Informe de Resultados – Prueba de Carga (JMeter/K6)

1. Descripción de la Prueba

La prueba tuvo como objetivo evaluar el desempeño del servicio App Transaction Balance bajo condiciones de carga con hasta 140 usuarios virtuales concurrentes (VUs).  
El propósito fue medir el comportamiento del sistema frente a múltiples solicitudes simultáneas y determinar su estabilidad, tiempos de respuesta y posibles cuellos de botella.

2. Resultados Generales

|  |  |
| --- | --- |
| Métrica | Valor Obtenido |
| Usuarios virtuales (máx.) | 140 |
| Solicitudes totales | 269,891 |
| Solicitudes fallidas | 6,759 (2.44%) |
| Tasa de éxito (checks OK) | 97.55% |
| Promedio de duración total de solicitud | 861.68 ms |
| Mediana de duración | 613.42 ms |
| Percentil 90 (p90) | 1.28 s |
| Percentil 95 (p95) | 1.57 s |
| Promedio de envío | 43.2 μs |
| Promedio de espera (waiting) | 861.21 ms |
| Datos recibidos | 842 MB (223 kB/s) |
| Datos enviados | 588 MB (156 kB/s) |

3. Análisis de Resultados

3.1. Tasa de Éxito

El sistema presentó un 97.55% de respuestas exitosas, lo cual indica una buena estabilidad general.  
No obstante, el 2.44% de fallas HTTP representa aproximadamente 6,759 solicitudes con errores (principalmente códigos 4xx y 5xx).

Esto puede deberse a saturación temporal del servidor o límites en la capacidad de manejo de conexiones concurrentes.

3.2. Tiempos de Respuesta

El tiempo medio de respuesta (861.68 ms) y la mediana (613.42 ms) son aceptables para entornos web, aunque el percentil 95 (1.57 s) muestra que en momentos de mayor carga, los tiempos se duplican.

Esto evidencia que:

El sistema mantiene tiempos aceptables bajo **carga moderada.**

En picos de concurrencia, hay una ligera degradación del rendimiento (posible saturación de CPU o red).

3.3. Análisis del Diagrama VU vs. Requests

El gráfico muestra que:

Los usuarios virtuales (VUs) se mantuvieron en 140 durante la mayor parte de la prueba.

Se observan caídas momentáneas en el número de solicitudes por segundo (HTTP req/s) cerca del minuto 1:50 y 2:00, aunque los VUs permanecen activos.

Esto sugiere que el backend no respondió consistentemente en esos momentos, causando un descenso en las peticiones procesadas aun cuando los usuarios seguían generando carga.

Este comportamiento suele indicar problemas de rendimiento intermitente en el servidor (p. ej., pausas del GC, bloqueo de hilos o limitaciones de base de datos).

3.4. Comportamiento por Etapas

|  |  |  |
| --- | --- | --- |
| Métrica | Etapa | Resultado |
| y\_failed\_request\_stage\_0\_HTTP5xx | Fase inicial | 2 errores (0.00026/s) |
| y\_failed\_request\_stage\_1\_HTTP4xx | Fase media | 769 errores (0.23/s) |
| y\_failed\_request\_stage\_1\_HTTP5xx | Fase media | 5987 errores (1.58/s) |
| y\_failed\_request\_stage\_2\_HTTP5xx | Fase final | 2 errores (0.00052/s) |

Se evidencia que el mayor número de fallos (HTTP 5xx) ocurrió durante la fase intermedia, momento donde la carga fue más sostenida.

4. Conclusiones

**Rendimiento general aceptable:** El 97.55% de éxito indica que el servicio soporta correctamente la mayoría de las peticiones bajo la carga aplicada.

**Fallas bajo alta concurrencia:** Los errores 5xx se concentran en los momentos de máxima demanda, lo que podría apuntar a un límite de capacidad del servidor o de la capa de base de datos.

**Degradación temporal:** El descenso en solicitudes procesadas durante la prueba evidencia que el sistema no mantiene un throughput constante bajo carga máxima.

**Tiempos dentro del rango esperado:** La mediana y los percentiles están dentro de límites razonables para una API (p95 < 1.6 s).

5. Recomendaciones

a. Optimización del servidor:

Revisar parámetros de thread pool, conexiones a base de datos y configuración de timeouts. Evaluar uso de caching o balanceadores si no existen actualmente.

b. Repetir la prueba con mayor duración:

Una ejecución de al menos 1 hora sostenida permitirá confirmar estabilidad a largo plazo.

c. Monitorear infraestructura:

Registrar métricas de CPU, memoria, I/O y base de datos durante la prueba para correlacionar los picos de errores.

d. Validar capacidad de escalado:

Probar incrementos progresivos (por ejemplo, de 100 a 200 VUs) para identificar el punto de saturación.

e. Corregir errores HTTP 5xx:

Analizar logs del servidor para determinar si fueron causados por timeout, falla de autenticación, o errores internos de API.