# Plan de pruebas

## Objetivo

Evaluar el rendimiento y la capacidad de respuesta de la aplicación Cloud Conversion Tool bajo diferentes escenarios y cargas de trabajo.

## Objetivos específicos

- Evaluar el tiempo de respuesta de la aplicación bajo diferentes cargas de trabajo.
- Determinar el límite de capacidad de la aplicación en términos de usuarios concurrentes y transacciones por minuto.
- Identificar y abordar cuellos de botella en el rendimiento del sistema.
- Comprobar que la aplicación cumple con los criterios de aceptación definidos en cuanto a tiempo de respuesta y tasa de errores.

## Descripción general

La prueba evaluará el rendimiento de la aplicación de conversión de formatos de video bajo diferentes escenarios de carga simulada. Conforme a nuestros criterios de aceptación, la aplicación deberá mantener tiempos de respuesta aceptables y tasas de errores bajas bajo estas condiciones.

## Tipos de prueba a realizar

- Pruebas de capacidad
- Pruebas de carga

## Entorno de prueba

El entorno de pruebas requerido para realizar este plan debe coincidir con las siguientes características a nivel de hardware y software de un dispositivo local.

Hardware:
- Procesador: 2.6 GHz Intel Core i7 de seis núcleos
- Memoria: 16 GB 2667 MHz DDR4

Software:
- SO: macOS Ventura Versión 13.6

Red:
- Se configura interfaz de red tipo Bridge Network, con la finalidad de obtener la misma información de conexión del Host.

## Herramientas utilizadas

Entre las tecnologías que utilizamos se encuentran Docker, que nos permite empacar nuestras aplicaciones en contenedores para lograr un despliegue sencillo y consistente en diversos entornos. Empleamos Redis, un ágil motor de base de datos en memoria que ofrece un almacenamiento eficiente y opcionalmente persistente. Para la ejecución de tareas paralelas y asíncronas, hemos incorporado Celery, una biblioteca de Python de código abierto. Adicionalmente, utilizamos Postman para realizar pruebas y verificar el funcionamiento de nuestros endpoints. Por último, nuestra base de datos relacional está respaldada por Postgres, un sistema de gestión de bases de datos de código abierto y orientado a objetos.

Además, las herramientas de prueba a utilizar son:

- JMeter para la creación y ejecución de escenarios de prueba.
- Herramientas de monitoreo de recursos para supervisar el rendimiento del sistema.

## Métricas consideradas

- Respuestas HTTP por segundo.
- Tiempo mínimo, máximo y promedio de respuesta.
- Respuestas por códigos HTTP.
- Uso de CPU, memoria y almacenamiento.

## Riesgos y limitaciones

La disponibilidad de recursos como hardware, ancho de banda de red y servidores puede impactar la ejecución de las pruebas.

Las pruebas se realizan en un entorno controlado, lo que significa que los resultados pueden no reflejar completamente el rendimiento en un entorno de producción real.

## Datos de prueba

Cuentas de Usuario:

Se utilizarán cuentas de usuario válidas para simular las operaciones de inicio de sesión y autenticación. Cada cuenta de usuario deberá incluir la siguiente información:

- Email válido.
- Nombre de usuario (username).
- Contraseña válida para la autenticación.

Datos de Entrada para Tareas:

Para las pruebas de creación de tareas de conversión, se necesitarán datos de entrada válidos. Cada tarea de conversión debe incluir:

- Nombre del archivo multimedia que se desea convertir.
- Formato de conversión al que se desea transformar el archivo.

Bases de Datos de Prueba:

Se utilizará una base de datos de prueba para reflejar el estado inicial del sistema antes de ejecutar las pruebas. La base de datos de prueba debe contener datos consistentes con las cuentas de usuario, tareas de conversión y otros datos necesarios para las pruebas.

## Criterios de aceptación

- El 100% de las peticiones configuradas y realizadas deben ser atendidas por el aplicativo.
- El tiempo de respuesta de las peticiones debe ser menos a 2 segundos y debe ser un código status exitoso (Familia de los códigos 2XX
- Criterios de exitoso (para configuraciones diferentes)
- Se considera exitoso a la combinación de ajustes en el consumo de memoria menor al 80% asignada al recurso computacional.
- La aplicación Docker no debe de utilizar más del 70% del CPU disponible cuando los contenedores estén ejecución.
- El 100% de las peticiones configuradas y realizadas deben ser atendidas por el aplicativo con la combinación de ajustes definidas.

