

Guía de usuario

-Openapi2postman-

Diciembre 2020

/cloudappi

Control de Cambios

Versión	Autor	Fecha	Cambios
v1.0	Javier González	01/12/2020	Documento inicial

Índice

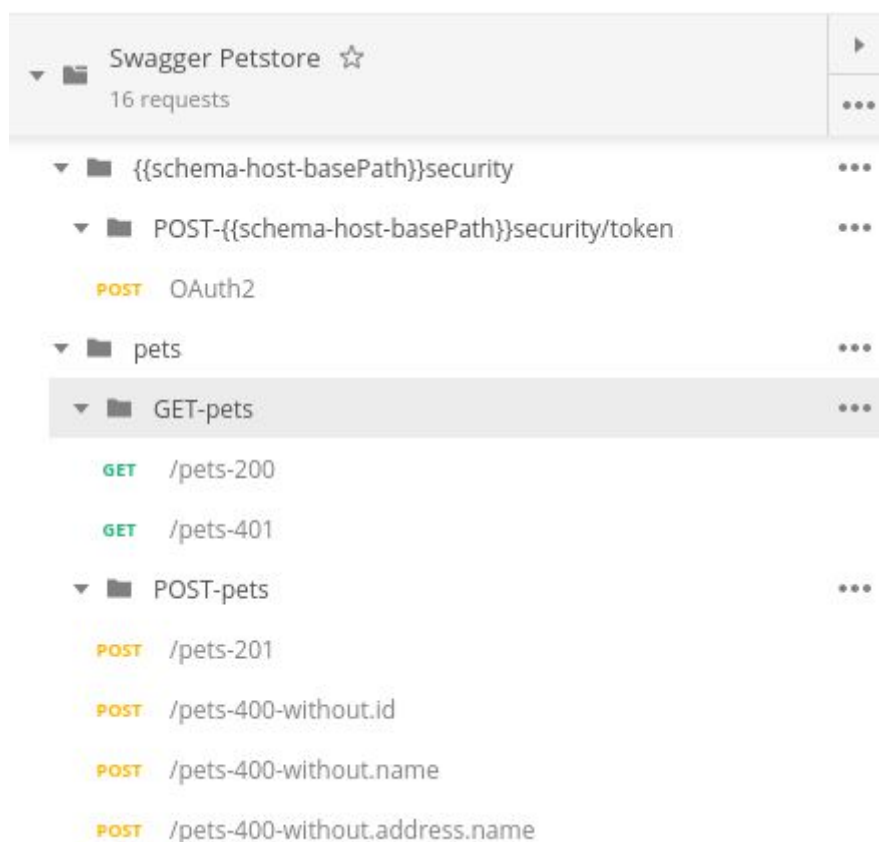
1. Introducción	3
2. Openapi2postman	3
3. Archivo de configuración	8
4. Configuración del entorno de trabajo.	9
5. Ejecución de la herramienta	9

