API TESTING

Funtional Test and Performance

Descripción

Se ejecutan una serie de pruebas Performance y funcionales sobre una suite de APIs

Andres Velez
V Partners

Contenido

API Testing

La tienda en línea "Your Store" está desarrollando una nueva versión de su sitio web.Para garantizar que las funcionalidades críticas del sistema estén listas antes de que el front-end esté completamente implementado, el equipo de calidad ha decidido validar estas funcionalidades a través de pruebas directas a la API. Además, se implementará automatización para asegurar una ejecución eficiente y repetible de los casos de prueba.

La tienda ha proporcionado la documentación de la API para facilitar la creación de pruebas: https://fakestoreapi.com/docs

Requerimientos de la prueba

Como administrador de "Your Store", quiero poder realizar diferentes requisitos a través de la API para gestionar los productos de forma eficiente.

Validación funcional

1. Consulta todos los productos que pertenezcan a la categoría de “electronics”.
2. Consultar los datos de un producto en específico (Puede ser cualquiera).
3. Crear un producto (con cualquier detalle).
4. Actualiza la imagen del producto que creaste.
5. Simula 150 usuarios concurrentes realizando solicitudes durante un período de 2 minutos a los endpoints de listar todos los productos y de agregar un nuevo producto.
6. Escala el número de usuarios concurrentes desde 100 hasta 1000 en intervalos de 150.

Herramienta para el testing

Para la ejecución de este requerimiento se elige la herramienta K6 por los siguientes motivos:

- Ligero y rápido
- Scripting en JavaScript/TypeScript
- Escenarios avanzados de concurrencia
- Validación funcional integrada
- Automatización y CI/CD
- Integración con ecosistema de monitoreo
- Resultados claros en consola.
- Escalabilidad
- Open Source y gratuito

Guía de instalación de Visual Studio Code:

1. Descargar el instalador

1. Abre tu navegador (Chrome, Edge, etc.).
2. Ve a la página oficial: https://code.visualstudio.com/
3. Haz clic en Download for Windows (descarga automática del instalador .exe).

2. Ejecutar el instalador

1. Localiza el archivo descargado: VSCodeSetup-x64-x.x.x.exe (en tu carpeta Descargas).
2. Haz doble clic para iniciar el asistente de instalación.
3. Acepta los términos de licencia.
4. Selecciona la carpeta de instalación (por defecto está bien).

3. Configurar la instalación (opcional, recomendado)

En la ventana de opciones, marca:

- Agregar a PATH (para poder ejecutar code desde la terminal)
- Registrar como editor de código soportado
- Agregar acción en el menú contextual "Abrir con Code"

4. Finalizar instalación

1. Haz clic en Instalar.
2. Cuando termine, marca la opción Iniciar Visual Studio Code.

6. Verificar instalación

1. Abre la terminal de Windows (bash).
2. Escribe:
   code –versión

Guía de instalación de K6 en visual studio code:

1. Instala Chocolatey.
   Abre PowerShell como administrador y ejecuta:
   choco install k6
2. Verifica la instalación
   K6 version

Configuración en visual estudio code

1. Abre Visual Studio Code
2. Crea una carpeta para tus pruebas en este caso
   Performance_K6/fakestore_test.js
3. Crea tu script de prueba
4. Ejecuta en la terminal integrada bash
   k6 run fakestore_test.js