En relación con el desarrollo de la aplicación en entornos de prueba se definieron los siguientes criterios de aceptación:

- Escenario 1:
    - El sistema debe convertir el archivo al formato especificado por el usuario sin errores.
    - El archivo convertido debe tener un formato correcto y ser funcional según el nuevo formato.
    - El tiempo promedio de respuesta de la aplicación debe ser menos de 1500 ms.
    - La aplicación debe soportar como mínimo el 99% de los requests enviados en la prueba.
    - Evaluar la configuración de ajustes donde se obtengan las características de rendimiento más deseables.
- Escenario 2:
    - Se debe probar la cantidad de archivos que la aplicación puede procesar en 1 minuto.
    - Evaluar máximo hasta que el tiempo en cargar un archivo a la aplicación tarde 600 segundos.
    - Los archivos enviados en la prueba deben de ser máximo de 5 MB.
    - Evaluar la configuración de ajustes donde se obtengan las características de rendimiento más deseables.

# Escenarios de prueba

## Escenario 1

Subir y cambiar el formato de un archivo. El usuario debe proveer el archivo que desea convertir, el formato al cual desea cambiarlo y el token de autenticación para realizar dicha operación. El archivo debe ser almacenado en la plataforma, se debe guardar en base datos la marca de tiempo en el que fue subido el archivo y el estado del proceso de conversión (uploaded).

Datos de prueba: Nombre del archivo, formato de conversión, token del usuario

Métricas a recopilar: Tiempo de respuesta.

## Escenario 2

Para el segundo escenario se probará la máxima cantidad de conversón de formatos de archivos procesados de capacidad no mayor a 10MB que se pueden procesar en un minuto por un total de cincuenta usuarios de manera concurrente. Las métricas que se evaluarán durante la prueba serán capacidad de procesamiento mediante el procesamiento de archivos por minuto, tiempo de respuesta promedio y utilización de recursos. Las tres métricas deben cumplir con los criterios de aceptación definidos anteriormente para que la prueba se considere exitosa. Esta información se representará en una gráfica que ilustre el comportamiento de la aplicación en base a la recopilación de la prueba.

Datos de prueba: Nombre del archivo, formato de conversión, token del usuario

Métricas a recopilar: Tiempo de respuesta promedio y utilización de recursos.

# Ejecución de las pruebas

ESCENARIO 1

- 1000 peticiones con 300 de concurrencia

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 1005 | 1951 | 0 | 3539 | 764.89 | 0.00% | 104.7/sec | 25.56 | 54.71 | 250.0 |
| TOTAL | 1005 | 1951 | 0 | 3539 | 764.89 | 0.00% | 104.7/sec | 25.56 | 54.71 | 250.0 |

- 2000 peticiones con 300 de concurrencia

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 2010 | 3441 | 0 | 12109 | 1370.70 | 3.68% | 79.8/sec | 19.35 | 41.71 | 248.2 |
| TOTAL | 2010 | 3441 | 0 | 12109 | 1370.70 | 3.68% | 79.8/sec | 19.35 | 41.71 | 248.2 |

- 3000 peticiones con 300 de concurrencia

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 3015 | 2507 | 0 | 3864 | 623.90 | 0.00% | 116.5/sec | 28.45 | 60.88 | 250.0 |
| TOTAL | 3015 | 2507 | 0 | 3864 | 623.90 | 0.00% | 116.5/sec | 28.45 | 60.88 | 250.0 |

