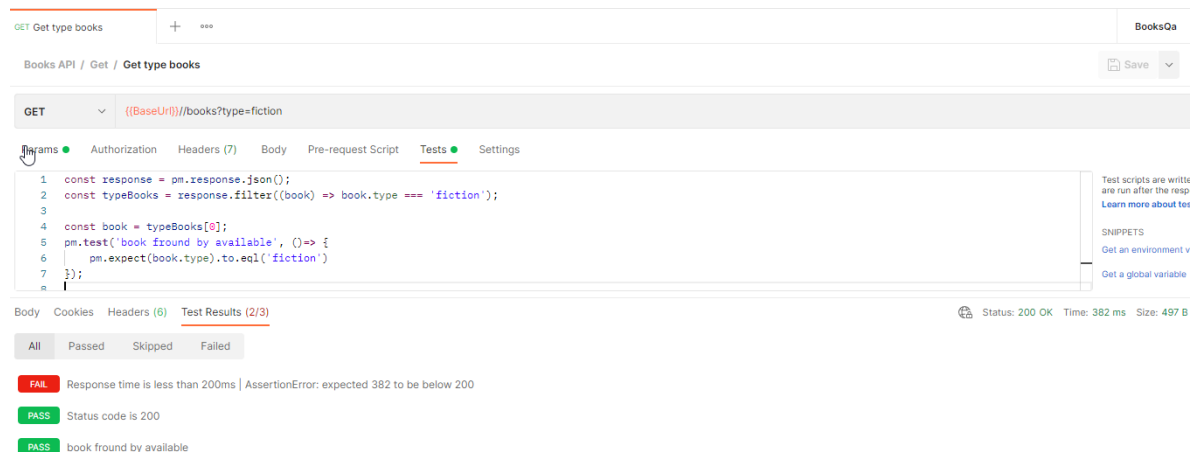


## Validaciones:

Se valido que el que el tiempo de respuesta de la solicitud fuera inferior a 200ms y falla porque el tiempo fue de 382ms.

Se valido que el código de respuesta devuelto por la API fuera de 200 y pasa la prueba.

Y por último se comprueba que se hace el filtro por libros de ficción y es exitosa.

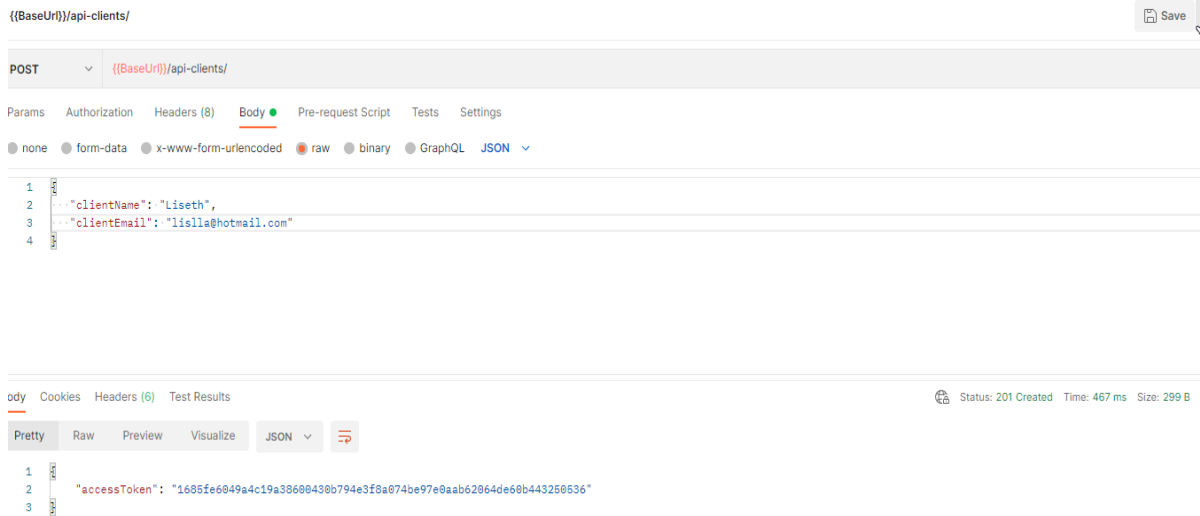


## 2) Crear una orden

Para realizar esta solicitud necesitamos estar registrados en la API e incluir las siguientes propiedades: id del libro y el nombre del cliente.

