Informe de pruebas de performance.

Proyecto: API Restful-Booker.

Flujo/funcionalidad: (Crear y Consultar Reservas).
Url de la API, para mayor información de los flujos:
https://restful-booker.herokuapp.com/apidoc/index.html#api-Booking-CreateBooking

Pruebas de Performance:

Fecha de ejecución de la prueba: 05 al 06 de enero 2025.

Fecha del informe: 05 al 06 de enero 2025

Introducción

El presente informe detalla el proceso, análisis y resultados de las pruebas de performance realizadas a la API "Crear y Consultar Reservas". El objetivo principal es evaluar la capacidad de la API para manejar diferentes niveles de carga (poca, moderada y alta concurrencia), identificar posibles cuellos de botella, y verificar si cumple con los estándares de rendimiento definidos para escenarios específicos. Las pruebas se realizaron utilizando JMeter y K6 como herramientas principales para la simulación de carga y InfluxDB para almacenar los datos con Grafana para la visualización de métricas. No se contó con acceso al servidor, por lo que no fue posible incluir métricas del uso de CPU y memoria.

Análisis de Requerimientos

Se identificaron los siguientes puntos clave para las pruebas de performance:

- **KPIs Relevantes:**
  - ✓ **Tiempo promedio de respuesta:** Promedio de tiempo que tarda la API en responder.
  - ✓ **Percentiles**: Percentiles 90%, 95% y 99% para analizar tiempos de respuesta en diferentes rangos.
  - ✓ **Throughput**: Número de solicitudes atendidas por segundo.
  - ✓ **Errores (%)**: Porcentaje de errores durante la ejecución.
  - ✓ **Velocidad de datos (KB/sec)**: Cantidad de datos enviados y recibidos por segundo.

- **Objetivos:**
  - ✓ Verificar la escalabilidad de la API.
  - ✓ Evaluar la confiabilidad y estabilidad bajo diferentes niveles de carga.
  - ✓ Identificar posibles cuellos de botella.

Diseño de Pruebas

Escenarios Claves

Se identificaron los siguientes escenarios para evaluar el rendimiento de la API:

Escenario de Prueba 1: Poca Concurrencia:

Detalles del Escenario:

1) **Carga Simulada:** Se simularon 10 usuarios concurrentes realizando el proceso de **crear y consultar una reserva mediante la API**.
2) **Duración de la Prueba:** La prueba se llevó a cabo durante un periodo de tiempo de una (6) minutos que permitió alcanzar una carga significativa y evaluar el rendimiento de la API para estos dos flujos, bajo condiciones sostenidas.
3) **Configuración:** Se utilizaron configuraciones específicas incluyendo el número de hilos (10 usuarios), tiempo de rampa, y cualquier configuración especial necesaria para replicar el escenario de producción.
4) **Datos de Entrada:** Se utilizaron datos de entrada representativos como clientes ficticios generados al azar (Nombres y Apellidos) para garantizar que la prueba refleje un escenario realista.
5) **Escenario de Error:** Se incluyeron transacciones que generan errores de negocio para simular situaciones no ideales y evaluar cómo el servicio maneja situaciones excepcionales.
6) **Métricas de Interés:** Se monitorearon métricas clave, como tiempos de respuesta, errores, throughput, y uso de recursos, para evaluar el rendimiento del servicio.
7) **Justificación**: Este escenario representa el tráfico mínimo esperado en condiciones normales de uso, como fuera del horario pico o cuando el sistema está siendo usado por un número limitado de usuarios finales. Sirve para establecer un punto base de rendimiento y detectar posibles problemas en cargas ligeras.

Escenario de Prueba 2: Carga moderada:

Detalles del Escenario:

1) **Carga Simulada:** Se simularon 50 usuarios concurrentes realizando el proceso de **crear y consultar una reserva mediante la API**.
2) **Duración de la Prueba:** La prueba se llevó a cabo durante un periodo de tiempo de una (6) minutos que permitió alcanzar una carga significativa y evaluar el rendimiento de la API para estos dos flujos, bajo condiciones sostenidas.
3) **Configuración:** Se utilizaron configuraciones específicas incluyendo el número de hilos (10 usuarios), tiempo de rampa, y cualquier configuración especial necesaria para replicar el escenario de producción.
4) **Datos de Entrada:** Se utilizaron datos de entrada representativos como clientes ficticios generados al azar (Nombres y Apellidos) para garantizar que la prueba refleje un escenario realista.
5) **Escenario de Error:** Se incluyeron transacciones que generan errores de negocio para simular situaciones no ideales y evaluar cómo el servicio maneja situaciones excepcionales.
6) **Métricas de Interés:** Se monitorearon métricas clave, como tiempos de respuesta, errores, throughput, y uso de recursos, para evaluar el rendimiento del servicio.
7) **Justificación:** Este nivel de concurrencia simula una carga promedio durante horas laborales o momentos de actividad regular. Ayuda a evaluar cómo responde el sistema bajo una carga constante y típica, asegurando que los tiempos de respuesta y el rendimiento general sean aceptables en condiciones habituales.

Diseño de Pruebas

Escenario de Prueba 3: Alta Concurrencia:

Detalles del Escenario:

1) **Carga Simulada:** Se simularon 100 usuarios concurrentes realizando el proceso de **crear y consultar una reserva mediante la API**.
2) **Duración de la Prueba:** La prueba se llevó a cabo durante un periodo de tiempo de una (10) minutos que permitió alcanzar una carga significativa y evaluar el rendimiento de la API para estos dos flujos, bajo condiciones sostenidas.
3) **Configuración:** Se utilizaron configuraciones específicas incluyendo el número de hilos (100 usuarios), tiempo de rampa, y cualquier configuración especial necesaria para replicar el escenario de producción.
4) **Datos de Entrada:** Se utilizaron datos de entrada representativos como clientes ficticios generados al azar (Nombres y Apellidos) para garantizar que la prueba refleje un escenario realista.
5) **Escenario de Error:** Se incluyeron transacciones que generan errores de negocio para simular situaciones no ideales y evaluar cómo el servicio maneja situaciones excepcionales.
6) **Métricas de Interés:** Se monitorearon métricas clave, como tiempos de respuesta, errores, throughput, y uso de recursos, para evaluar el rendimiento del servicio.
7) **Justificación**: Este escenario representa una alta demanda en el sistema, como durante picos de uso o eventos importantes. Es crucial para analizar la capacidad del sistema de manejar cargas máximas, identificar cuellos de botella y evaluar la escalabilidad de la API bajo presión extrema.

Diseño de Pruebas

➢ Herramientas utilizadas

1) **JMeter y K6:** Simulación de carga.
2) **InfluxDB:** Almacenamiento de métricas.
3) **Grafana:** Visualización de métricas.

➢ Aserciones Configuradas

1) Validación de códigos HTTP: Respuestas exitosas: 200 (OK)
2) Verificación de la estructura de la respuesta JSON: Campo **bookingid** presente en la respuesta al crear reservas.

Diseño de Pruebas

➢ Metodología Aplicada en la prueba de: API Restful-Booker

Se define una prueba la cual requiere evaluar la capacidad de la API para manejar diferentes niveles de carga, la prueba esta compuesta por el flujo de crear y consultar una reserva.

Consultar una reserva: Devuelve una reserva específica en función del ID de reserva proporcionado

Crear una reserva: Crea una nueva reserva en la API

Se establece una prueba de carga, en donde se desarrolla el script mediante la herramienta JMeter y K6, aplicando una prueba END to END de acuerdo a lo siguiente:

1. **Poca Concurrencia:** De acuerdo a la justificación dada a este escenario, en esta prueba se simulan 10 usuarios concurrentes. La prueba se tardará un total de 6 minutos, distribuidos así: 1 minutos que se demora la rampa (Ramp-Up- Steps Count) de 5 pasos, para que todos los 10 hilos (usuarios) estén activos y durante ese tiempo se agregaran los usuarios, hasta completar los 10. Después de la rampa, se mantendrá la carga máxima de 10 usuarios concurrentes durante 5 minutos adicionales antes de finalizar la prueba.
2. **Carga moderada:** En esta prueba se simulan 50 usuarios concurrentes. La prueba se tardará un total de 6 minutos, distribuidos así: 1 minutos que se demora la rampa (Ramp-Up- Steps Count) de 5 pasos, para que todos los 50 hilos (usuarios) estén activos y durante ese tiempo se agregaran los usuarios, hasta completar los 50. Después de la rampa, se mantendrá la carga máxima de 50 usuarios concurrentes durante 5 minutos adicionales antes de finalizar la prueba.
3. **Alta Concurrencia:** En esta prueba se simulan 100 usuarios concurrentes. La prueba se tardará un total de 10 minutos, distribuidos así: 5 minutos que se demora la rampa (Ramp-Up- Steps Count) de 10 pasos, para que todos los 100 hilos (usuarios) estén activos y durante ese tiempo se agregaran los usuarios, hasta completar los 100. Después de la rampa, se mantendrá la carga máxima de 100 usuarios concurrentes durante 5 minutos adicionales antes de finalizar la prueba.

