

API REST para diagnóstico de red con FastAPI y Postman

Carlos Ramon Anthony Aldair
20211104E

Guerrero Ccompí Jhiens Angel
20210145J

Iman Noriega Melissa
20224041G

Seminario Serna Joel Benjamin
20210056G

Rodriguez Ricra Emhir Carlo Andre
20200483J

Abstract—Postman es una de las herramientas más completas y populares para la automatización y documentación de APIs. Nacida como un complemento para el navegador, se ha convertido en una plataforma fundamental utilizada por millones de desarrolladores y empresas en todo el mundo. En el contexto del crecimiento de la economía digital basada en APIs, Postman permite afrontar los desafíos de desarrollo, prueba y mantenimiento de servicios que se comunican mediante APIs REST. Este trabajo presenta el uso práctico de Postman en la construcción y validación de solicitudes REST, destacando su importancia en la eficiencia del desarrollo y en la mejora de la calidad del software.

I. INTRODUCCIÓN

En la actualidad, el desarrollo de software moderno depende en gran medida de la integración y comunicación entre múltiples servicios mediante interfaces de programación de aplicaciones (APIs). Las API REST se han convertido en el estándar más común para facilitar esta interacción entre sistemas distribuidos.

Contar con herramientas que permitan diseñar, probar, automatizar y documentar APIs de forma eficiente es crucial. Postman surge como una solución integral para enfrentar estos desafíos, brindando a los desarrolladores y equipos de trabajo una plataforma robusta que promueve la colaboración, mejora la calidad del software y acelera los ciclos de desarrollo.

El presente informe describe el uso de Postman en un entorno práctico, explorando cómo se pueden construir, enviar y verificar solicitudes REST de manera sencilla y efectiva. A lo largo del documento, se abordarán los fundamentos de las API REST, el uso de Postman en el desarrollo y prueba de solicitudes, y los beneficios observados en su aplicación dentro del proceso de desarrollo.

II. API REST

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que permite la comunicación entre aplicaciones de software. Las APIs funcionan como intermediarios que permiten que distintos sistemas intercambien información de manera estructurada. Existen diversos tipos de APIs, tales como APIs de sistema operativo, bibliotecas de software, APIs de hardware y APIs web.

Las APIs web son interfaces accesibles mediante protocolos como HTTP o HTTPS, y permiten la comunicación entre aplicaciones a través de internet. Son fundamentales para el desarrollo de servicios en la nube, aplicaciones móviles, sitios web dinámicos y microservicios. Su popularidad ha crecido debido a la necesidad de integrar servicios heterogéneos, como pagos en línea, autenticación con redes sociales o acceso a bases de datos remotas.

A. Arquitectura REST (Representational State Transfer)

REST es un estilo de arquitectura para el diseño de servicios web, propuesto por Roy Fielding en el año 2000. Se basa en principios como la separación cliente-servidor, la ausencia de estado (stateless), el uso uniforme de recursos a través de URIs, y operaciones definidas por el protocolo HTTP: GET, POST, PUT, DELETE, entre otros. Una API RESTful sigue estos principios, permitiendo que los clientes interactúen con los recursos del servidor usando representaciones como JSON o XML.

B. Estructura de una API REST

Una API REST típica se organiza en torno a los recursos, los cuales se identifican mediante URLs. Cada recurso puede ser manipulado mediante los métodos HTTP mencionados. Por ejemplo:

- GET /usuarios: obtiene todos los usuarios.
- POST /usuarios: crea un nuevo usuario.
- GET /usuarios/5: obtiene al usuario con ID 5.
- PUT /usuarios/5: actualiza al usuario con ID 5.
- DELETE /usuarios/5: elimina al usuario con ID 5.

C. Postman como herramienta para pruebas de APIs REST

Postman es una plataforma para desarrollo y pruebas de APIs. Permite enviar solicitudes HTTP de forma interactiva, visualizar respuestas, guardar colecciones de pruebas, automatizar procesos y documentar endpoints. Gracias a su interfaz gráfica, Postman es ideal para desarrolladores que desean probar rápidamente APIs REST, sin necesidad de escribir código adicional. Además, Postman permite la integración con scripts en JavaScript, pruebas automatizadas, generación de documentación y colaboración en equipo.