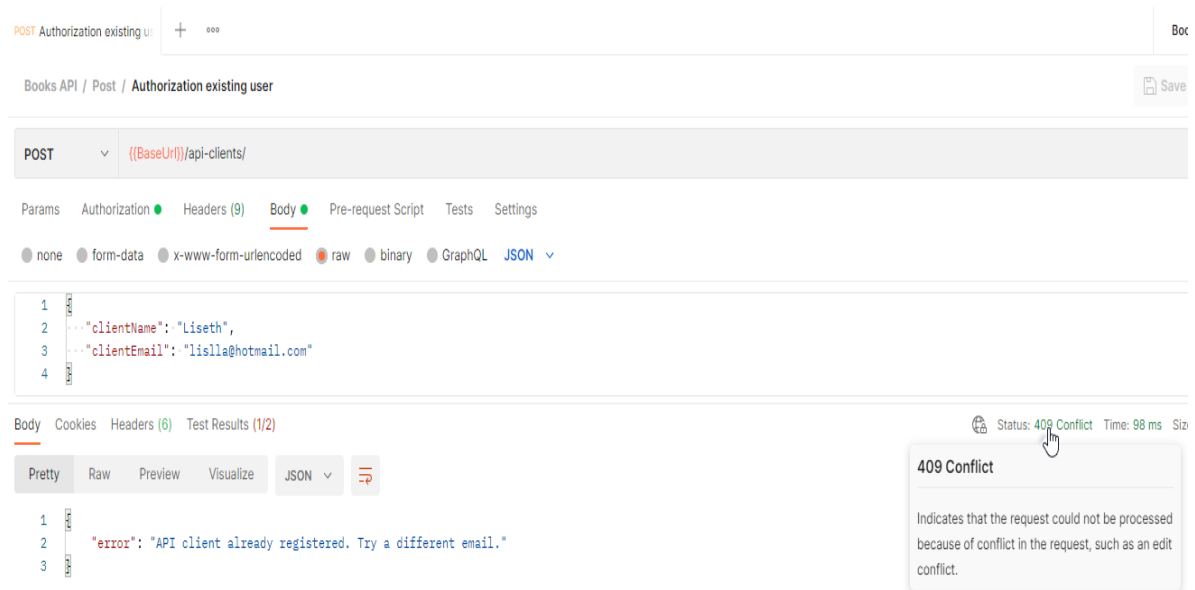
- Primeramente, procederemos a autenticarnos en la API, realizando la siguiente solicitud:





Y obtenemos el `accessToken` para poder realizar nuestra orden.

Como camino no feliz intentamos registrarnos con un usuario existente:



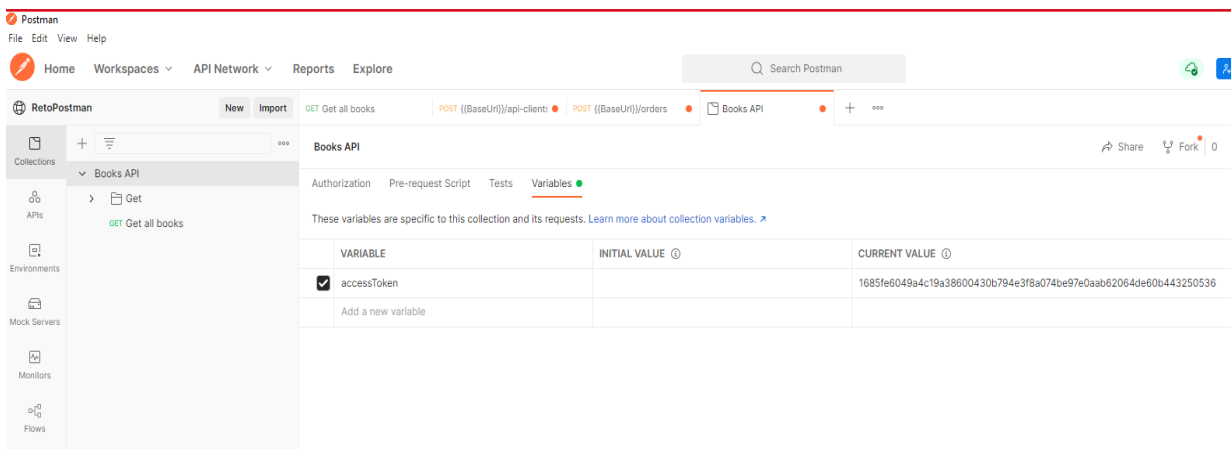
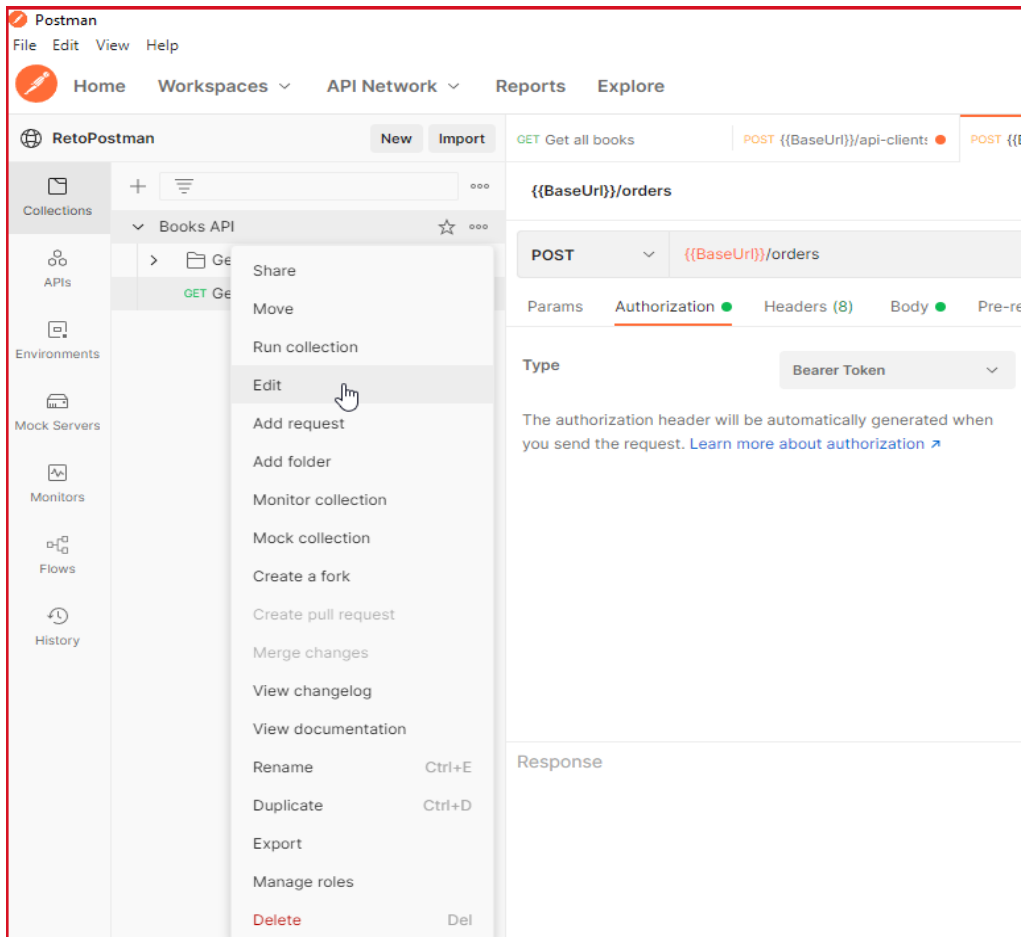
Y obtenemos el siguiente error que nos dice que ya este usuario existe:

"error": "API client already registered. Try a different email."