- 4000 peticiones con 300 de concurrencia

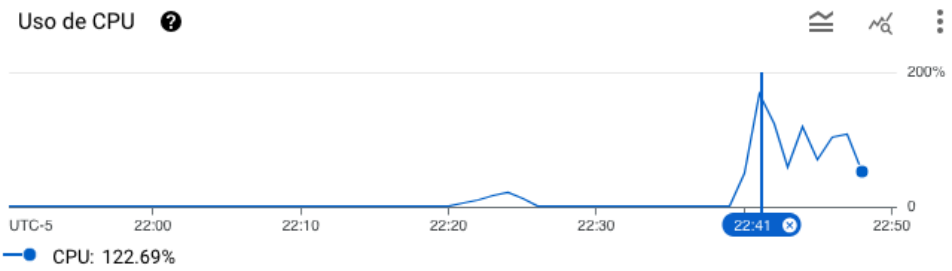
| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 4020 | 2584 | 0 | 9456 | 837.60 | 0.00% | 115.1/sec | 27.99 | 60.14 | 249.0 |
| TOTAL | 4020 | 2584 | 0 | 9456 | 837.60 | 0.00% | 115.1/sec | 27.99 | 60.14 | 249.0 |

- 5000 peticiones con 300 de concurrencia

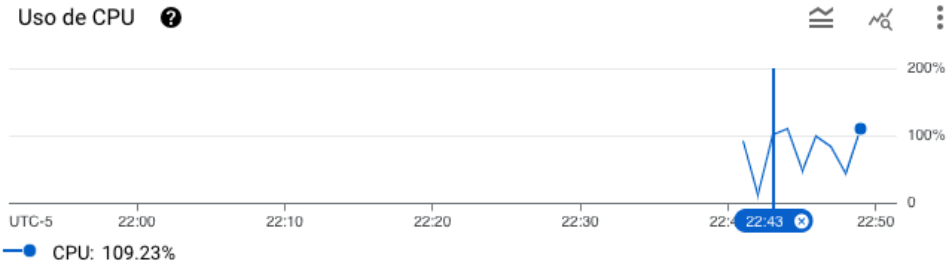
| Label | # Samples | Average | Median | 90% Line | 95% Line | 99% Line | Min | Maximum | Error % | Throughput | Received KB/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 5025 | 2571 | 2604 | 3381 | 4172 | 6263 | 152 | 12476 | 0.00% | 114.5/sec | 27.93 | 59.81 |
| TOTAL | 5025 | 2571 | 2604 | 3381 | 4172 | 6263 | 152 | 12476 | 0.00% | 114.5/sec | 27.93 | 59.81 |

**Uso de CPU en la capa web (3 instancia con autoscaling) en el Escenario 1**

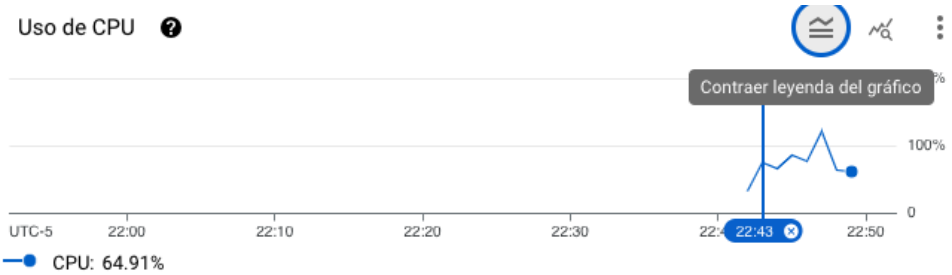
- **App 1**




- **App 2**




- **App 3**




**Uso de CPU en la capa web (5 instancia con autoscaling) en el Escenario 1**

**Tabla comparativa de tiempos con sprint anterior**

| Escenario #1 | Entrega #3 | Entrega #4 |
|---|---|---|
| **Descripción del escenario** | **Tiempo respuesta (milisegundos)** | **Tiempo respuesta (milisegundos)** |
| 1000 peticiones con 300 de concurrencia | 3837 | 1951 |
| 2000 peticiones con 300 de concurrencia | 4487 | 3441 |
| 3000 peticiones con 300 de concurrencia | 3780 | 2507 |
| 4000 peticiones con 300 de concurrencia | 2404 | 2584 |
| 5000 peticiones con 300 de concurrencia | 1611 | 2571 |

**ESCENARIO 2**

- 50 peticiones con 50 de concurrencia (2 seg)