La mayoría de las APIs REST utilizan JSON (JavaScript Object Notation) como formato para representar datos. JSON es ligero, fácil de leer para humanos y sencillo de procesar por máquinas. Un ejemplo básico de respuesta JSON podría ser:

```
{
  "id": 5,
  "nombre": "Carlos",
  "email": "carlos@gmail.com"
}
```

Postman permite enviar datos en formato JSON y visualizar las respuestas de forma estructurada, lo que mejora la experiencia de prueba y depuración.

D. Ventajas de trabajar con APIs REST y Postman

- Escalabilidad: REST permite escalar aplicaciones fácilmente mediante la separación de cliente y servidor.
- Simplicidad: Al basarse en HTTP, es fácil de entender y adoptar.
- Portabilidad: Puede ser utilizada en distintos lenguajes de programación.
- Pruebas efectivas: Postman facilita las pruebas sin necesidad de programación adicional.
- Automatización y documentación: La combinación de Postman y REST permite automatizar pruebas y generar documentación clara para equipos de desarrollo.

III. POSTMAN

A. Uso de Postman para la Interacción y Validación de APIs REST

El desarrollo de aplicaciones modernas basadas en arquitecturas cliente-servidor, especialmente aquellas que implementan interfaces de programación de aplicaciones (APIs) siguiendo el paradigma REST (*Representational State Transfer*), exige herramientas que faciliten la construcción, prueba, automatización y documentación de dichos servicios. En este contexto, **Postman** se posiciona como una de las herramientas más robustas y ampliamente adoptadas por la comunidad de desarrolladores.

Postman es una aplicación multiplataforma diseñada específicamente para enviar y analizar solicitudes HTTP(S). Su interfaz gráfica permite a los usuarios interactuar con endpoints de una API REST sin necesidad de implementar una interfaz de usuario, lo que resulta esencial durante las fases de diseño, prueba y depuración. Su adopción en proyectos CRUD (*Create, Read, Update, Delete*) facilita la verificación precisa del comportamiento de cada operación definida por el servidor.

1) *Funcionalidad principal*: Postman permite construir solicitudes HTTP utilizando los métodos estándar: GET, POST, PUT, DELETE, entre otros. Cada una de estas solicitudes puede incluir parámetros, encabezados personalizados, cuerpos en formato JSON, autenticación por tokens y otros elementos necesarios para replicar el comportamiento de un cliente real. Asimismo, ofrece la capacidad de visualizar las respuestas del servidor de manera detallada, incluyendo códigos de estado (por ejemplo, 200 OK, 201 Created,

400 Bad Request, 404 Not Found, 500 Internal Server Error), tiempos de respuesta, estructura del cuerpo y encabezados de retorno.

2) *Aplicación práctica en un entorno CRUD*: Durante la implementación de una API REST para un sistema CRUD, Postman permite:

- Verificar la creación de recursos mediante solicitudes POST, enviando datos estructurados en JSON.
- Consultar datos existentes a través de GET, especificando parámetros de búsqueda o paginación.
- Actualizar registros utilizando PUT o PATCH, modificando información específica por ID.
- Eliminar entradas mediante solicitudes DELETE, comprobando respuestas apropiadas (204 No Content).

Estas operaciones pueden almacenarse dentro de *colecciones*, agrupando los endpoints por módulos o entidades, lo que facilita su reutilización y documentación. Además, Postman permite definir *variables de entorno* para gestionar rutas base (`base_url`), tokens de autenticación o identificadores dinámicos durante las pruebas.

3) *Automatización y documentación*: Una característica destacada de Postman es su sistema de *tests automatizados*, que permite agregar scripts escritos en JavaScript para validar automáticamente condiciones específicas de la respuesta, tales como:

```
pm.test("El código es 200", function() {
  pm.response.to.have.status(200);
});
```

De esta manera, los equipos de desarrollo pueden construir suites de pruebas que aseguren la estabilidad de la API durante su evolución. Asimismo, Postman ofrece la opción de generar documentación dinámica a partir de las colecciones, la cual puede ser exportada o publicada de forma colaborativa.

4) *Ventajas dentro del flujo de desarrollo*: El uso de Postman en un flujo de trabajo basado en API REST presenta múltiples beneficios:

- Reducción del tiempo de depuración gracias a la visualización clara de errores.
- Mejora de la colaboración entre desarrolladores backend y frontend al establecer contratos de API claros.
- Pruebas independientes de la interfaz final, permitiendo validaciones tempranas del backend.
- Escalabilidad en pruebas mediante la integración con herramientas como Newman o plataformas CI/CD.

