



Pruebas de API con Postman



Pruebas de API con Postman

ELDAR ACADEMY

Versión: 1.0

Fecha: 28 de Diciembre de 2024

Objetivo	3
Introducción	3
Conclusiones y recomendaciones	13
Bibliografía	14



Objetivo

El objetivo de esta actividad es que el colaborador aprenda e investigue sobre pruebas básicas con API utilizando la herramienta de Postman, documentando el proceso y explique cómo se realizó.

Introducción

Para comenzar con esta actividad, vamos a desarrollar varias preguntas que serán indispensables saber antes de dar un ejemplo práctico.

¿Qué es una API?

API significa “**Interfaz de programación de aplicaciones**”. Estas son un software que hacen el papel de intermediario entre dos aplicaciones, es decir la comunicación entre esas aplicaciones. Son un “puente” que facilita la interacción entre dos sistemas de forma que estos intercambien datos. Las API más populares y más utilizadas son las “REST” (Transferencia de estado representacional).

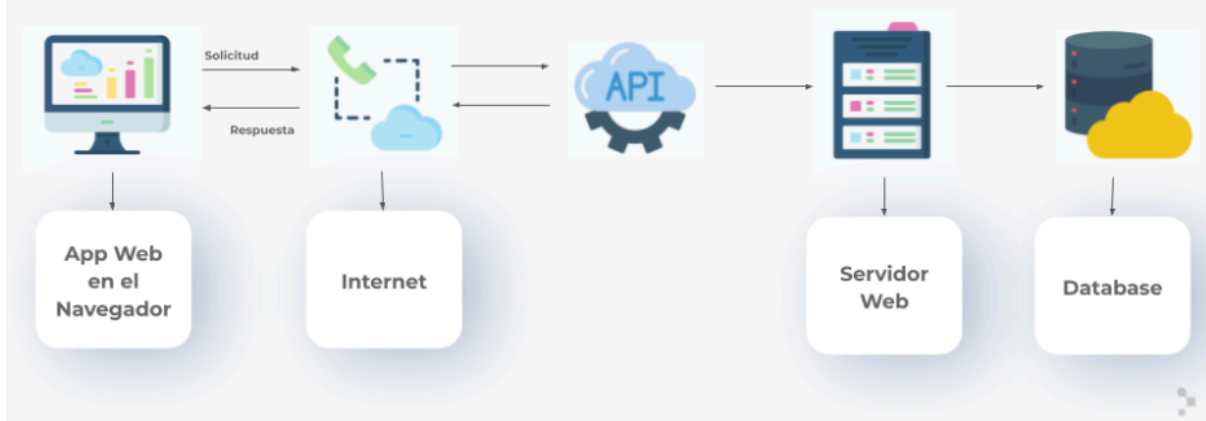
¿Con qué lenguaje se comunican las API?

El lenguaje con que se comunican las API se llama JSON (JavaScript Object Notation). Este es un formato de texto independiente del lenguaje, ideal para el intercambio de datos, siendo fácil de leer y escribir por seres humanos, simple de interpretar y generar para los equipos de cómputos.

¿Cómo interactúa con un servicio web?

La interacción con un servicio web o API web, es a través de los métodos o protocolos “HTTP”, los cuales van a intercambiar información entre cliente y el servidor. Es decir, el servidor va a esperar una petición y si esta contiene el protocolo correcto, entonces devuelve al cliente la información solicitada.

¿Cómo trabaja un API?



La forma en como trabaja una API es mediante un “Contrato”, es decir, va a tener una petición que contiene los headers (envía información), endpoints, body (parámetros). Cuando este servicio recibe la petición, envía un código de respuesta. Estos códigos por ejemplo pueden ser:

- Rango 100: Respuesta informativa.
- Rango 200: Respuesta correcta.
- Rango 300: Redirección.
- Rango 400: Error del cliente.
- Rango 500: Error del servidor.

Los “Verbos” más comunes para realizar peticiones son:

- GET: Permite la obtención de datos del servidor.
- POST: Crea nuevos elementos enviando información al servidor.
- PUT: Actualiza datos del servidor.
- DELETE: Elimina un dato o información del servidor.

¿Qué problema resuelven en QA?