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 50 | 322 | 0 | 617 | 83.75 | 0.00% | 21.4/sec | 5.15 | 11.17 |
| TOTAL | 50 | 322 | 0 | 617 | 83.75 | 0.00% | 21.4/sec | 5.15 | 11.17 |

- 100 peticiones con 50 de concurrencia

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 100 | 377 | 0 | 708 | 113.19 | 0.00% | 37.0/sec | 8.95 | 19.34 |
| TOTAL | 100 | 377 | 0 | 708 | 113.19 | 0.00% | 37.0/sec | 8.95 | 19.34 |

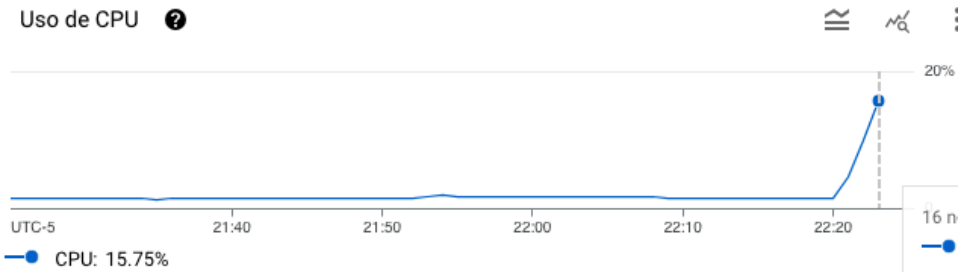
- 150 peticiones con 50 de concurrencia

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 150 | 702 | 0 | 1191 | 223.88 | 0.00% | 37.4/sec | 9.06 | 19.54 |
| TOTAL | 150 | 702 | 0 | 1191 | 223.88 | 0.00% | 37.4/sec | 9.06 | 19.54 |

- 200 peticiones con 50 de concurrencia

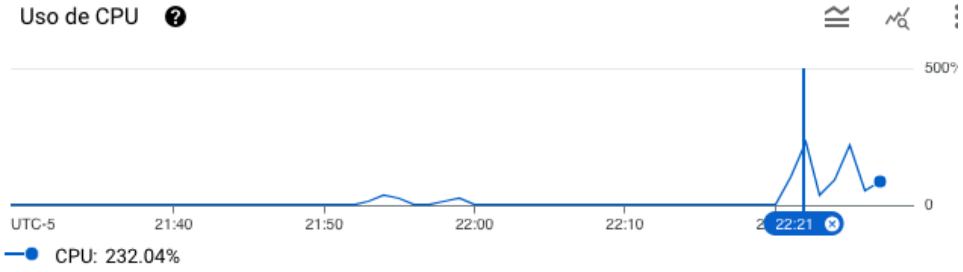
| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|
| HTTP Request | 200 | 841 | 0 | 1968 | 294.26 | 0.00% | 37.1/sec | 8.98 | 19.38 |
| TOTAL | 200 | 841 | 0 | 1968 | 294.26 | 0.00% | 37.1/sec | 8.98 | 19.38 |