IV. METODOLOGÍA E IMPLEMENTACIÓN

Ahora describiremos el proceso seguido para desarrollar y probar una API REST con FastAPI, así como la utilización de Postman como herramienta principal de consumo, documentación y validación de endpoints.

A. Desarrollo de la API REST

La API se desarrolló utilizando el framework **FastAPI**, debido a su rendimiento, facilidad de uso y soporte para documentación automática con Swagger. Se implementaron

endpoints para verificar la conectividad de red mediante comandos `ping` y `traceroute`, permitiendo probar tanto disponibilidad como latencia entre nodos.

Los endpoints principales de la API son:

- GET `/health` – Verifica que el servicio esté funcionando correctamente.
- GET `/ping` – Ejecuta un ping a un host específico.
- GET `/traceroute` – Realiza traceroute para analizar la ruta de red hacia un host.
- GET `/allowed-hosts` – Lista los dominios permitidos para pruebas.
- POST `/ping/bulk` – Ejecuta ping en paralelo a varios hosts.

El proyecto sigue una arquitectura modular, con separación de lógica en archivos como `main.py`, `utils.py` y `models.py`. Para su ejecución se utilizó el siguiente comando:

```
uvicorn main:app --reload --port 8000
```

B. Desarrollo de las peticiones con Postman

Postman fue empleado como entorno de pruebas RESTful para construir, ejecutar y validar solicitudes HTTP hacia la API. Se creó una colección llamada `Connectivity API`, la cual contiene las siguientes peticiones configuradas:

1. Health Check

Verifica si la API está activa y responde correctamente.

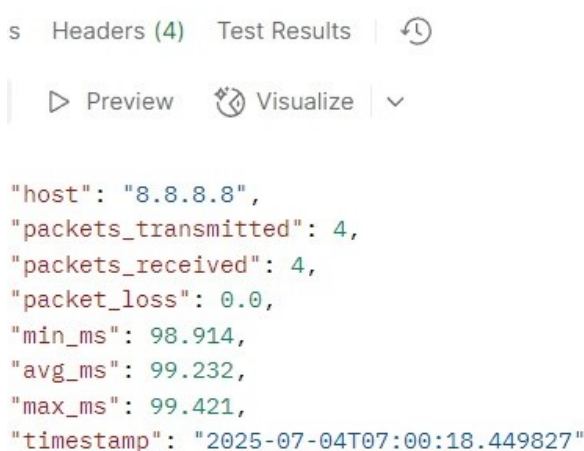


```
"status": "healthy",
"timestamp": "2025-07-04T06:59:38.753620",
"version": "1.0.0"
```

Fig. 1. Petición GET `/health` realizada en Postman

2. Ping Test

Envía un ping al host especificado (por ejemplo, `8.8.8.8`) y devuelve el resultado con la latencia promedio, paquetes enviados, recibidos y perdidos.




```
"host": "8.8.8.8",
"packets_transmitted": 4,
"packets_received": 4,
"packet_loss": 0.0,
"min_ms": 98.914,
"avg_ms": 99.232,
"max_ms": 99.421,
"timestamp": "2025-07-04T07:00:18.449827"
```

Fig. 2. Petición GET `/ping` con el host `8.8.8.8`

3. Traceroute Test

Realiza un traceroute al host indicado, devolviendo los saltos por los que pasa el paquete hasta alcanzar el destino.



```
"host": "google.com",
"hops": [
  {
    "hop": 1,
    "host": "172.27.176.1",
    "rtt_ms": 0.644
  },
  {
    "hop": 2,
    "host": "10.2.0.1",
    "rtt_ms": 129.186
  },
  {
    "hop": 3,
    "host": "149.88.18.252",
    "rtt_ms": 129.149
  },
  {
    "hop": 4,
    "host": "79.127.195.129",
    "rtt_ms": 129.919
  },
  {
    "hop": 5,
    "host": "72.14.210.176",
    "rtt_ms": 129.433
  }
]
```

Fig. 3. Petición GET `/traceroute` con el host `google.com`

4. Allowed Hosts

Muestra una lista de hosts permitidos por la aplicación para realizar pruebas de conectividad. Esto permite controlar el acceso a dominios específicos.

```

"allowed_hosts": [
    "1.0.0.1",
    "1.1.1.1",
    "8.8.4.4",
    "8.8.8.8",
    "cloudflare.com",
    "fastapi.tiangolo.com",
    "github.com",
    "google.com",
    "python.org",
    "stackoverflow.com"
],
"count": 10,
"timestamp": "2025-07-04T07:02:02.450842"

