Uso de Postman para Pruebas de APIs en Verificación y Validación de Software

William Alvarado, Max Carrion, Johana Cruz, Cristian Trávez

Resumen

En el desarrollo de software moderno, las APIs son fundamentales para la comunicación entre aplicaciones y servicios. Este informe explora el uso de Postman como herramienta esencial para probar y validar APIs, con un enfoque en las prácticas de testing relacionadas con la verificación y validación de software. Se abordan desde la realización de solicitudes básicas hasta la automatización de pruebas y la integración con herramientas de CI/CD. Se incluyen ejemplos prácticos y recomendaciones para optimizar su uso en el contexto de la calidad del software.

1. Introducción

Las APIs (Interfaces de Programación de Aplicaciones) permiten la interacción entre diferentes sistemas, siendo cruciales en arquitecturas modernas como microservicios y aplicaciones móviles. Garantizar su correcto funcionamiento es vital para la integridad y eficiencia de los sistemas. La verificación y validación de software son procesos esenciales que aseguran que el software cumple con los requisitos y funciona según lo esperado. En este contexto, Postman se ha consolidado como una herramienta líder para desarrollar, probar y documentar APIs de manera eficiente.

1.1. Objetivos del Informe

Este informe tiene como objetivos:

- Explorar las funcionalidades clave de Postman en pruebas de APIs.
- Demostrar cómo realizar solicitudes HTTP y gestionar colecciones.
- Profundizar en las técnicas de testing y automatización en Postman.
- Ilustrar la creación y uso de scripts para validación de respuestas.
- Analizar la integración con herramientas de CI/CD en el proceso de verificación y validación.

2. ¿Qué es Postman?

Postman es una plataforma colaborativa que simplifica el ciclo de vida de las APIs, permitiendo construir, probar, documentar y compartir APIs de forma eficiente. Proporciona una interfaz intuitiva para enviar solicitudes HTTP y analizar respuestas, facilitando la detección de errores y la optimización de las APIs.

3. Instalación y Configuración

Para comenzar, descargue Postman desde `https://www.postman.com/downloads/` e instálelo según su sistema operativo. Al iniciar la aplicación, puede crear una cuenta para sincronizar sus colecciones y configuraciones.

4. Descarga y Configuración del Proyecto

Para realizar prácticas y ejemplos reales, utilizaremos el repositorio *PostmanAPI_Practice* disponible en GitHub. Este proyecto proporciona una API sencilla para gestionar películas, que utilizaremos para nuestras pruebas con Postman.

Para obtener el código del proyecto, clone el repositorio usando Git:

Listing 1: Clonación del repositorio

```
git clone https://github.com/MaxCar31/PostmanAPI_Practice.git
```

5. Realizando Solicitudes HTTP

5.1. Nuestro Primer Request

Una vez instalado, puede crear una nueva solicitud seleccionando el método HTTP adecuado (GET, POST, etc.) y proporcionando la URL del endpoint. Por ejemplo, para realizar una solicitud GET:

- Seleccione el método **GET**.
- Ingrese la URL: `http://localhost:3000/api/movies`.
- Haga clic en **Send** y revise la respuesta.

5.2. Creación de Colecciones

Las colecciones permiten organizar las solicitudes:

- Cree una nueva colección llamada *API de Películas*.
- Guarde las solicitudes dentro de esta colección para mantenerlas organizadas.

5.3. Uso de Variables

Las variables parametrizan las solicitudes y facilitan su reutilización:

- Defina una variable `{{base_url}}` con el valor `http://localhost:3000`.
- Use la variable en las solicitudes: `{{base_url}}/api/movies`.

6. Testing y Automatización en Postman

La verificación y validación de software implican asegurar que el software cumple con los requisitos especificados y funciona correctamente. Postman proporciona herramientas robustas para realizar pruebas automatizadas y validar las respuestas de las APIs.

6.1. Escritura de Tests

En la pestaña **Tests** de cada solicitud, puede escribir scripts en JavaScript para validar las respuestas. Esto es fundamental para verificar que la API responde según lo esperado.

```
pm.test("El c digo de estado es 200", function () {
    pm.response.to.have.status(200);
});
```

Listing 2: Prueba de código de estado

6.2. Validación de Estructura y Contenido de Respuestas

Es importante verificar no solo el código de estado, sino también la estructura y contenido de la respuesta.

```
pm.test("La respuesta contiene una lista de pel culas", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.be.an('array');
});
```

Listing 3: Validación del cuerpo de la respuesta

6.3. Uso de Snippets

Los *snippets* facilitan la creación de pruebas comunes. Puede insertarlos y modificarlos según sus necesidades, lo que agiliza el proceso de escritura de tests.