**Uso de CPU en la capa web (1 instancia no autoscaling) en el Escenario 2**

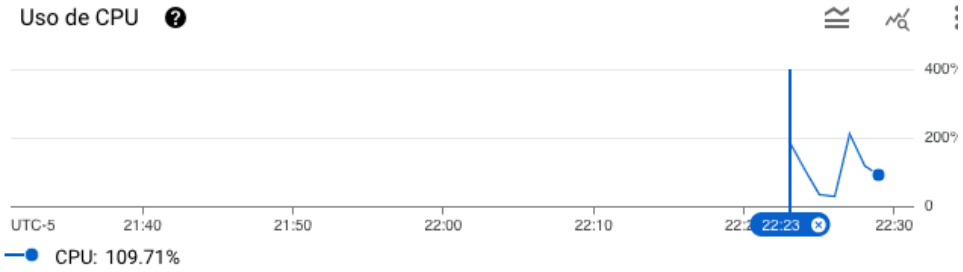



**Uso de la CPU en la capa worker (5 instancias autoscaling) en el Escenario 2**

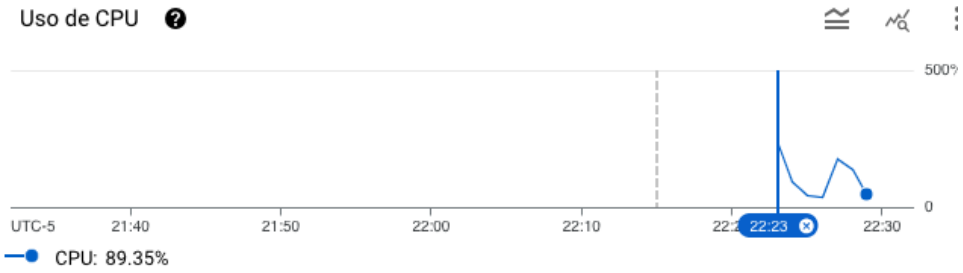
- **Worker 1**



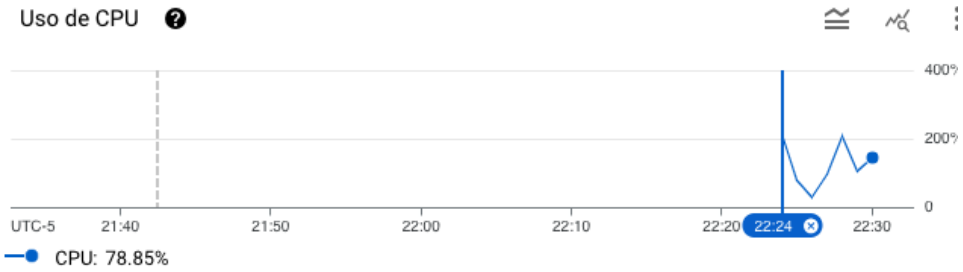

- **Worker 2**




- **Worker 3**




- **Worker 4**




- **Worker 5**

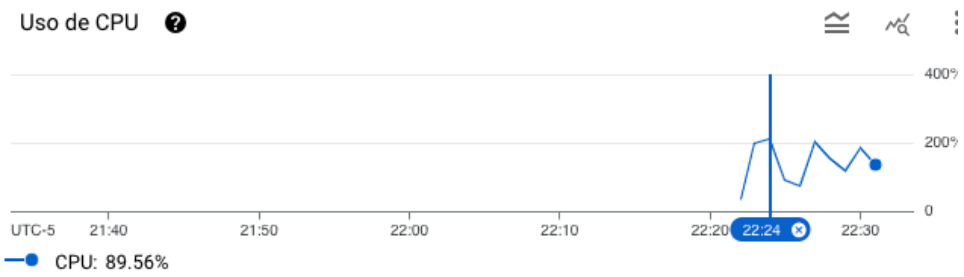
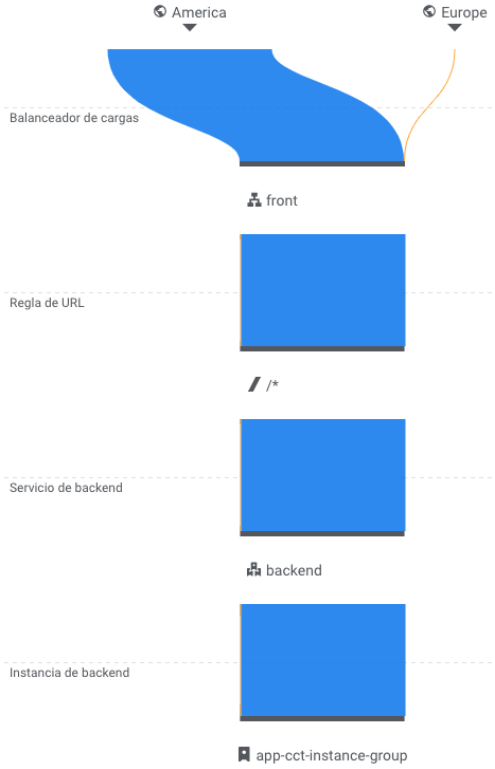