1. Introducción

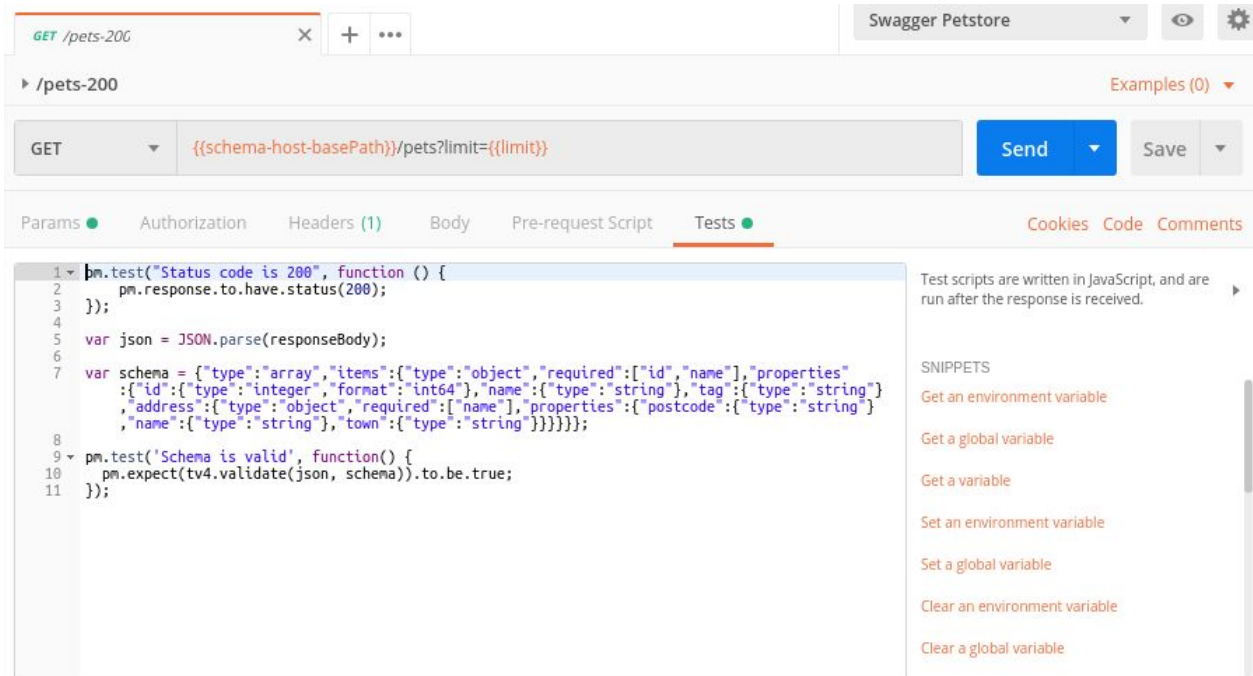
En el presente documento se describe el uso de la herramienta swagger to postman, la cual sirve para generar una serie de colecciones y entornos para ser usados en el conocido cliente de APIs Postman. Dichas colecciones y entornos se basan en la definición en OpenApi 2 (AKA swagger) de una API. Dicha definición se debe aportar para la ejecución de la herramienta, así como un archivo de configuración cuyo uso se describe en el documento anexo.

2. Openapi2postman

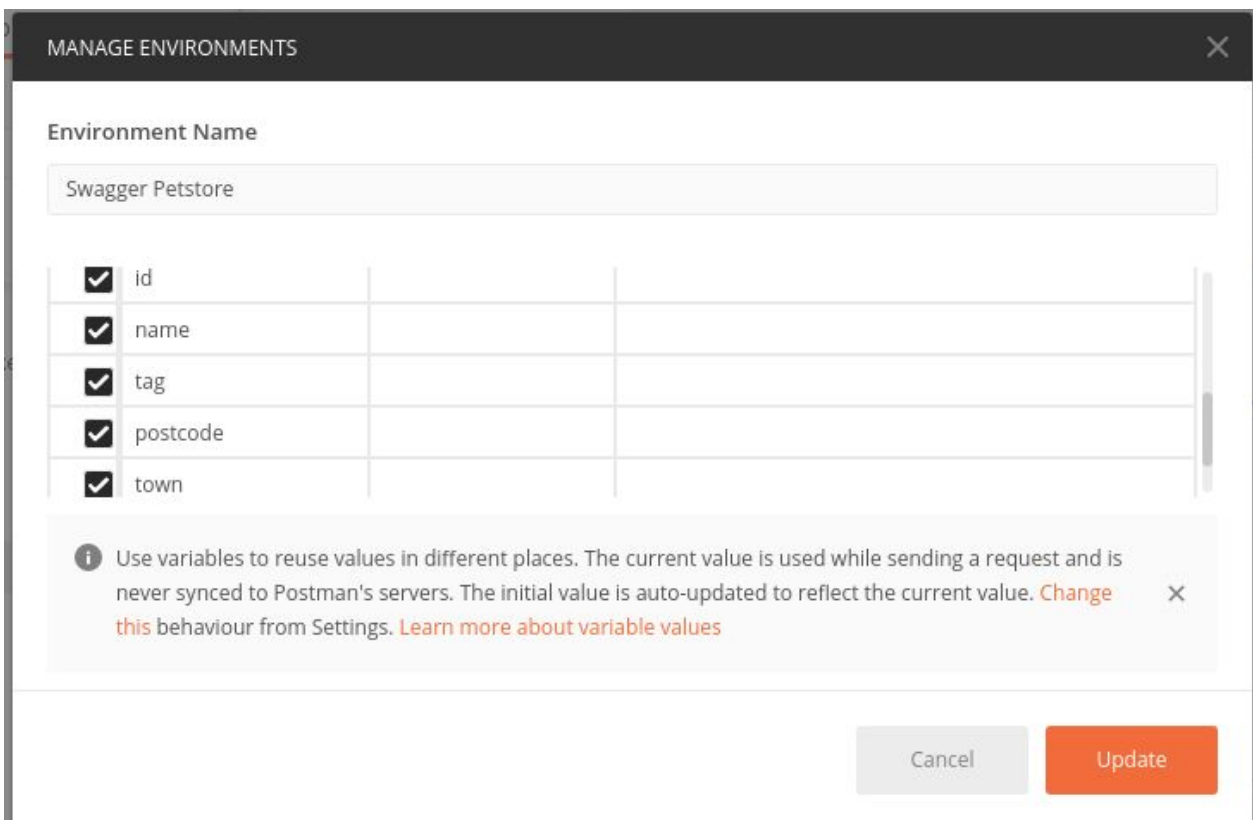
Openapi2postman genera una colección y un entorno Postman a partir de un archivo de definición OpenApi v. 2.0. Dicha colección incluye tests para respuestas con status HTTP 2xx, 400, 401 y 404, en función de las peticiones realizadas a la API. Su utilidad es probar que la implementación de la API se corresponde con la definición de la misma y se comporta tal y como su documentación describe.