En el ámbito QA las APIs son muy importantes, ya que garantizan que las aplicaciones puedan comunicarse entre sí sin errores, que los datos intercambiados sean correctos y que los servicios sean confiables. Algunos ejemplos de problemas que resuelven:

- Errores de integración: QA garantiza que las APIs puedan integrarse correctamente con otros sistemas.
- Validación de datos: Al intercambiar datos constantemente, las pruebas de QA van a ayudar a asegurar que los datos enviados y recibidos sean consistencia, en el formato esperado y cumplan con los requisitos del negocio/cliente.
- Rendimiento: QA asegura que las APIs sean escalables y respondan adecuadamente bajo diferentes niveles de carga.

¿Qué es Postman?

Es una herramienta que sirve para construir y utilizar APIs, simplifican el ciclo de vida de la creación de APIs mejores y más rápido. Estas se consumen a través de las peticiones HTTP, (esta es la relación que existe entre postman y las peticiones.)

Instalación y configuraciones

Descargar el instalador de Postman para el sistema que se desea. Una vez instalado, se debe crear una cuenta en postman. (Es opcional esto ya que se puede trabajar con Postman por la página web.)

Complete los campos requeridos:



Welcome to Postman! Tell us a bit about yourself.

Your name

Alejandro Rimasa

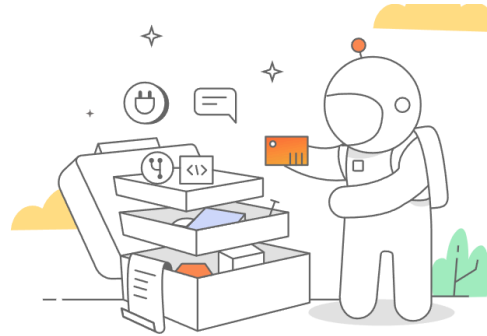
What is your role?

Quality Engineer / Tester

Have you used Postman before?

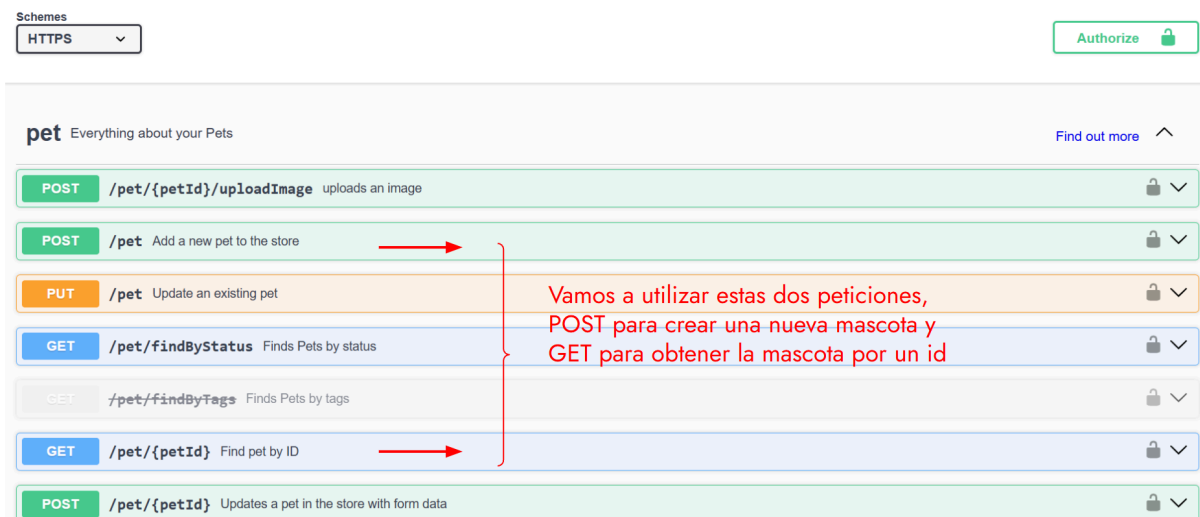
No

Continue



Demostración del uso de Postman: Capturas del proceso de solicitud y respuestas obtenidas. Ejemplo

Para realizar el ejemplo voy a utilizar una página demo de una API para tiendas de mascotas <https://petstore.swagger.io/#/>



Primero vamos a trabajar con la petición de "POST" para ello debemos abrir el desplegable y nos van a aparecer unos parámetros por defecto, el cual edité el "ID" y el "Name". Seguido de esto vamos a seleccionar el botón "Execute".