6.4. Variables Globales y de Entorno

Para pruebas más complejas, puede utilizar variables para almacenar datos de respuestas y utilizarlos en solicitudes subsecuentes.

```
var jsonData = pm.response.json();
pm.environment.set("movieId", jsonData.id);
```

Listing 4: Almacenamiento de datos en variables

6.5. Pruebas de Creación, Actualización y Eliminación

Creación de una nueva película (POST):

```
pm.test("Pel cula creada correctamente", function () {
    pm.response.to.have.status(201);
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
    pm.environment.set("movieId", jsonData.id);
});
```

Listing 5: Prueba de creación

Actualización de una película (PUT):

```
pm.test("Pel cula actualizada correctamente", function () {
    pm.response.to.have.status(200);
    var jsonData = pm.response.json();
    pm.expect(jsonData.title).to.eql("T tulo Actualizado");
});
```

Listing 6: Prueba de actualización

Eliminación de una película (DELETE):

```
pm.test("Pel cula eliminada correctamente", function () {
    pm.response.to.have.status(204);
});
```

Listing 7: Prueba de eliminación

6.6. Collection Runner para Automatización

El *Collection Runner* permite ejecutar todas las solicitudes y pruebas de una colección de forma automatizada, lo que es esencial para pruebas de regresión y validación continua.

- Inicie el *Collection Runner* desde la interfaz principal.
- Seleccione la colección y el entorno adecuados.
- Configure el número de iteraciones y datos de entrada si es necesario.
- Ejecute las pruebas y revise los resultados detallados.

6.7. Gestión de Datos de Prueba

Puede utilizar archivos CSV o JSON para proporcionar datos de prueba al *Collection Runner*, lo que permite probar la API con diferentes conjuntos de datos y escenarios.

6.8. Consola de Postman para Depuración

La consola de Postman es una herramienta útil para depurar solicitudes y scripts de pruebas.

- Acceda a la consola desde el menú **View >Show Postman Console**.
- Use `console.log()` en sus scripts para imprimir mensajes y valores de variables.

7. Integración con CI/CD

La integración de Postman con herramientas de Integración Continua y Despliegue Continuo (CI/CD) permite incorporar las pruebas de APIs en los pipelines de desarrollo, asegurando que las nuevas implementaciones no introduzcan regresiones.

7.1. Uso de Newman

Newman es la herramienta de línea de comandos de Postman que permite ejecutar colecciones en entornos de CI/CD.

- Instale Newman globalmente: `npm install -g newman`
- Ejecute una colección: `newman run coleccion.json`

7.2. Integración con Jenkins

En un pipeline de Jenkins, puede agregar un paso para ejecutar las pruebas de Postman utilizando Newman, y configurar el trabajo para que falle si alguna prueba no pasa.

7.3. Beneficios en Verificación y Validación

La integración de pruebas automatizadas en CI/CD permite:

- Detectar errores de forma temprana en el ciclo de desarrollo.
- Asegurar la consistencia y calidad del software en cada despliegue.
- Reducir el tiempo y esfuerzo en pruebas manuales repetitivas.

8. Monitoreo de APIs

Los monitores permiten programar la ejecución de colecciones para verificar la disponibilidad y rendimiento de las APIs en producción.

8.1. Configuración de Monitores

- Configure un monitor desde la colección en Postman.
- Establezca la frecuencia de ejecución y las notificaciones.
- Revise los reportes para identificar problemas de disponibilidad o rendimiento.

8.2. Limitaciones y Consideraciones

- Los monitores solo pueden acceder a APIs públicas.
- Es importante manejar adecuadamente las credenciales y datos sensibles.

9. Conclusiones

Postman es una herramienta versátil y poderosa que facilita la verificación y validación de software a través de pruebas exhaustivas de APIs. Su capacidad para automatizar pruebas, gestionar datos y variables, y su integración con herramientas de CI/CD, la convierten en un componente esencial en el aseguramiento de la calidad del software.

Referencias

[1] Postman. *API Testing*. Disponible en: `https://www.postman.com/api-testing`

[2] Newman. *Newman Documentation*. Disponible en: `https://www.npmjs.com/package/newman`

[3] Postman. *Continuous Integration with Postman*. Disponible en: `https://learning.postman.com/docs/integrations/continuous-integration/`

[4] Video Tutorial en YouTube. *Postman Tutorial*. Disponible en: `https://www.youtube.com/watch?v=video_id`