Además, el código de respuesta de la solicitud es Status 409 Conflict, indicando que la solicitud no pudo ser procesada debido a un conflicto en la solicitud, como un conflicto de edición.

- **Creación de la variable Token:**

Se puede realizar de dos maneras a nivel de colección o en la petición, de la primera forma, damos clic en los tres puntos del lado izquierdo donde esta la colección y le damos en edit, en el menú buscamos la opción variable, colocamos el nombre de la variable accessToken y su valor en initial value que comparte esta variable en la colección o en el entorno de ambiente y current value solo utiliza este valor en la solicitud.



También se puede crear la variable Token en la solicitud, vamos a la pestaña de autorización, seleccionamos el tipo de dato **Type: Bearer Token** y en token escribimos `{{accessToken}}` el nombre de nuestra variable con su formato y automáticamente nos va a aparecer en la cabecera de la solicitud.

POST Authorization

Books API / Post / Authorization

POST

{{BaseUrl}}/api-clients/

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collabo

Learn more about variables

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token

{{accessToken}}

Response

POST Submit an order

Books API / Post / Submit an order

POST

{{BaseUrl}}/orders

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

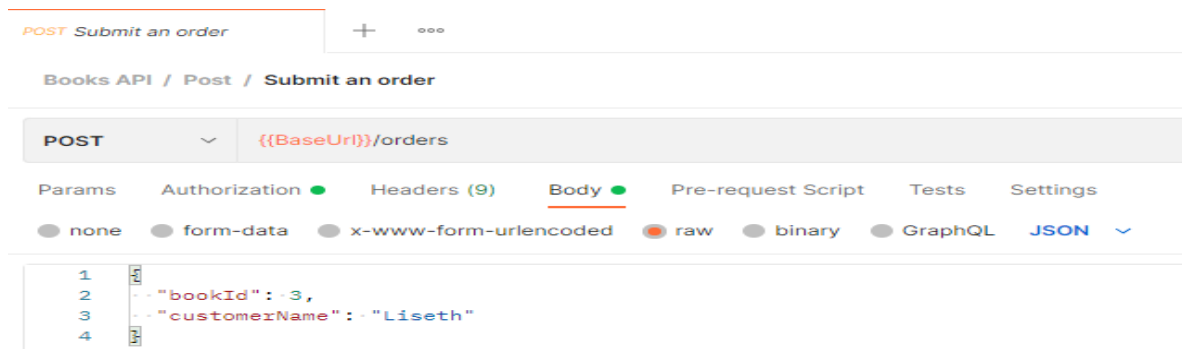
Headers

Hide auto-generated headers

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	Bearer 1685fe6049a4c19a38600430b794e3f8a074be97e0aab62064de60b44...	
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Content-Length	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
Key	Value	Description