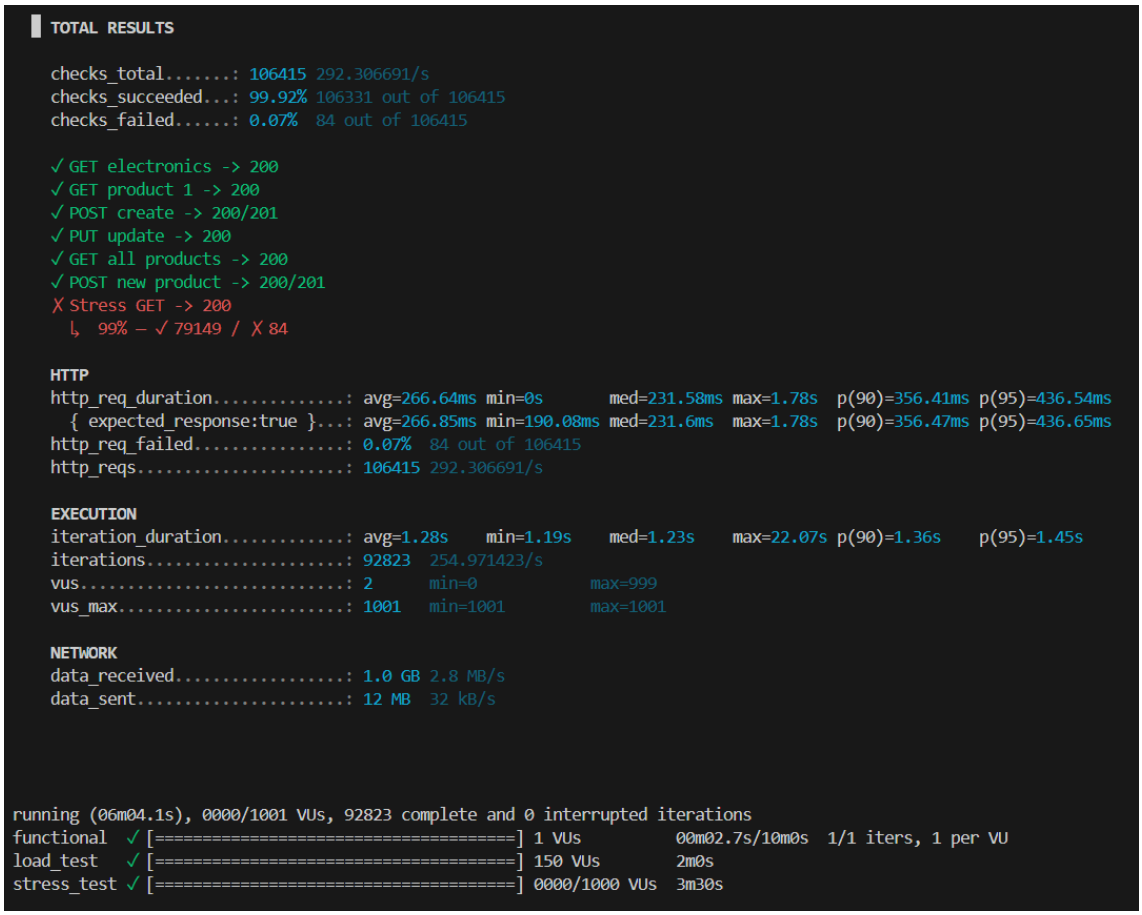
Casos de prueba

Se encuentran definidos en un archivo Excel llamado *Matriz_casos_de_prueba_API* adjunto en la carpeta del entregable.

ID	Escenario	Endpoint	Método	Descripción	Datos de entrada	Resultado esperado	Resutlado
TC01	Funcional	/products/category/electronics	GET	Validar consulta de productos por categoría	N/A	Respuesta **200 OK**, retorna lista de productos	PASSED/FAILURE
TC02	Funcional	/products/1	GET	Validar consulta de producto específico	ID=1	Respuesta **200 OK**, retorna JSON con producto con ID=1	
TC03	Funcional	/products	POST	Validar creación de producto	{ title, price, description, image, category }	Respuesta **200/201**, retorna JSON con id del producto creado	
TC04	Funcional	/products/{id}	PUT	Validar actualización de producto creado	{ image }	Respuesta **200 OK**, el campo image actualizado	
TC05	Carga	/products	GET	Validar consulta de productos bajo carga (150 VUs / 2m)	N/A	Respuestas **200 OK**, tiempo de respuesta dentro del SLA	
TC06	Carga	/products	POST	Validar creación de productos bajo carga (150 VUs / 2m)	Datos dinámicos por cada usuario	Respuestas **200/201**, sin errores de saturación	
TC07	Estrés	/products	GET	Validar estabilidad del endpoint con ramp-up (100 → 1000 VUs)	N/A	Respuestas **200 OK** la mayoría, identificar punto de quiebre (429/500)	
TC08	Transversal	Todos	GET / POST / PUT	Validar integridad de la respuesta	N/A	Respuesta en formato **JSON válido**, sin errores de parseo	
TC09	Transversal	Todos	GET / POST / PUT	Validar cabeceras de respuesta	N/A	Content-Type: application/json presente	
TC10	Transversal	Todos	GET / POST / PUT	Medición de métricas clave	N/A	Registrar **Tasa de éxito, Latencia, Throughput, Errores (4xx/5xx)**	

Informe de ejecución

Tras finalizar la ejecución de las pruebas se genera el siguiente resultado:



1. Resumen General

- Total de verificaciones realizadas: 106,415

- Verificaciones exitosas: 106,331 (99.92%)

- Verificaciones fallidas: 84 (0.07%)

- Tiempo total de ejecución: 06m04s

- Máximo de usuarios virtuales (VUs): 1,001

Conclusión: El sistema presentó un comportamiento estable con un índice de fallos muy bajo (<0.1%).

2. Resultados por Endpoint

- GET electronics → 200 (OK)

- GET product 1 → 200 (OK)

- POST create → 200/201 (OK)

- PUT update → 200 (OK)

- GET all products → 200 (OK)

- POST new product → 200/201 (OK)

- Stress GET → 200 (99% éxito, 84 fallos)

3. Métricas HTTP

- Duración promedio de respuesta: 266.64 ms

- Mínimo: 0 ms

- Mediana: 231.58 ms

- Máximo: 1.78 s

- Percentiles: p90=356.41 ms, p95=436.54 ms

- Interpretación: El 90% de solicitudes respondió en <357 ms, y el 95% en <437 ms. Buen desempeño bajo carga normal.

5. Métricas de Red

- Datos recibidos: 1.0 GB (~2.8 MB/s)

- Datos enviados: 12 MB (~32 kB/s)

- Interpretación: Tráfico mayormente de lectura (GET). Estable en consumo de red.

6. Escenarios Ejecutados

- Prueba funcional: 1 VU, completada en 2.7s

- Prueba de carga: 150 VUs durante 2m, sin errores

- Prueba de estrés: Hasta 1000 VUs durante 3m30s → 84 fallos detectados

- Conclusión: Carga moderada estable, estrés máximo genera fallos menores.

7. Conclusiones y Recomendaciones

- Puntos positivos: Alta tasa de éxito (99.92%), buen desempeño p90/p95, soporta 150 VUs sin errores.

- Riesgos: 84 fallos en estrés GET, pico de 22s, latencias hasta 1.78s.

- Recomendaciones: Revisar escalabilidad de GET bajo estrés, monitorear CPU/RAM/DB, ajustar timeouts y conexiones, repetir pruebas con ramp-up 500–1000 VUs.