Informe de Resultados — Prueba de Carga

1. Resumen Ejecutivo

Se ejecutó una prueba de carga con hasta 140 usuarios virtuales (VUs). Durante la prueba se alcanzó un promedio de ~73 solicitudes por segundo (req/s) y un total de ~276.650 peticiones. El porcentaje de verificaciones exitosas fue ≈97,55%, con una tasa de fallos global ≈2,44%. Los tiempos de respuesta exhibieron p90 ≈1,28s y p95 ≈1,57s (promedio ≈0,86s), con picos que llegaron hasta ~29,93s. Se observaron caídas significativas en el throughput (req/s) mientras los VUs se mantenían en 140, lo que indica probables cuellos de botella del lado del servidor o mecanismos de protección que limitaron la tasa efectiva de peticiones.

2. Alcance y Escenario de Prueba

- **Herramienta:** k6.
- **Perfil de usuarios:** mínimo 2 VUs, máximo 140 VUs (sosteniendo 140 VUs por un periodo significativo).
- **Métrica objetivo:** latencia (percentiles), throughput (req/s), errores (4xx/5xx), y relación VUs vs req/s.
- **Endpoints bajo prueba:** funcionalidad de "Transaction Balance".

3. Resultados Clave

Métrica	Valor observado
Usuarios virtuales (VUs)	min 2, máx 140
Solicitudes totales (http_reqs)	≈ 276.650
Throughput promedio	≈ 73,17 req/s
Checks exitosos	≈ 97,55%
Tasa de fallos (global)	≈ 2,44%
Errores por tipo	HTTP 4xx: ~769 HTTP 5xx: ~5.989 (repartidos en etapas)
Latencia promedio (http_req_duration)	≈ 861 ms
Latencia p90 / p95	≈ 1,28 s / ≈ 1,57 s

Latencia máx	≈ 29,93 s
Duración de iteración promedio	≈ 1,86 s (p95 ≈ 2,57 s)
Datos recibidos / enviados	≈ 842 MB / ≈ 588 MB

Observaciones adicionales:

- La mayor parte del tiempo de la solicitud se invierte en "waiting" (tiempo de servidor), lo que sugiere que el cuello de botella principal está del lado de la aplicación/infraestructura y no en la red.
- TLS handshaking, conexión y envío/recepción muestran tiempos medios en el orden de microsegundos, lo que respalda que la red no es el limitante predominante.
- La presencia de múltiples respuestas 5xx indica errores del servidor bajo carga.

4. Relación VUs vs Peticiones por Segundo (req/s)

El diagrama de monitoreo muestra que, aun manteniéndose los VUs en ~140, el throughput (http_reqs/s) sufrió caídas abruptas en dos periodos. Esto es característico de saturación en la capa de aplicación o en dependencias (p. ej., base de datos, servicios aguas abajo) o de mecanismos de protección (p. ej., límites de concurrencia, colas, backpressure). Tras cada caída, el throughput se recupera parcialmente, pero permanece por debajo del potencial esperado para 140 VUs.

Resumen de métricas reportado por la herramienta:

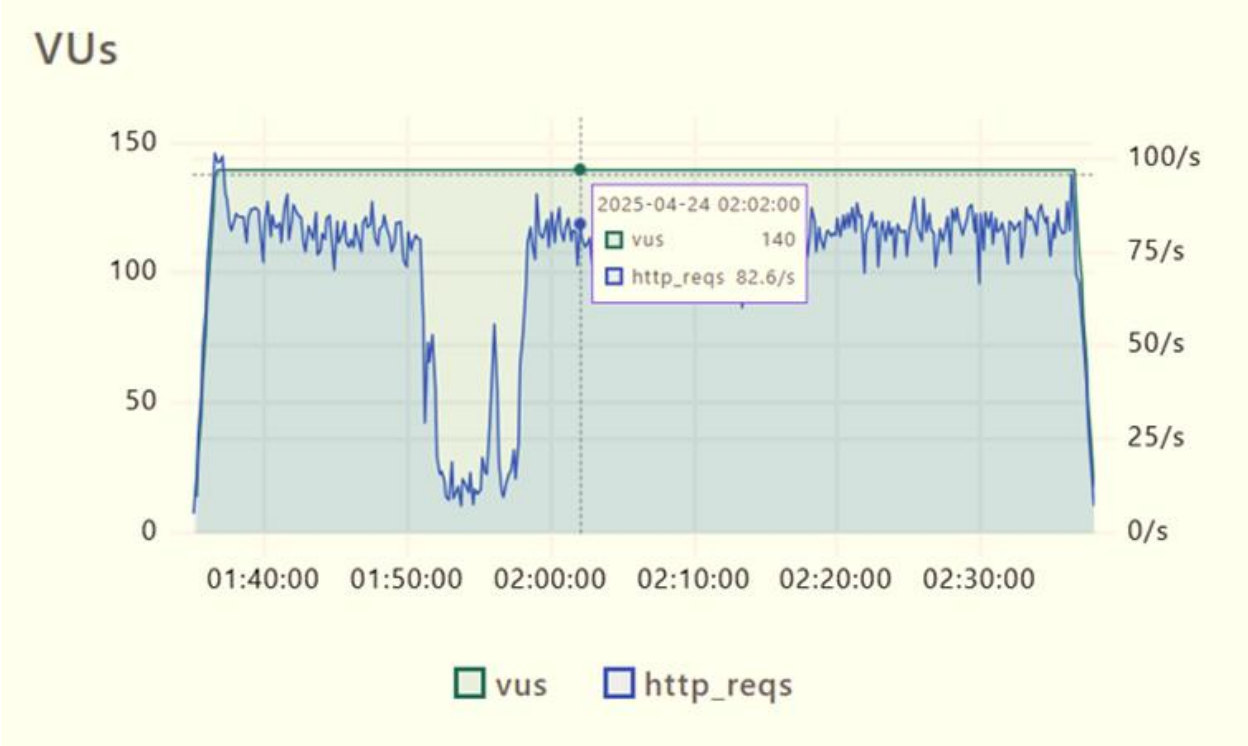
textSummary.txt

```
✗ App Transaction Balance response was OK
  ↳  97% — ✓ 269891 / ✗ 6759

  █ setup

✓ checks.............................: 97.55% ✓ 269891    ✗ 6759
  data_received.....................: 842 MB 223 kB/s
  data_sent.........................: 588 MB 156 kB/s
  failed_request....................: 2.44%  ✓ 6759      ✗ 269891
  http_req_blocked..................: avg=10.97µs  min=0s       med=0s       max=35.02ms p(90)=0s      p(95)=0s
  http_req_connecting...............: avg=3.3µs    min=0s       med=0s       max=11.82ms p(90)=0s      p(95)=0s
✓ http_req_duration.................: avg=861.68ms min=191.86ms med=613.42ms max=29.93s  p(90)=1.28s   p(95)=1.57s
  ✓ { expected_response:true }.....: avg=735.84ms min=244.92ms med=600.7ms  max=26.72s  p(90)=1.22s   p(95)=1.42s
✓ http_req_failed...................: 2.44%  ✓ 6759      ✗ 269891
  http_req_receiving................: avg=424.03µs min=0s       med=320.7µs  max=39.58ms p(90)=988.4µs p(95)=1.05ms
  http_req_sending..................: avg=43.22µs  min=0s       med=0s       max=31.25ms p(90)=0s      p(95)=517µs
  http_req_tls_handshaking..........: avg=7.36µs   min=0s       med=0s       max=27.02ms p(90)=0s      p(95)=0s
  http_req_waiting..................: avg=861.21ms min=191.86ms med=613.01ms max=29.93s  p(90)=1.28s   p(95)=1.57s
✓ http_reqs.........................: 276650 73.176857/s
  iteration_duration................: avg=1.86s    min=0s       med=1.61s    max=30.94s  p(90)=2.29s   p(95)=2.57s
✓ iterations........................: 276650 73.176857/s
  vus...............................: 2      min=2       max=140
  vus_max...........................: 140    min=140     max=140
  y_failed_request_stage_0_HTTP5xx...: 1      0.000265/s
  y_failed_request_stage_1_HTTP4xx...: 769    0.203409/s
  y_failed_request_stage_1_HTTP5xx...: 5987   1.583625/s
  y_failed_request_stage_2_HTTP5xx...: 2      0.000529/s
```

Relación VUs vs req/s observada en el monitoreo:



5. Hallazgos

1) Errores del servidor (5xx) bajo carga (~6k eventos): indican que la aplicación o algún componente dependiente no soporta el volumen/concurrencia alcanzada.
2) Latencias elevadas en p90/p95 (≈1,28–1,57 s) y máximos cercanos a 30 s: síntomas de saturación o timeouts internos.
3) Throughput inestable con VUs constantes: evidencia de throttling/saturación, colas creciendo o lock contentions en recursos críticos (DB, cachés, servicios internos).
4) Tiempos de red despreciables frente al "waiting": el problema no es de red sino de procesamiento en servidor o dependencias.

6. Conclusiones

La plataforma no mantiene un desempeño consistente a 140 VUs: aparecen errores 5xx y aumentos de latencia que degradan la experiencia. El cuello de botella se ubica principalmente en la capa de aplicación/infraestructura (tiempo de servidor y dependencias), no en la red. Se requiere optimización y, posiblemente, escalamiento vertical/horizontal y/o ajuste de dependencias (DB, cachés, colas) para sostener o incrementar el throughput con estabilidad.

7. Recomendaciones

- Revisar logs y traces durante los periodos de caída de throughput para identificar excepciones, timeouts, o límites de recursos.
- APM y profiling para medir tiempos por capa/endpoint y detectar funciones o queries costosas.
- Base de datos: optimizar queries e índices. Revisar conexiones máximas, pool y locks. Aplicar caché para lecturas frecuentes.
- Escalamiento: aumentar réplicas de aplicación y componentes dependientes. Validar políticas de autoescalado.Revisar límites de concurrencia y de threads.
- Timeouts y reintentos: ajustar timeouts razonables y backoff exponencial para evitar tormentas de reintentos bajo saturación.
- Pruebas adicionales: repetir con cargas controladas (stress/soak) y encontrar el punto de quiebre. Ejecutar pruebas por endpoint crítico. Medir capacidad por SLA (p90<1s, p95<1.5s, error rate<1%, como ejemplo).
- Protecciones: aplicar circuit breakers y rate limiting para aislar fallos y prevenir cascadas.

8. Próximos Pasos Propuestos

1) Correlacionar las ventanas de caída en req/s con métricas de infraestructura (CPU, memoria, GC, conexiones DB, latencia de servicios internos).
2) Ejecutar una prueba de línea base a 50–80 VUs para confirmar estabilidad y establecer un umbral seguro actual.
3) Iterar en optimizaciones y volver a ejecutar pruebas comparativas (antes/después) manteniendo la misma metodología y set de datos.