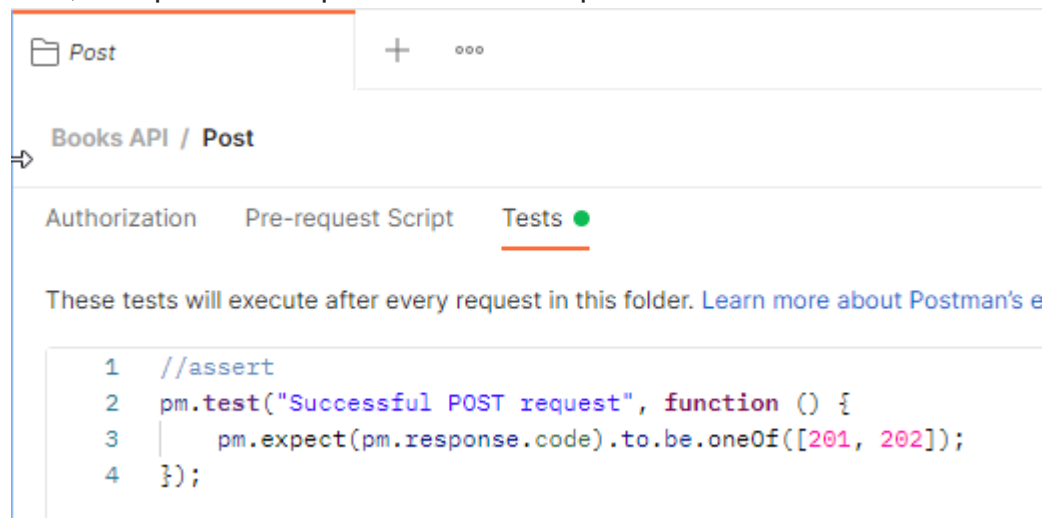
Response

Y procederemos a realizar la solicitud:



## Validaciones:

- En la ejecución las pruebas con el método Post se validará que el código de respuesta devuelto por la API, si es 201 o 202 la prueba pasa, de lo contrario falla, esta prueba se aplica a nivel de carpeta:



Se valida que el que el tiempo de respuesta de la solicitud fuera inferior a 200ms y falla porque el tiempo fue de 663ms.

Se valida que el código de respuesta devuelto por la API fuera de 201 o 202 y pasa la prueba.

Y por último se comprueba que el orderId si existe.

Además se extrae la variable de orderId y se guarda en una variable global para que se actualice automáticamente cada vez que se ejecuta un endpoint que utilice esta variable.

POST Submit an order

Books API / Post / Submit an order

POST {{BaseUrl}}/orders

Params Authorization Headers (9) Body Pre-request Script Tests Settings

```
1 const response = pm.response.json();
2
3 const hasorderId = Object.keys(response).includes('orderId')
4
5 pm.test("Validate orderId", () => {
6   pm.expect(true).to.eql(hasorderId);
7 });
8
9 pm.globals.set('orderId', response.orderId);
```

Body Cookies Headers (6) Test Results (2/3)

Pretty Raw Preview Visualize JSON

```
1
2   "created": true,
3   "orderId": "h546YHrjASilJYi_d-bF9"
4
```

Body Cookies Headers (6) Test Results (2/3) Status: 201 Created Time: 663 ms

All Passed Skipped Failed

**FAIL** Response time is less than 200ms | AssertionError: expected 663 to be below 200

**PASS** Successful POST request

**PASS** Validate orderId

Si verificamos la orden generada con un endpoint tenemos:

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

RetoPostman New Import

Collections

- Books API
  - Get
    - GET Get all books

GET Get an order PATCH Update an order

Books API / Get / Get an order

GET {{BaseUrl}}/orders:orderId

Params Authorization Headers (7) Body

Headers		Hide auto-generated headers
KEY	VALUE	
<input checked="" type="checkbox"/> Authorization	Bearer 1685fe6049a4c19a38600430b794e3f8a074be97e0aab62064de60b44...	
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	

body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": "cPKrxCeW9dgU7MGo9GeNI",
3    "bookId": 3,
4    "customerName": "Liseth",
5    "createdBy": "7a184406768d05187d8922acf605f42be19da1756246769425886a6b17f8af94",
6    "quantity": 1,
7    "timestamp": 1653692145128
8  }

```

Realizaremos una solicitud con datos erróneos y veremos si la API nos permite realizar la solicitud:

POST Submit an order wrong + ...

Books API / Post / Submit an order wrong data

POST

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

```

1  {
2    "bookId": Jose,
3    "customerName": "11142545"
4  }

```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text

1 Unprocessable Entity

Status: 422 Unprocessable Entity Time: 1

**422 Unprocessable Entity**

The request was well-formed but was unable to be followed due to semantic errors.

El test de prueba nos genera un error 422 que nos indica que tenemos errores semánticos en nuestra prueba.

### 3) Actualizar una orden:

PATCH Update an order + ...

Books API / Patch / Update an order

PATCH {{BaseUrl}}/orders/:orderId

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Path Variables

KEY	VALUE
orderId	cPKrxCeW9dgU7MGo9GeNI

PATCH Update an order + ...

Books API / Patch / Update an order

PATCH {{BaseUrl}}/orders/:orderId

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