➢ Definición de las métricas a monitorear en esta prueba:

Promedio:	Medida empleada para describir un conjunto de usuarios o datos.
Percentil:	Métrica de prueba de performance bajo la cual se encuentra un porcentaje de la muestra y por lo tanto el máximo tiempo de respuesta permitido para el percentil.
Throughput	El valor de throughput se expresa generalmente como solicitudes por segundo (RPS) o transacciones por segundo (TPS). Indica la cantidad de transacciones exitosas completadas en un segundo determinado durante la ejecución de la prueba.

Resultados y Análisis

Resultado prueba escenario 1: Poca concurrencia: Se ejecutaron en total 1328 solicitudes para el flujo de crear y consultar una reserva en la API, en un tiempo total de (6) minutos, con una concurrencia máxima de 10 usuarios, el tiempo de respuesta máximo en el percentil 90 fue de 0,509 milésimas de segundos.
El percentil 90 lo que nos dice en un término simple, es que el 90% de la prueba o de las peticiones respondieron en un tiempo inferior al pico máximo de 0,509 milésimas de segundos definido por la muestra procesada para este percentil.

De acuerdo a lo anterior tenemos la capacidad de recibir en 6 minutos 1328 solicitudes para el flujo de crear y consultar una reserva en la API.

Resultado prueba escenario 2: Carga moderada: Se ejecutaron en total 6690 solicitudes para el flujo de crear y consultar una reserva en la API, en un tiempo total de (6) minutos, con una concurrencia máxima de 50 usuarios, el tiempo de respuesta máximo en el percentil 90 fue de 0,521 milésimas de segundos.
El percentil 90 lo que nos dice en un término simple, es que el 90% de la prueba o de las peticiones respondieron en un tiempo inferior al pico máximo de 0,521 milésimas de segundos definido por la muestra procesada para este percentil.

De acuerdo a lo anterior tenemos la capacidad de recibir en 6 minutos 6690 solicitudes para el flujo de crear y consultar una reserva en la API.

Resultado prueba escenario 3: Alta concurrencia: Se ejecutaron en total 18099 solicitudes para el flujo de crear y consultar una reserva en la API, en un tiempo total de (10) minutos, con una concurrencia máxima de 100 usuarios, el tiempo de respuesta máximo en el percentil 90 fue de 0,636 milésimas de segundos.
El percentil 90 lo que nos dice en un término simple, es que el 90% de la prueba o de las peticiones respondieron en un tiempo inferior al pico máximo de 0,636 milésimas de segundos definido por la muestra procesada para este percentil.

De acuerdo a lo anterior tenemos la capacidad de recibir en 10 minutos 18099 solicitudes para el flujo de crear y consultar una reserva en la API.

Resultados y Análisis

Resultado específicos prueba escenario 1: Poca concurrencia: Tenemos la capacidad de recibir **3,7** solicitudes por segundo de acuerdo al valor tomado de la tabla, en la columna Throughput, En resumen esto nos indica que se ejecutaron **3,7** solicitudes aproximadamente cada segundo, durante el tiempo total de 6 minutos.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
1_Crear_Reservas	1338	388	375	406	416	522	331	4395	0,00%	3,7/sec	3,50	1,74
2_Obtener_Reserva	1331	121	96	106	110	125	81	9100	0,45%	3,7/sec	3,39	0,71
Flujo_Crear_Consultar_Reservas	1328	509	471	510	524	678	419	9493	0,45%	3,7/sec	6,85	2,43
Total	3997	339	375	488	505	578	81	9493	0,30%	11,1/sec	13,67	4,85