**Tabla comparativa de tiempos con sprint anterior**

| Escenario #1 | Entrega #3 | Entrega #4 |
|---|---|---|
| **Descripción del escenario** | **Tiempo respuesta (milisegundos)** | **Tiempo respuesta (milisegundos)** |
| 50 peticiones con 50 de concurrencia | 144 | 322 |
| 100 peticiones con 50 de concurrencia | 396 | 377 |
| 150 peticiones con 50 de concurrencia | 577 | 702 |
| 200 peticiones con 50 de concurrencia | 630 | 841 |

## Evidencia del tráfico de red en el Load Balancer




## Análisis de resultados

El análisis de los tiempos de respuesta en el primer escenario de conversión de archivos muestra una relación clara entre el aumento de la concurrencia y los tiempos de respuesta. A medida que se incrementa la concurrencia, se observa un aumento en los tiempos de respuesta. Esto se debe a que el servidor se ve sometido a una mayor presión, ya que debe administrar y procesar varias solicitudes simultáneamente. Sin embargo, cuando llegamos a las 3000 peticiones, el sistema empieza a usar el escalado automático y empieza a optimizar los tiempos de respuesta al usar toda la capacidad programada.
Por el lado del Worker, se nota una notable mejoría con el tiempo y cargas no tan altas. A medida que se incrementa la carga (> 3000 peticiones), se mantienen unos buenos tiempos de respuesta sin importar la carga pero se incrementa en comparación con el sprint anterior. Esto indica que la coordinación del flujo de trabajo en la capa del Worker está generando retraso pues al incrementar el uso de la CPU para atender las demandas, se crean múltiples instancias (no una, sino más de una a la vez). Se nota un uso importante del uso de CPU llevando al límite la capacidad de instancias permitidas para la prueba.

En el segundo escenario de conversión de múltiples archivos, las métricas clave son el tiempo de respuesta promedio y la utilización de recursos. El tiempo de respuesta promedio es generalmente bajo en todas las cargas de prueba, lo que indica una buena capacidad de respuesta del sistema incluso con una alta concurrencia. Esto es positivo y sugiere que el sistema puede manejar múltiples solicitudes concurrentes de manera eficiente. Sin embargo, realizando una comparación con la arquitectura de la entrega anterior, se puede observar que los tiempos de respuesta promedio y el uso máximo de la CPU incrementaron aunque que no sobrepasó el 25% (en la capa Web) lo cual es

un resultado muy bueno para el escenario planteado.
Por el lado del Worker, se pueden apreciar los picos de trabajo debo a la alta demanda de procesamiento de las solicitudes. Luego de procesar las solicitudes, los recursos buscan estabilizarse. Durante estos picos y la organización de la distribución de carga, se puede notar que afecta los tiempos de respuesta.

## Conclusiones

- La aplicación presentó un balanceo de carga y un escalado automático al recibir más de 3000 peticiones de manera simultánea conservando unos buenos tiempos de respuesta y uso de CPU. Son más demorados que en el sprint anterior pero sigue manejando un buen tiempo a mayor carga.
- Cuando se realizan pocas peticiones, el uso de CPU es relativamente bajo y estable.
- La aplicación Worker experimentó un aumento en el uso de recursos de CPU y memoria a medida que se incrementaron las peticiones y la concurrencia. El sistema escala correctamente al detectar este incremento permitiendo manejar cargas elevadas sin comprometer el rendimiento.
- Para la capa del worker con cargas altas, se plantea una solución para mejorar el uso de CPU y así los tiempos de respuesta. Esto sería incrementar de manera vertical u horizontal la capacidad del worker para la prueba propuesta.
- La capa del worker presentó una alta escritura en disco durante las pruebas pero nunca se presentaron bloqueos.
- Las pruebas con JMeter permitieron evaluar la capacidad de procesamiento de la aplicación en diferentes situaciones.
- El monitoreo detallado de métricas, como el uso de CPU, tráfico de red y conexiones, proporcionó información valiosa sobre el comportamiento de la aplicación bajo carga.
- Los registros por nivel de gravedad destacaron la importancia de abordar errores críticos, como la falta de memoria, que pueden afectar significativamente la estabilidad y el rendimiento de la aplicación.
- La integración de herramientas como JMeter resultó útil para realizar pruebas de carga y analizar los resultados de manera efectiva, permitiendo la identificación de áreas de mejora y la optimización de la aplicación.