```
1 PATCH
2 {
3   "customerName": "{{randomFullName}}"
}
```

Utilizamos una variable aleatoria dinámica para actualizar una orden

PATCH Update an order

+

...

Books API / Patch / Update an order

PATCH

▼

{{BaseUrl}}/orders/:orderId

Params ● Authorization ● Headers (9) Body ● Pre-request Script Tests ● Settings

```
1 pm.test("Date is present", function () {
2   |   pm.response.to.have.header("Date");
3   | });
4
5 pm.test("Status code name has string", function () {
6   |   pm.response.to.have.status("No Content");
7   | });
```

Body Cookies Headers (3) Test Results (3/4)

All Passed Skipped Failed

FAIL

Response time is less than 200ms | AssertionError: expected 554 to be below 200

PASS

Status code is 204

PASS

Date is present

PASS

Status code name has string

## Validaciones:

- En la ejecución las pruebas con el método Patch se validará que el código de respuesta devuelto por la API, si es 204 la prueba pasa, de lo contrario falla, esta prueba se aplica a nivel de carpeta.

Se valido que el que el tiempo de respuesta de la solicitud fuera inferior a 200ms y falla porque el tiempo fue de 554ms.

Se valido que el código de respuesta devuelto por la API fuera de 204 y pasa la prueba.

Se valido que el dato Date este presente en la cabecera

Y por último se validó que el contenido del código status fuera no content

Si realizamos una solicitud con el campo obligatorio customerName vacío, de hecho paso la prueba pero no debió tomar la solicitud, esto seria un error.



PATCH Update an order void + ...

Books API / Patch / Update an order void

PATCH {{BaseUrl}}/orders/:orderId

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

```
1 {
2   "customerName": ""
3 }
```

Body Cookies Headers (3) Test Results (3/4)

All Passed Skipped Failed

**FAIL** Response time is less than 200ms | AssertionError: expected 595 to be below 200

**PASS** Status code is 204

**PASS** Date is present

**PASS** Status code name has string

Si validamos con un endpoint la orden actualizada tenemos:

GET Get an order PATCH Update an order + ...

Books API / Get / Get an order

GET {{BaseUrl}}/orders/:orderId

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers Hide auto-generated headers

	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization	Bearer 1685fe6049a4c19a38600430b794e3f8a074be97e0aab62064de60b44...
<input checked="" type="checkbox"/>	Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.0
<input checked="" type="checkbox"/>	Accept	*/*
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "cPKrxCeW9dgU7MGo9GeNI",
3   "bookId": 3,
4   "customerName": "Toni Bradtke",
5   "createdBy": "7a184406768d05187d8922acf605f42be19da1756246769425886a6b17f8af94",
6   "quantity": 1,
7   "timestamp": 1653692145128
8 }
```

#### 4) Borrar una orden:

DEL Delete an order Copy

Books API / Delete / Delete an order Copy

DELETE

{{BaseUrl}}/orders/:orderId

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Path Variables

KEY	VALUE
orderId	cPKrxCeW9dgU7MGo9GeNI

DEL Delete an order Copy

Books API / Delete / Delete an order Copy

DELETE

{{BaseUrl}}/orders/:orderId

Params Authorization Headers (7) Body Pre-request Script Tests Settings

```
1 pm.test("Check the active environment", () => {
2   pm.expect(pm.environment.name).to.eql("BooksQa");
3 });
```

Body Cookies Headers (3) Test Results (2/3)