Resultado específicos prueba escenario 2: Carga moderada: Tenemos la capacidad de recibir **18,6** solicitudes por segundo de acuerdo al valor tomado de la tabla, en la columna Throughput, En resumen esto nos indica que se ejecutaron **18,6** solicitudes aproximadamente cada segundo, durante el tiempo total de 6 minutos.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	6740	382	377	413	425	489	323	782	0,00%	18,6/sec	17,55	8,70
2_Obtener_Reserva	6711	101	98	110	117	175	78	424	0,00%	18,6/sec	17,02	3,56
Flujo_Crear_Consultar_Reservas	6690	483	476	521	539	635	410	898	0,00%	18,6/sec	34,49	12,23
Total	20141	322	377	495	514	567	78	898	0,00%	55,6/sec	68,74	24,39

Resultado específicos prueba escenario 3: Alta concurrencia: Tenemos la capacidad de recibir **29,8** solicitudes por segundo de acuerdo al valor tomado de la tabla, en la columna Throughput, En resumen esto nos indica que se ejecutaron **29,8** solicitudes aproximadamente cada segundo, durante el tiempo total de 6 minutos.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	18188	420	378	428	585	1354	327	21382	0,20%	30,1/sec	28,47	14,09
2_Obtener_Reserva	18062	132	101	124	259	926	78	8094	0,26%	29,8/sec	27,29	5,72
Flujo_Crear_Consultar_Reservas	18099	540	482	636	948	1486	412	21479	0,46%	29,8/sec	55,39	19,64
Total	54349	364	381	519	687	1340	78	21479	0,31%	89,5/sec	110,92	39,35

Resultados y Análisis

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
1_Crear_Reservas	1338	388	375	406	416	522	331	4395	0,00%	3,7/sec	3,50	1,74
2_Obtener_Reserva	1331	121	96	106	110	125	81	9100	0,45%	3,7/sec	3,39	0,71
Flujo_Crear_Consultar_Reservas	1328	509	471	510	524	678	419	9493	0,45%	3,7/sec	6,85	2,43
Total	3997	339	375	488	505	578	81	9493	0,30%	11,1/sec	13,67	4,85

Análisis de resultado específicos prueba escenario 1: Poca concurrencia:

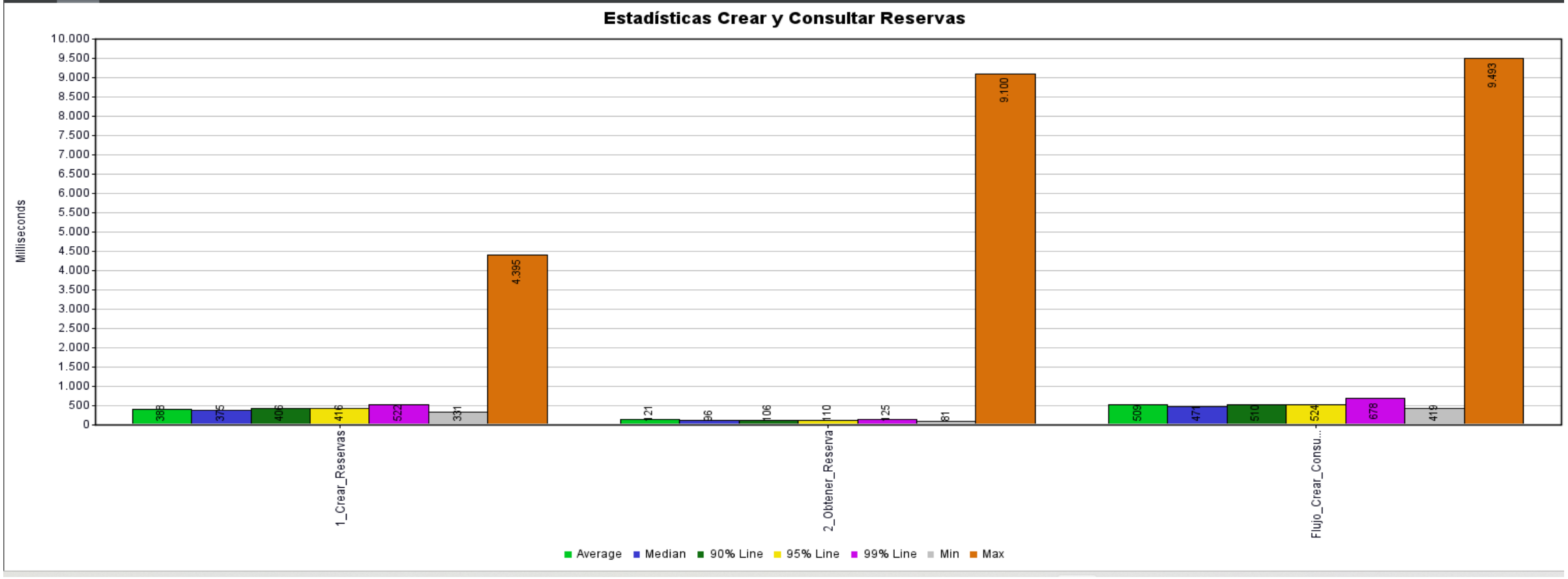
- ✓ #Samples (Muestras): 1328, representa la cantidad de transacciones completadas durante el escenario de prueba. En condiciones de poca carga, se realizaron un total de 1328 solicitudes.
- ✓ Average (Promedio): 509 ms, el tiempo promedio de respuesta es adecuado para una carga baja, pero podría optimizarse, ya que en escenarios de baja concurrencia los tiempos suelen ser más bajos.
- ✓ Median (Mediana): 471 ms, la mitad de las solicitudes tardaron menos de 471 ms, lo cual indica estabilidad, pero existe un desvío al compararlo con el promedio.
- ✓ Percentiles (90%, 95%, 99% Line):90% Line: 510 ms, 95% Line: 524 ms, 99% Line: 678 ms. El 90% de las transacciones respondieron en menos de 510 ms, pero hay una tendencia a mayores tiempos en el extremo superior (678 ms en el 99%). Esto implica que algunas solicitudes tardan más tiempo del esperado.
- ✓ Min: 419 ms, el tiempo mínimo es consistente y razonable para una API de este tipo.
- ✓ Maximum: 9493 ms, un valor atípico (outlier) significativo, que podría estar relacionado con problemas intermitentes en el sistema, como falta de optimización o conexiones inestables.
- ✓ Error %: 0.45%, aunque bajo, refleja fallos que deben analizarse. Los errores probablemente sean causados por solicitudes con parámetros erróneos o fallas del servidor.
- ✓ Throughput: 3.7 transacciones/segundo, la capacidad del sistema es aceptable en carga baja.
- ✓ Received KB/sec & Sent KB/sec: 6.85 / 2.43 KB, estos valores son proporcionales al Throughput y reflejan un consumo de datos moderado en el escenario.

Resultados y Análisis

Gráficos escenario 1: Poca concurrencia:

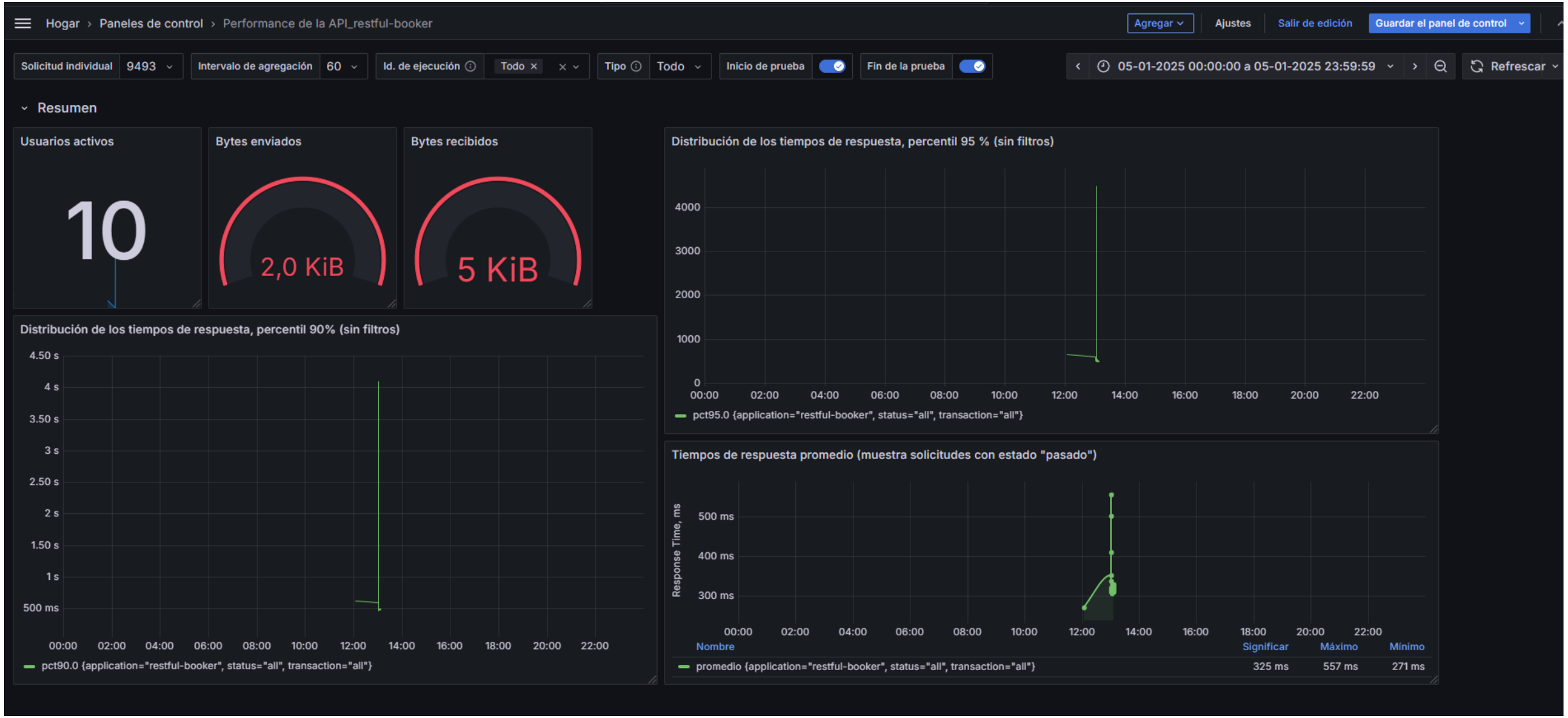
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
1_Crear_Reservas	1338	388	375	406	416	522	331	4395	0,00%	3,7/sec	3,50	1,74
2_Obtener_Reserva	1331	121	96	106	110	125	81	9100	0,45%	3,7/sec	3,39	0,71
Flujo_Crear_Consultar_Reservas	1328	509	471	510	524	678	419	9493	0,45%	3,7/sec	6,85	2,43
Total	3997	339	375	488	505	578	81	9493	0,30%	11,1/sec	13,67	4,85

Settings Graph



Resultados y Análisis

Gráficos escenario 1: Poca concurrencia:



Resultados y Análisis

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	6740	382	377	413	425	489	323	782	0,00%	18,6/sec	17,55	8,70
2_Obtener_Reserva	6711	101	98	110	117	175	78	424	0,00%	18,6/sec	17,02	3,56
Flujo_Crear_Consultar_Reservas	6690	483	476	521	539	635	410	898	0,00%	18,6/sec	34,49	12,23
Total	20141	322	377	495	514	567	78	898	0,00%	55,6/sec	68,74	24,39

Resultado específicos prueba escenario 2: Carga moderada:

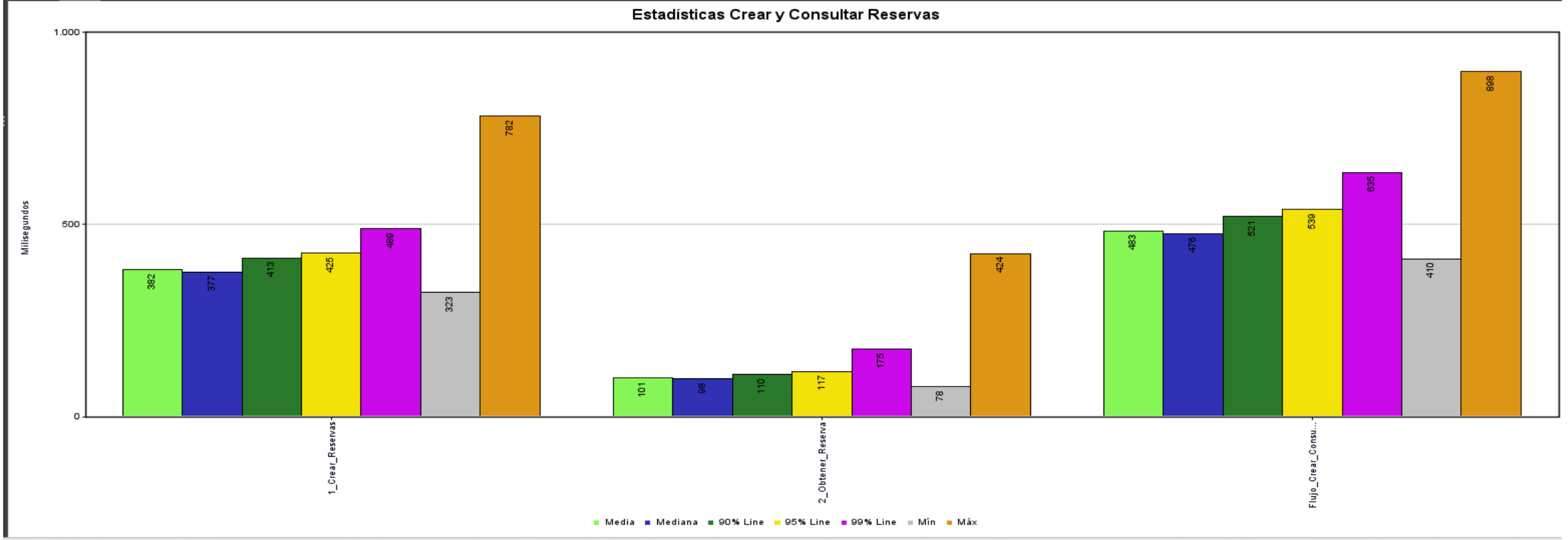
- ✓ #Samples (Muestras): 6690, un volumen significativamente mayor de transacciones, que refleja condiciones de uso promedio del sistema.
- ✓ Average (Promedio): 483 ms, el tiempo promedio de respuesta es mejor que en la carga baja, lo cual indica que el sistema maneja mejor un nivel de concurrencia moderado. Esto podría ser resultado de optimizaciones en el manejo de solicitudes concurrentes.
- ✓ Median (Mediana): 476 ms, el 50% de las solicitudes se completaron en menos de 476 ms. Es un valor muy cercano al promedio, lo que indica que el sistema es más consistente bajo esta carga.
- ✓ Percentiles (90%, 95%, 99% Line):90% Line: 521 ms, 95% Line: 539 ms, 99% Line: 635 ms. Las solicitudes más lentas aún están dentro de un rango razonable (635 ms en el 99%), mostrando que el sistema mantiene su estabilidad bajo condiciones moderadas.
- ✓ Min: 410 ms, el tiempo mínimo es consistente con los otros escenarios.
- ✓ Maximum: 898 ms, la ausencia de outliers extremos en esta prueba refuerza la idea de que el sistema es más estable en esta carga.
- ✓ Error %: 0.0%, no se registraron errores en este nivel de concurrencia, lo que demuestra la robustez del sistema bajo carga media.
- ✓ Throughput: 18.6 transacciones/segundo, la capacidad del sistema aumentó significativamente, mostrando que maneja esta carga sin comprometer el rendimiento.
- ✓ Received KB/sec & Sent KB/sec: 34.49 / 12.23 KB, el tráfico de datos también aumentó proporcionalmente al Throughput, pero sigue siendo manejable.