En todos los casos se comprueba el status HTTP y se verifica que la respuesta cumple con el esquema definido en el swagger. En las respuestas con status 400 se comprueban los campos obligatorios de la petición, tanto su tipo y formato como su existencia. Las respuestas con status 401 se prueban eliminando la cabecera *Authorization*.



El fichero de entorno generado contiene todas las variables necesarias en los cuerpos y las URLs. Estas variables se usan en la colección Postman generada.



Para las llamadas a endpoints protegidos por esquemas de seguridad HTTP basic o APIKey se generan las variables de entorno que contienen los identificadores.

MANAGE ENVIRONMENTS

Environment Name

Swagger Petstore

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	schema-host-basePa...	http://petstore.swag...	http://petstore.swagger.io/v1			
<input checked="" type="checkbox"/>	petId					
<input checked="" type="checkbox"/>	BasicAuth					

ⓘ Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. [Change this behaviour from Settings.](#) [Learn more about variable values](#)

Cancel
Update

Filter
History
Collections

Trash
1 request

POST OAuth2

Swagger Petstore
16 requests

{{schema-host-basePath}}security

POST-{{schema-host-basePath}}security/token

POST OAuth2

pets

GET-pets

GET /pets-200

GET /pets-401

POST-pets

POST /pets-201

POST /pets-400-without.id

POST /pets-400-without.name

POST /pets-400-without.address.name

GET /pets-200

GET /pets-200

{{schema-host-basePath}}/pets?limit={{limit}}

Send
Save

Params
Authorization
Headers (1)
Body
Pre-request Script
Tests

KEY
VALUE
DESCRIPTION

☒ Authorization
{{BasicAuth}}

Key
Value
Description

Response

Hit the Send button to get a response.

Si se usa una definición de seguridad de tipo Oauth 2 se debe pasar la ruta a un archivo que debe contener una colección postman con las peticiones necesarias para obtener los tokens. El nombre de cada petición será el que se especifique en *securityDefinition* y se creará una variable de entorno con el mismo nombre que contendrá dicho token . Después se añadirá la petición a la colección generada, así como la variable con el token al entorno generado.

