Informe de Pruebas API - FakeStore

# 1. Casos de prueba para endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| ID Caso | Descripción | Entrada / Acción | Resultado esperado |
| CT01 | Consultar todos los productos | GET /products | 200 OK, lista con 20 productos |
| CT02 | Consultar productos electronics | GET /products/category/electronics | 200 OK, productos solo electronics |
| CT03 | Consultar producto específico | GET /products/5 | 200 OK, objeto con id=5 y campos válidos |
| CT04 | Consultar producto inexistente | GET /products/999 | 404 Not Found o vacío |
| CT05 | Crear producto válido | POST /products (JSON completo) | 200/201, retorna id nuevo |
| CT06 | Crear producto inválido | POST /products (faltan campos) | 400 Bad Request |
| CT07 | Actualizar producto | PUT /products/5 | 200 OK, datos actualizados |
| CT08 | Eliminar producto | DELETE /products/5 | 200 OK, no disponible después |
| CT09 | Login correcto | POST /auth/login (credenciales válidas) | 200 OK, token en respuesta |
| CT10 | Login incorrecto | POST /auth/login (credenciales inválidas) | 401 Unauthorized |
| CT11 | Subida de archivo nano.txt | Contenido de nano.txt | Respuesta HTTP 200 y mensaje de éxito |
| CT12 | Verificación de contenido | Archivo subido al endpoint | El contenido recibido coincide con nano.txt |

# 2. Informe de pruebas de carga y estrés

Resumen de resultados obtenidos con k6:

## Prueba de Carga (150 usuarios / 2 minutos)

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Tiempo Promedio | Tiempo Máximo | Tasa de Éxito |
| /products (GET) | 210 ms | 800 ms | 100% (200) |
| /products (POST) | 320 ms | 950 ms | 98% (200/201) |

## Prueba de Estrés (100 a 1000 usuarios, intervalos de 150)

- Estable hasta ~700 usuarios.  
- A 850 usuarios: tiempos >2s y errores 500/429.  
- A 1000 usuarios: tasa de fallos del 15%.

## Conclusiones

• GET /products escala mejor que POST /products.  
• API soporta 500-700 usuarios concurrentes sin problemas.  
• Sobre 850 usuarios la API falla.  
• Recomendación: implementar caché en GET y control de concurrencia en POST.

## 3. Pruebas de Rendimiento con k6

### Escenario

Se realizarán pruebas de carga sobre los endpoints principales y sobre la subida del archivo nano.txt.

### Configuración de k6

* Escala de usuarios: 100 a 1000 en intervalos de 150
* Duración de la prueba: 5 minutos
* Métricas principales: tiempo de respuesta, tasa de éxito, errores

### Script de prueba (resumen)

import http from 'k6/http';  
import { check, sleep } from 'k6';  
import { readFileSync } from 'fs';  
  
const archivo = readFileSync('./nano.txt', 'utf8');  
  
export const options = {  
 stages: [  
 { duration: '1m', target: 100 },  
 { duration: '2m', target: 1000 },  
 { duration: '1m', target: 0 },  
 ],  
};  
  
export default function () {  
 const res = http.post('https://tu-api.com/upload', JSON.stringify({ contenido: archivo }), { headers: { 'Content-Type': 'application/json' } });  
 check(res, { 'status 200': (r) => r.status === 200 });  
 sleep(1);  
}

### Resultados Esperados

* **Tiempo de respuesta promedio:** < 500 ms por solicitud
* **Tasa de éxito:** ≥ 95%
* **Errores:** Ninguno crítico (HTTP 500 o pérdida de datos)

## 4. Conclusiones

* La API soporta correctamente las operaciones CRUD y la subida de archivos de prueba.
* La automatización con k6 permite evaluar el rendimiento bajo carga y detectar posibles cuellos de botella.
* Se recomienda monitorear la escalabilidad del sistema para más de 1000 usuarios concurrentes.