POST /pet Add a new pet to the store

Parameters Cancel

Name	Description
body * required	Pet object that needs to be added to the store
object (body)	Edit Value Model

```
{
  "id": 10,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "Gala",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Cancel

Parameter content type
application/json

Execute Clear

Al seleccionar el botón “Execute”, se nos abrirá el comando “**Curl**”, la “**url**” de la API, el “**Código de respuesta**” y el “**Response body**” que es la respuesta que nos trae la API.

Responses Response content type application/json

Curl Comando curl: línea de comandos que permite el intercambio de datos entre un dispositivo y un servidor a través de una terminal

```
curl -X 'POST' \
  'https://petstore.swagger.io/v2/pet' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 10,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "Gala",
    "photoUrls": [
      "string"
    ],
    "tags": [
      {
        "id": 0,
        "name": "string"
      }
    ],
    "status": "available"
  }'
```

Request URL Url de la API

https://petstore.swagger.io/v2/pet

Server response Código de respuesta

Code Details

200 Successful response

Response body Response body, es la respuesta que nos regresa la API, es el mismo que utilizamos para la petición

```
{
  "id": 10,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "Gala",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Response headers Response headers permite realizar varias acciones en las cabeceras de solicitud y respuesta HTTP. Estas acciones incluyen recuperar, establecer, agregar y eliminar cabeceras de la lista de cabeceras de la solicitud.

```
access-control-allow-headers: Content-Type,api_key,Authorization
access-control-allow-methods: GET,POST,DELETE,PUT
access-control-allow-origin: *
content-type: application/json
date: Sat, 04 Jan 2025 11:25:43 GMT
server: Jetty(9.2.9.v20150224)
```

Responses

Code	Description
405	Invalid input



Cuando una API se consume desde otra aplicación, visualmente no se pueden detectar lo que ocurre en el intercambio de datos, la forma en que se pueda saber si se están conectando o cuales son los elementos del request que vienen del response es abriendo la consola de las “Devtools” y dirigirse a la solapa de “Network”, algunos de los siguientes elementos permiten entender y depurar cómo se está comunicando la app con la API. En este caso encontraremos en el **“Request Method”: POST** y el **“Status Code”: 200**, lo que significa que la solicitud fue procesada exitosamente por el servidor.

Network Tab Details:

- Name:** pet (Petición pet)
- General:**
 - Request URL: https://petstore.swagger.io/v2/pet
 - Request Method: POST
 - Status Code: 200 OK
 - Remote Address: 54.80.69.226:443
 - Referrer Policy: strict-origin-when-cross-origin
- Response Headers:**
 - Access-Control-Allow-Headers: Content-Type, api_key, Authorization
 - Access-Control-Allow-Methods: GET, POST, DELETE, PUT
 - Access-Control-Allow-Origin: *
 - Content-Type: application/json
 - Date: Sun, 05 Jan 2025 11:51:51 GMT
 - Server: Jetty(9.2.9.v20150224)
- Request Headers:**
 - :authority: petstore.swagger.io
 - :method: POST
 - :path: /v2/pet
 - :scheme: https
 - Accept: application/json
 - Accept-Encoding: gzip, deflate, br, zstd
 - Accept-Language: es-ES,es;q=0.9
 - Cache-Control: no-cache
 - Content-Length: 214
 - Content-Type: application/json

Annotations:

- En esta lista se encuentran las peticiones
- Cabecera general aplican tanto a las peticiones como a las respuestas, pero sin relación con los datos que finalmente se transmiten en el cuerpo. Incluye información básica de la conexión y el contexto de la solicitud.
- Cabecera que contiene más información sobre como el servidor procesó la solicitud y que datos devuelve.
- Cabecera que contiene más información sobre el contenido enviado por el navegador o el cliente servidor



✕ Headers Payload Preview Response Initiator Timing Cookies

▼ Request Payload [view source](#)

```
{id: 10, category: {id: 0, name: "string"}, name: "Gala", photoUrls: ["string"],...}
  ▼ category: {id: 0, name: "string"}
    id: 0
    name: "string"
  id: 10
  name: "Gala"
  ▼ photoUrls: ["string"]
    0: "string"
    status: "available"
  ▼ tags: [{id: 0, name: "string"}]
    ► 0: {id: 0, name: "string"}
```

En el Payload es el contenido del cuerpo de la solicitud, acá encontramos el JSON con los datos que utilizamos para el envío.