```

Fig. 4. Petición GET /allowed-hosts

5. Bulk Ping Test

Permite enviar múltiples pings a distintos hosts al mismo tiempo, retornando los resultados de cada uno en paralelo.

```

"results": [
  {
    "host": "8.8.8.8",
    "packets_transmitted": 4,
    "packets_received": 4,
    "packet_loss": 0.0,
    "min_ms": 98.713,
    "avg_ms": 98.748,
    "max_ms": 98.812,
    "timestamp": "2025-07-04T07:02:36.162154"
  },
  {
    "host": "1.1.1.1",
    "packets_transmitted": 4,
    "packets_received": 4,
    "packet_loss": 0.0,
    "min_ms": 105.5,
    "avg_ms": 105.779,
    "max_ms": 106.444,
    "timestamp": "2025-07-04T07:02:36.166508"
  }
],
"timestamp": "2025-07-04T07:02:36.166778"

```

Fig. 5. Petición POST /ping/bulk con múltiples hosts

Cada solicitud incluye pruebas automáticas con scripts de validación usando la API de Postman (pm), lo cual permite verificar la estructura de respuesta, los códigos de estado y condiciones como la pérdida de paquetes o el tiempo de respuesta promedio.

C. Pruebas y validación

Las pruebas se realizaron de dos maneras complementarias:

- 1) **Manual e interactiva con Postman**, verificando cada endpoint con diferentes parámetros de entrada y analizando la respuesta en la interfaz gráfica.
- 2) **Automatizada con pytest**, utilizando scripts en la carpeta `tests/`, los cuales validan el comportamiento esperado de cada endpoint. Estas pruebas se ejecutan con el siguiente comando:

```
pytest tests/test_ping.py
```

D. Herramientas utilizadas

- **FastAPI**: framework backend para construir la API REST.
- **Postman**: herramienta para consumir y validar endpoints REST.
- **Uvicorn**: servidor ASGI para correr la aplicación.
- **Pytest**: framework para pruebas automatizadas.
- **Ping/Traceroute**: comandos del sistema integrados en la lógica para diagnóstico de red.

V. CONCLUSIONES

El desarrollo de esta práctica permitió evidenciar la importancia de contar con herramientas como Postman para el diseño, prueba y documentación de servicios RESTful. A través de la implementación de una API de conectividad con FastAPI, se logró construir una solución funcional, modular y fácil de consumir desde clientes externos.

Postman se consolidó como una herramienta esencial durante todo el proceso, facilitando la ejecución de pruebas tanto manuales como automatizadas. Su capacidad para organizar colecciones, crear scripts de validación y visualizar respuestas permitió detectar errores, validar parámetros y asegurar el correcto comportamiento de cada endpoint.

Además, la experiencia de integrar comandos del sistema como `ping` y `traceroute` en una API accesible vía HTTP demostró cómo es posible abstraer funciones del sistema operativo y ofrecerlas como servicios reutilizables en redes más amplias. Esta experiencia reafirma la utilidad de los estándares REST y de herramientas como Postman dentro del ciclo de vida del desarrollo backend moderno, especialmente en escenarios donde la validación de servicios y la colaboración en equipos son fundamentales.

REFERENCES

- [1] Postman, "Postman Learning Center," [Online]. Available: <https://learning.postman.com/>. [Accessed: 5-Jul-2025].
- [2] Postman, "API Testing," [Online]. Available: <https://www.postman.com/api-platform/api-testing/>. [Accessed: 5-Jul-2025].
- [3] Postman Blog, "Why Postman is the Most Popular API Platform," 2022. [Online]. Available: <https://blog.postman.com/why-postman-is-the-most-popular-api-platform/>. [Accessed: 5-Jul-2025].
- [4] Mozilla Developer Network, "Introduction to APIs," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction. [Accessed: 5-Jul-2025].
- [5] Red Hat, "What is a REST API?," [Online]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. [Accessed: 5-Jul-2025].

- [6] L. Richardson, M. Amundsen, and S. Ruby, *RESTful Web APIs*. Sebastopol, CA: O'Reilly Media, 2013.
- [7] Postman, "2023 State of the API Report," [Online]. Available: <https://www.postman.com/state-of-api/>. [Accessed: 5-Jul-2025].