All Passed Skipped Failed

FAIL

Response time is less than 200ms | AssertionError: expected 273 to be below 200

PASS

Status code is 204

PASS

Check the active environment

#### Validaciones:

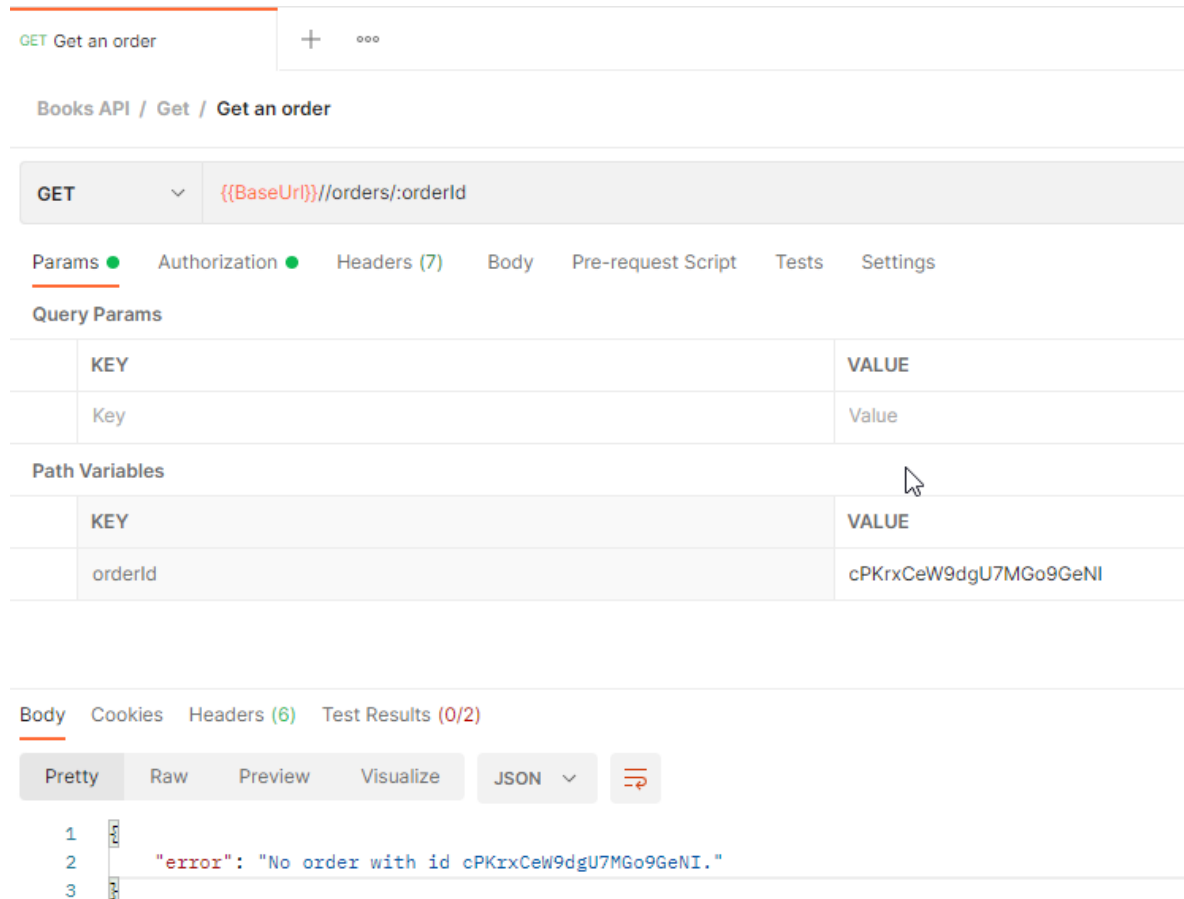
- En la ejecución las pruebas con el método Delete se validará que el código de respuesta devuelto por la API, si es 204 la prueba pasa, de lo contrario falla, esta prueba se aplica a nivel de carpeta.

Se valido que el que el tiempo de respuesta de la solicitud fuera inferior a 200ms y falla porque el tiempo fue de 273ms.

Se valido que el código de respuesta devuelto por la API fuera de 204 y pasa la prueba.

Se valido que el entorno de prueba BooksQa estuviera activo al ejecutar la prueba.