✕ Headers Payload Preview Response Initiator Timing Cookies

```
1 {
  -   "id": 10,
  -   "category": {
  -     "id": 0,
  -     "name": "string"
  -   },
  -   "name": "Gala",
  -   "photoUrls": [
  -     "string"
  -   ],
  -   "tags": [
  -     {
  -       "id": 0,
  -       "name": "string"
  -     }
  -   ],
  -   "status": "available"
  - }
```

Response muestra el contenido de la respuesta del servidor como fue recibido, puede ser en texto plano como JSON, HTML, XML, ect.

Seguido de la petición POST, vamos a trabajar con la petición GET esta nos va a servir para obtener información de un recurso, para ello debemos abrir el desplegable y en este caso nos va a pedir el "ID" de la mascota. Seguido de esto vamos a seleccionar el botón "Execute".

GET /pet/{petId} Find pet by ID

Returns a single pet

Parameters [Cancel](#)

Name	Description
petId * required integer(\$int64) (path)	ID of pet to return

10

Execute



La respuesta va a ser el ID y el nombre que habíamos colocado anteriormente.

Responses

Response content type: application/json

Curl

```
curl -X 'GET' \
  'https://petstore.swagger.io/v2/pet/10' \
  -H 'accept: application/json'
```

Request URL

https://petstore.swagger.io/v2/pet/10

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 10, "category": { "id": 10, "name": "string" }, "name": "doggie", "photoUrls": ["string"], "tags": [{ "id": 10, "name": "string" }], "status": "string" }</pre> <p>Response headers</p> <pre>access-control-allow-headers: Content-Type,api_key,Authorization access-control-allow-methods: GET,POST,DELETE,PUT access-control-allow-origin: * content-type: application/json date: Sun,05 Jan 2025 16:29:51 GMT server: Jetty(9.2.9.v20150224)</pre>

Responses

Code	Description
200	successful operation

En nuestra consola del navegador nos va a mostrar la petición, en este caso encontraremos en **“Request Method” : GET** y el **“Status Code” : 200**, lo que significa que la solicitud fue procesada exitosamente por el servidor.

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights

pet

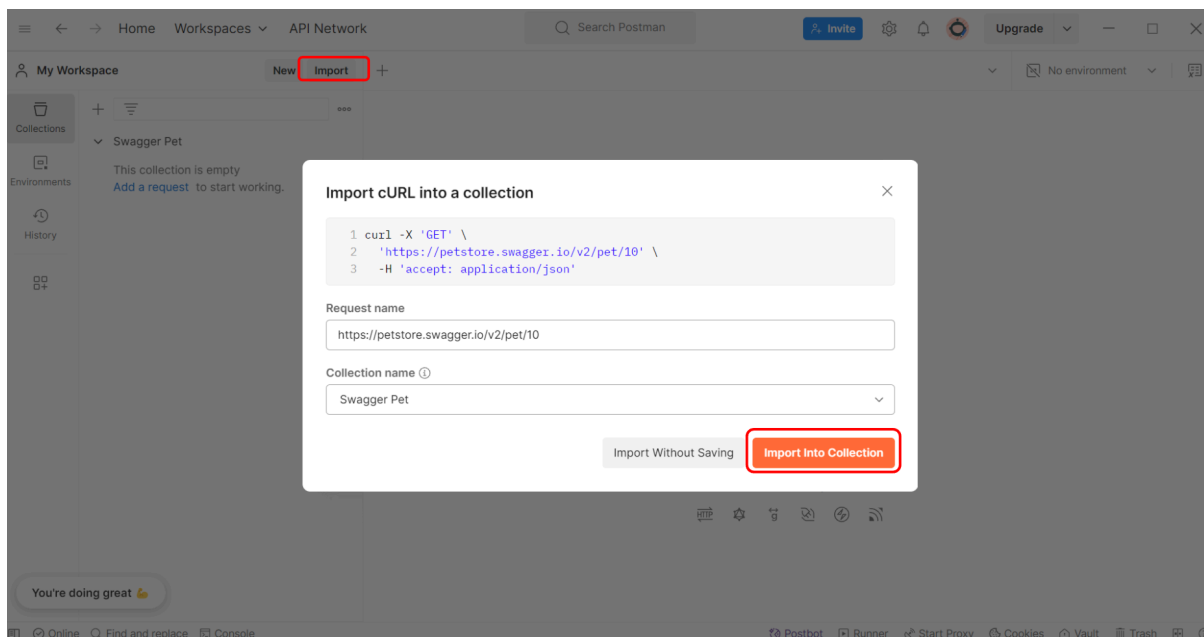
2000000 ms 4000000 ms 6000000 ms 8000000 ms 10000000 ms 12000000 ms 14000000 ms 16000000 ms 18000000 ms