The screenshot shows the Postman interface with a collection named 'Swagger Petstore'. The selected request is a POST request to 'POST-{{(schema-host-basePath)}}security/token' under the 'OAuth2' folder. The 'Tests' tab is active, showing a JavaScript test script:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 var json = JSON.parse(responseBody);
6
7 postman.setEnvironmentVariable("OAuth2", "Bearer "+json.data.access_token);
```

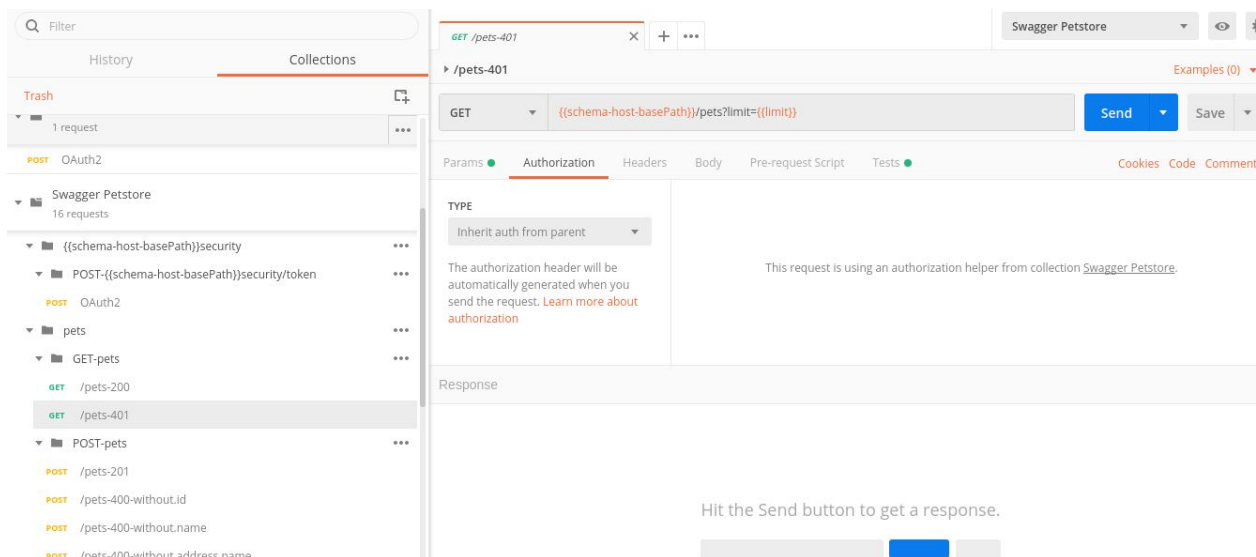
The right sidebar shows a list of snippets for setting and clearing environment and global variables.

The screenshot shows the Postman interface with the same collection. The selected request is a POST request to 'POST-{{(schema-host-basePath)}}pets' under the 'pets' folder. The 'Headers' tab is active, showing the following headers:

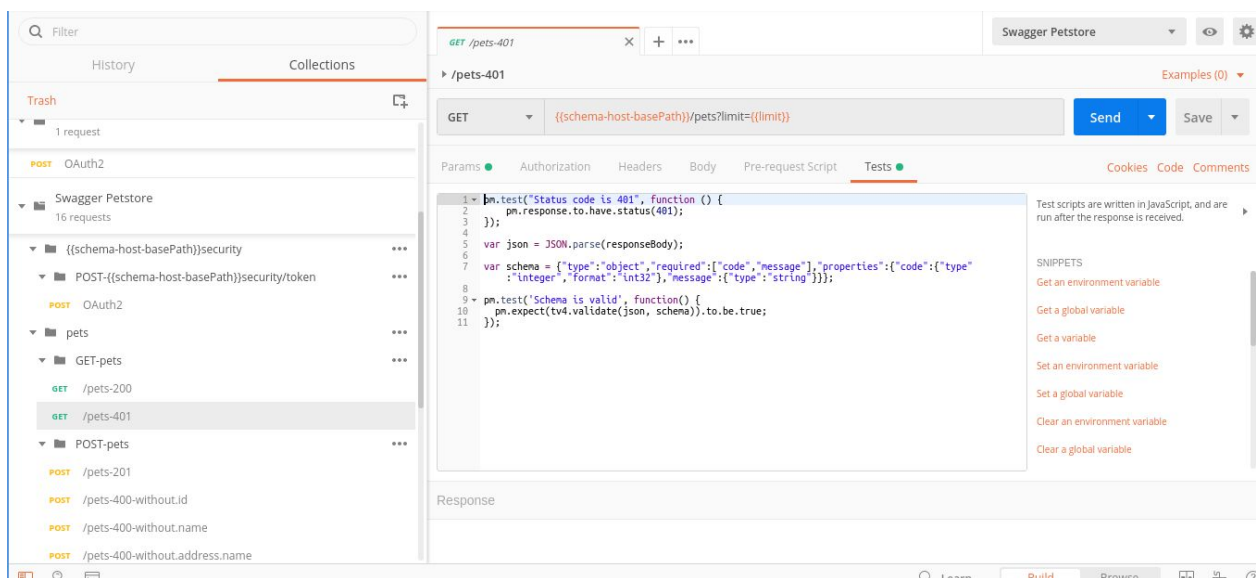
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Authorization	{{OAuth2}}	
Key	Value	Description

The 'Response' tab is active, showing a message: 'Hit the Send button to get a response.' with a blue 'Send' button.

El status 401 se prueba no enviando el token de acceso.



The screenshot shows the Swagger Petstore API client interface. On the left, the 'Collections' tab is active, showing a tree of API endpoints. The 'GET /pets-401' endpoint is selected. The main panel shows the 'Authorization' tab, which is currently empty. The 'Send' button is visible, and a message at the bottom says 'Hit the Send button to get a response.'

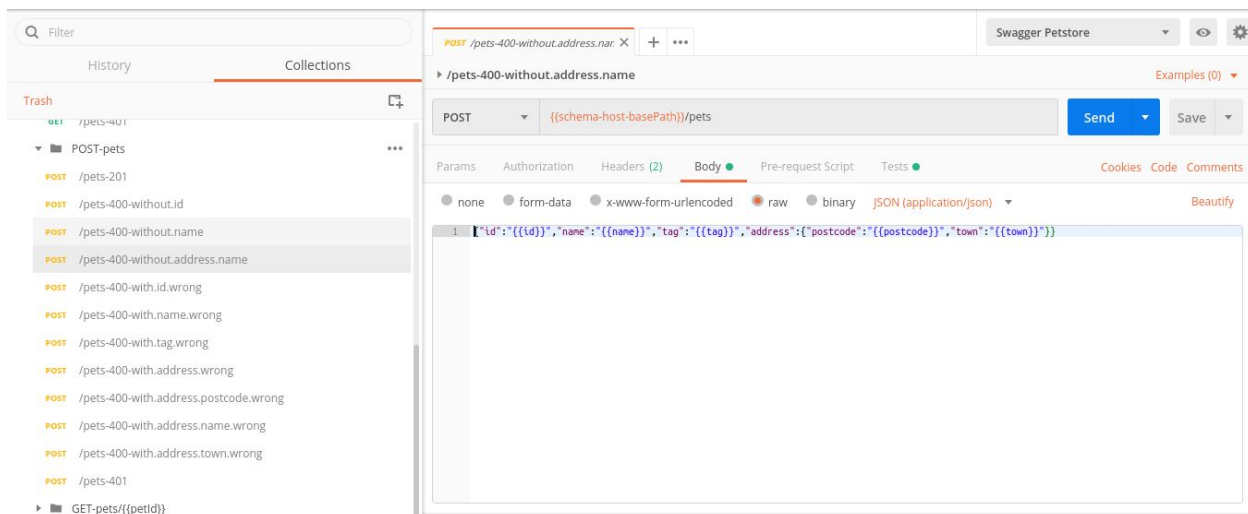


The screenshot shows the Swagger Petstore API client interface. On the left, the 'Collections' tab is active, showing a tree of API endpoints. The 'GET /pets-401' endpoint is selected. The main panel shows the 'Tests' tab, which contains a JavaScript test script. The script is as follows:

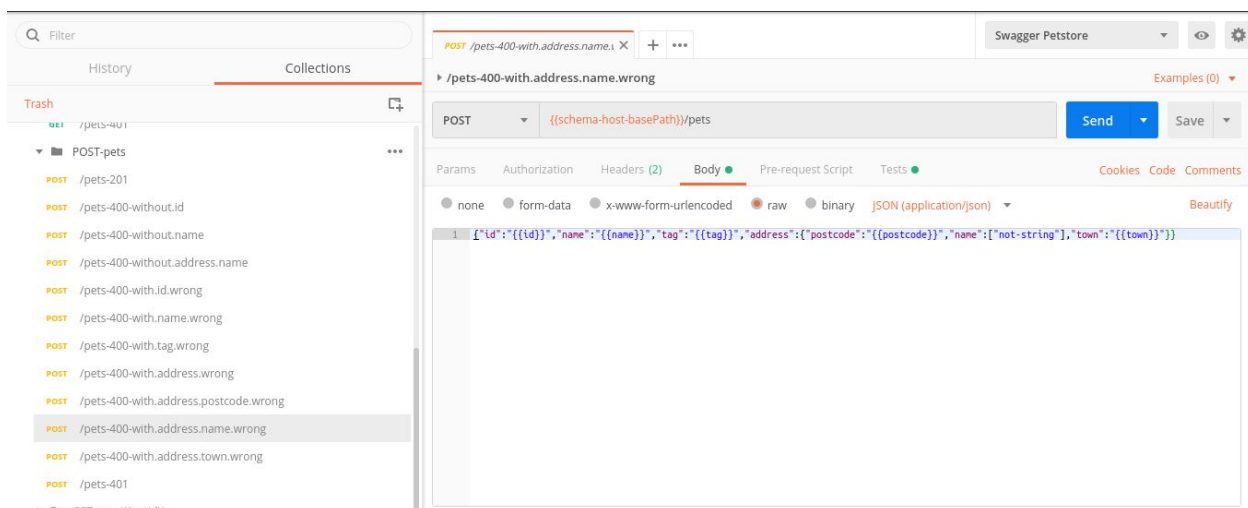
```
1 pm.test("Status code is 401", function () {
2   pm.response.to.have.status(401);
3 });
4
5 var json = JSON.parse(responseBody);
6
7 var schema = {
8   "type": "object",
9   "required": ["code", "message"],
10  "properties": {
11    "code": {
12      "type": "integer",
13      "format": "int32",
14      "message": {
15        "type": "string"
16      }
17    }
18  }
19 };
20 pm.test("Schema is valid", function () {
21   pm.expect(tv4.validate(json, schema)).to.be.true;
22 });
```

The 'Response' tab is also visible, showing a message: 'Test scripts are written in JavaScript, and are run after the response is received.'

El status 400 se prueba no enviando parámetros obligatorios.



También puede probarse enviando parámetros de tipo erróneo (distinto al definido en el swagger).



3. Archivo de configuración

Es posible configurar la ejecución de la herramienta para que genere las colecciones y entornos que se deseen, eligiendo el nombre de los archivos, la ubicación dónde se dejarán los resultados, el tipo de peticiones que se pueden realizar, etc. Dicha configuración se establece de forma individualizada para cada colección y entorno. Puede consultarse la definición y ejemplos de dicho archivo de configuración en el siguiente documento: [Archivo de configuración openapi2postman](#)

En la estructura de archivos presente en el repositorio de código se incluye un archivo de configuración que puede tomarse como ejemplo.

4. Configuración del entorno de trabajo.