Si buscamos la orden eliminada con un endpoint tenemos:



GET Get an order

Books API / Get / Get an order

GET `{{BaseUri}}/orders/:orderId`

Params ● Authorization ● Headers (7) Body Pre-request Script Tests Settings


Query Params

KEY	VALUE
Key	Value

Path Variables

KEY	VALUE
orderId	cPKrxCeW9dgU7MGo9GeNI

Body Cookies Headers (6) Test Results (0/2)

Pretty Raw Preview Visualize JSON 

```
1 {  
2   "error": "No order with id cPKrxCeW9dgU7MGo9GeNI."  
3 }
```

## Segunda parte:

Para esta segunda parte vamos a trabajar sobre la siguiente Api <https://rickandmortyapi.com/documentation> , la cual es muy usada para hacer pruebas de servicio, para este ejercicio se plantea hacer solicitudes sobre el endpoint de "Character" y se espera puedan hacer lo siguiente con este endpoint:

1. Hacer un listado de todos los personajes, y hacer un test con la información de un personaje que aparezca en esa respuesta (Tener en cuenta que deben hacer una búsqueda entre los diferentes objetos que hay anidados en la respuesta)
2. Hacer un listado de personajes por página (en la documentación se explica cómo hacer paginación), y hacer un test por aparición de personaje en esta página.
3. Traer un personaje por id, y hacer un test correspondiente.
4. Al momento de buscar múltiples Id, ingresando id's que no existen nos da una respuesta, la cual es un array vacío, hacer un test en donde se valide que se tiene una respuesta vacía

## Solución:

### 1. Listado de todos los personajes:

GET Get all characters ● GET Get a single character ● GET Get all characters page ● GET Get m

Rick and morty API / Get all characters

GET {{Urlbase}}/character

Params Authorization Headers (6) Body Pre-request Script Tests ● Settings

```
1 pm.test("name is Rick Sanchez", function () {
2   var jsonData = pm.response.json();
3   pm.expect(jsonData.results[0].name).to.be.eqls("Rick Sanchez");
4 });
5
```

Body Cookies Headers (14) Test Results (1/1)

All Passed Skipped Failed

PASS name

## 2. Listado personaje por pagina:

GET Get all characters ● GET Get a single character ● GET Get all characters page ● GET Get multiple character: ●

Rick and morty API / Get all characters page

GET {{Urlbase}}/character/?page=19

Params ● Authorization Headers (6) Body Pre-request Script Tests ● Settings

```
1 pm.test("API response contains the expected fields", () => {
2   var jsonData = pm.response.json();
3   pm.expect(jsonData.results[0].name).to.eqls("Toxic Rick");
4   pm.expect(jsonData.results[0].origin.name).to.eqls("Detoxifier");
5 });
```

Body Cookies Headers (14) Test Results (1/1)

All Passed Skipped Failed

**PASS** API response contains the expected fields

## 3. Traer personaje por Id:

GET Get all characters × GET Get a single character ● GET Get all characters page ● GET Get multiple cha

Rick and morty API / Get a single character

GET {{Urlbase}}/character/:id

Params ● Authorization Headers (6) Body Pre-request Script Tests ● Settings

```
1 pm.test("API response contains the expected fields", () => {
2   const response = pm.response.json();
3
4   pm.expect(response).to.have.property("id", 2);
5   pm.expect(response).to.have.property("name", "Morty Smith");
6 });
```

Body Cookies Headers (14) Test Results (1/1)

All Passed Skipped Failed

**PASS** API response contains the expected fields

## 4. Buscar múltiples id

GET Get all characters GET Get a single character GET Get all characters page GET Get multiple characters

Rick and morty API / Get multiple characters

GET

▼

{{Urlbase}}/character/2300,2301

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTests●Settings

```
1
2 pm.test("Verify the array empty" , function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData).to.be.an('array').that.is.empty;
5 });
```

BodyCookiesHeaders (12)Test Results (1/1)

AllPassedSkippedFailed

PASSVerify the status and name

### Publicación:

<https://documenter.getpostman.com/view/19967026/Uz5FJGdn>  
<https://documenter.getpostman.com/view/19967026/Uz5FJGdm>