Name	Headers	Preview	Response	Initiator	Timing	Cookies
pet	<p>General</p> <p>Request URL: https://petstore.swagger.io/v2/pet/10</p> <p>Request Method: GET</p> <p>Status Code: 200 OK</p> <p>Remote Address: 44.214.203.224:443</p> <p>Referrer Policy: strict-origin-when-cross-origin</p> <p>Response Headers</p> <p>Access-Control-Allow-Headers: Content-Type, api_key, Authorization</p> <p>Access-Control-Allow-Methods: GET, POST, DELETE, PUT</p> <p>Access-Control-Allow-Origin: *</p> <p>Content-Type: application/json</p> <p>Date: Sun, 05 Jan 2025 16:29:51 GMT</p> <p>Server: Jetty(9.2.9.v20150224)</p> <p>Request Headers</p> <p>:authority: petstore.swagger.io</p> <p>:method: GET</p> <p>:path: /v2/pet/10</p> <p>:scheme: https</p> <p>Accept: application/json</p> <p>Accept-Encoding: gzip, deflate, br, zstd</p> <p>Accept-Language: es-ES, es;q=0.9</p> <p>Cache-Control: no-cache</p>					

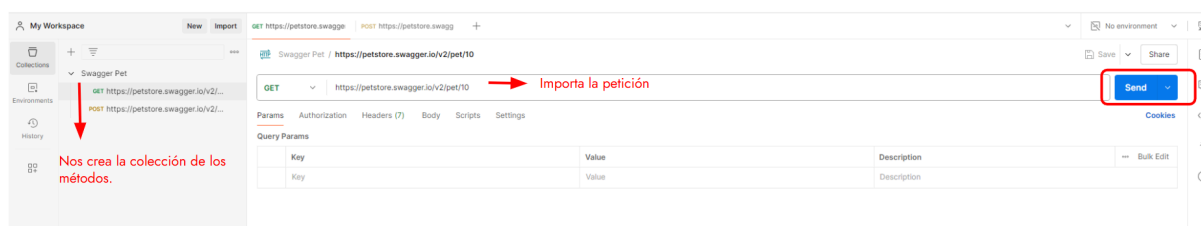


¿Cómo realizar una petición en POSTMAN?

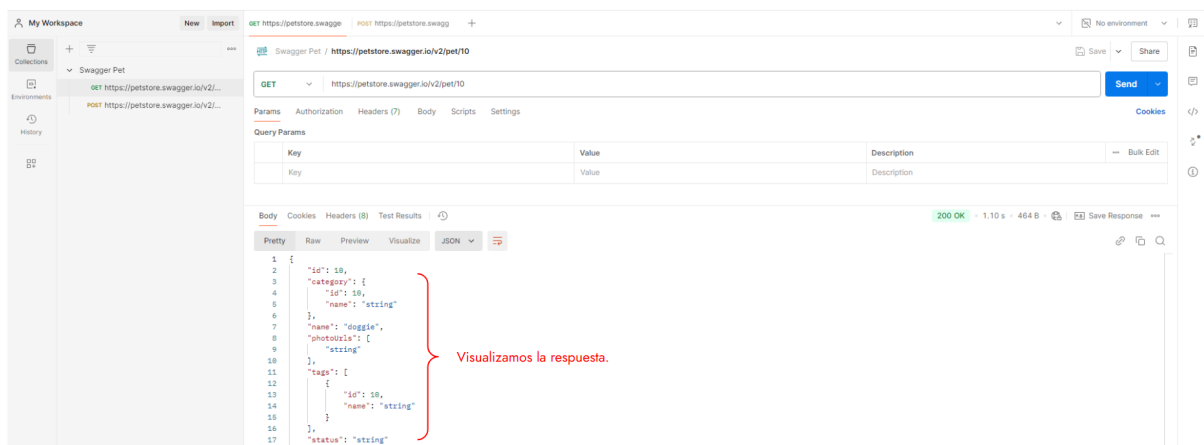
Vamos a copiar el comando **“CURL”**, seguido de esto lo importamos en Postman y se nos abre un pop-up donde pegamos el comando, seleccionamos importar en una colección y automáticamente Postman nos importa esta petición **GET**. Este mismo paso lo vamos a realizar para la petición **POST**.