Para poder ejecutar la herramienta es necesario tener instalado node.js y su gestor de paquetes, npm. Para instalar ambas herramientas en local podemos descargar el instalador preconfigurado que proporciona el fabricante:

<https://nodejs.org/es/download/>.

5. Ejecución de la herramienta

Los argumentos de ejecución de la herramienta son dos, ambos obligatorios:

- Archivo con la definición de la API acorde a la especificación OpenApi 2.0 en formato yaml.
- Archivo JSON de configuración.

En ambos casos se especificará la ruta local a ambos archivos.

Puede probarse la herramienta con el swagger de ejemplo y la colección de autorizaciones proporcionada. Se ejecutarán desde el directorio en el que se ubica la estructura de ficheros entregada los siguientes comandos:

```
npm install
node index.js --file example/swagger_provincias.yml --configuration
example/s2p_config_file.json
```

Tras la ejecución del comando se generarán los siguientes archivos:

- test_results/Provincias_API_TestSuite_DEV.postman_collection.json: Colección para pruebas de desarrollo sin peticiones de autorización.
- test_results/Provincias_API_TestSuite_VAL.postman_collection.json: Colección para pruebas en validación con todas las peticiones necesarias.
- test_results/Provincias_API_TestSuite_PROD.postman_collection.json: Colección para pruebas en producción sin peticiones de escritura de datos.
- test_results/Provincias_API_TestSuite_DEV.postman_environment.json: Entorno para la colección de desarrollo.
- test_results/Provincias_API_TestSuite_VAL.postman_environment.json: Entorno para la colección de validación.
- test_results/Provincias_API_TestSuite_PROD.postman_environment.json: Entorno para la colección de producción.