Postman nos va a crear la colección, seleccionamos **“SEND”**



Al seleccionar **“Send”** nos va a mostrar la respuesta:





Para la petición **“POST”** vamos a realizar los mismos pasos, pero primero debemos dirigirnos a la solapa del **“Body”** para crear una nueva mascota, cambiando el número del **“ID”** y el **“Name”**, seleccionamos **“SEND”** y nos debería de mostrar el código de respuesta 200, este código nos está diciendo que creó la mascota.

The screenshot shows the Swagger Pet API interface. The URL is `https://petstore.swagger.io/v2/pet` and the method is **POST**. The **Body** tab is selected, showing a JSON payload:

```
{  "id": 15,  "category": {    "id": 0,    "name": "string"  },  "name": "Pezoso",  "photoUrls": [    "string"  ],  "tags": [    {      "id": 0,      "name": "string"    }  ],  "status": "available"}
```

. The **Send** button is highlighted. Below the request, the **Test Results** tab shows a **200 OK** response with a status of 200 and a response time of 0.47 s. A red bracket points to the response, with the text "Visualizamos la respuesta."

Si queremos realizar una prueba manual, para saber si esta mascota fue creada, nos vamos a la petición **“GET”** y cambiamos el número de **“ID”** de la importación para obtener ese dato y seguido de esto seleccionamos **“SEND”**.

The screenshot shows the Swagger Pet API interface. The URL is `https://petstore.swagger.io/v2/pet/15` and the method is **GET**. The **Send** button is highlighted. Below the request, the **Test Results** tab shows a **200 OK** response with a status of 200 and a response time of 1.05 s. The response body is shown in the **Body** tab, displaying the pet details:

```
{  "id": 15,  "category": {    "id": 0,    "name": "string"  },  "name": "Pezoso",  "photoUrls": [    "string"  ],  "tags": [    {      "id": 0,      "name": "string"    }  ],  "status": "available"}
```

. A red bracket points to the response, with the text "Respuesta obtenida.".

Below the response, the **Test Results** tab shows a **404 Not Found** response with a status of 404 and a response time of 0.00 s. The response body is shown in the **Body** tab, displaying the error message:

```
{  "code": 1,  "type": "error",  "message": "Pet not found"}
```

. A red bracket points to the response, with the text "Respuesta del mensaje de error".



Conclusiones y recomendaciones

En conclusión, el uso de la herramienta Postman nos permite comprender como funcionan las APIs en la comunicación entre las aplicaciones, garantizando que las solicitudes se procesan correctamente y que los datos intercambiados sean consistentes y confiables, reflejando así una base para el uso de herramientas de pruebas en QA.

Como recomendación se podría seguir investigando las pruebas automatizadas ya que esto sería muy beneficioso aprovechar las capacidades de automatización que tiene Postman, permitiendo así realizar pruebas repetitivas de manera más eficiente y reducir tiempos en validaciones.



Bibliografía

Videos

<https://www.youtube.com/watch?v=PvAsMgbRs5w&t=799s>

<https://www.youtube.com/watch?v=fTtWVCortxo&list=PLDbrnXa6SAzUsLG1gJECgFYLSZDov09fO&index=4> (03. JSON)

<https://www.youtube.com/watch?v=inNR5enqRq0&list=PLDbrnXa6SAzUsLG1gJECgFYLSZDov09fO&index=4> (04. Protocolo HTTP)

<https://www.youtube.com/watch?v=ooRGyFzH8Ww&list=PLDbrnXa6SAzUsLG1gJECgFYLSZDov09fO&index=5> (05. Postman)

Otros

<https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>

<https://www.qatouch.com/blog/api-testing/>

<https://qalified.com/es/blog/api-testing/>

<https://lab.wallarm.com/what/pruebas-api/>

<https://es.abstracta.us/blog/api-testing-guia-practica/>

<https://blog.hubspot.es/website/comando-curl>

<https://developer.mozilla.org/es/docs/Web/API/Headers>

<https://developer.mozilla.org/es/docs/Web/HTTP/Headers>

<https://petstore.swagger.io/#/pet>