Resultados y Análisis

Gráficos escenario 2: Carga moderada:

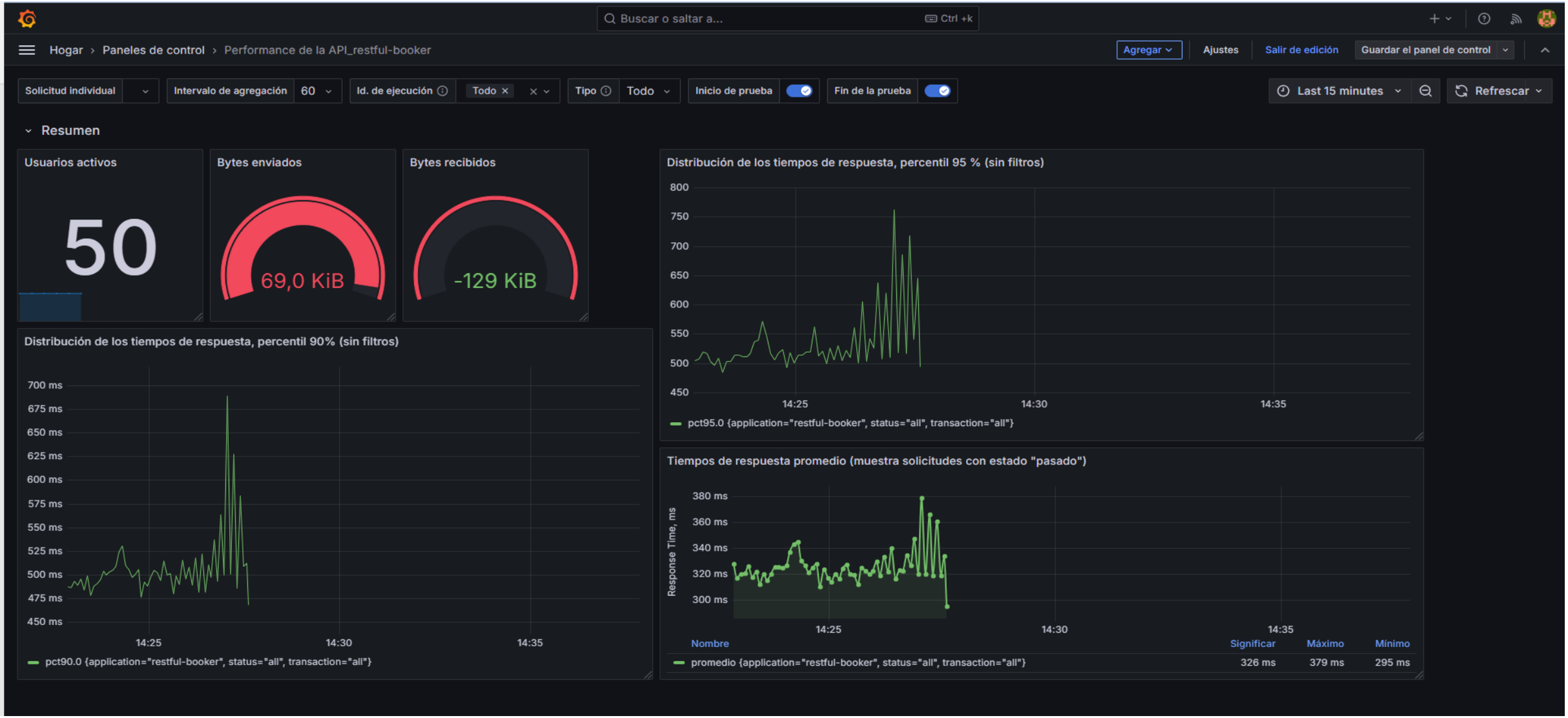
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	6740	382	377	413	425	489	323	782	0,00%	18,6/sec	17,55	8,70
2_Obtener_Reserva	6711	101	98	110	117	175	78	424	0,00%	18,6/sec	17,02	3,56
Flujo_Crear_Consultar_Reservas	6690	483	476	521	539	635	410	898	0,00%	18,6/sec	34,49	12,23
Total	20141	322	377	495	514	567	78	898	0,00%	55,6/sec	68,74	24,39

Settings Graph



Resultados y Análisis

Gráficos escenario 2: Carga moderada:



Resultados y Análisis

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	18188	420	378	428	585	1354	327	21382	0,20%	30,1/sec	28,47	14,09
2_Obtener_Reserva	18062	132	101	124	259	926	78	8094	0,26%	29,8/sec	27,29	5,72
Flujo_Crear_Consultar_Reservas	18099	540	482	636	948	1486	412	21479	0,46%	29,8/sec	55,39	19,64
Total	54349	364	381	519	687	1340	78	21479	0,31%	89,5/sec	110,92	39,35

Resultado específicos prueba escenario 3: Alta concurrencia:

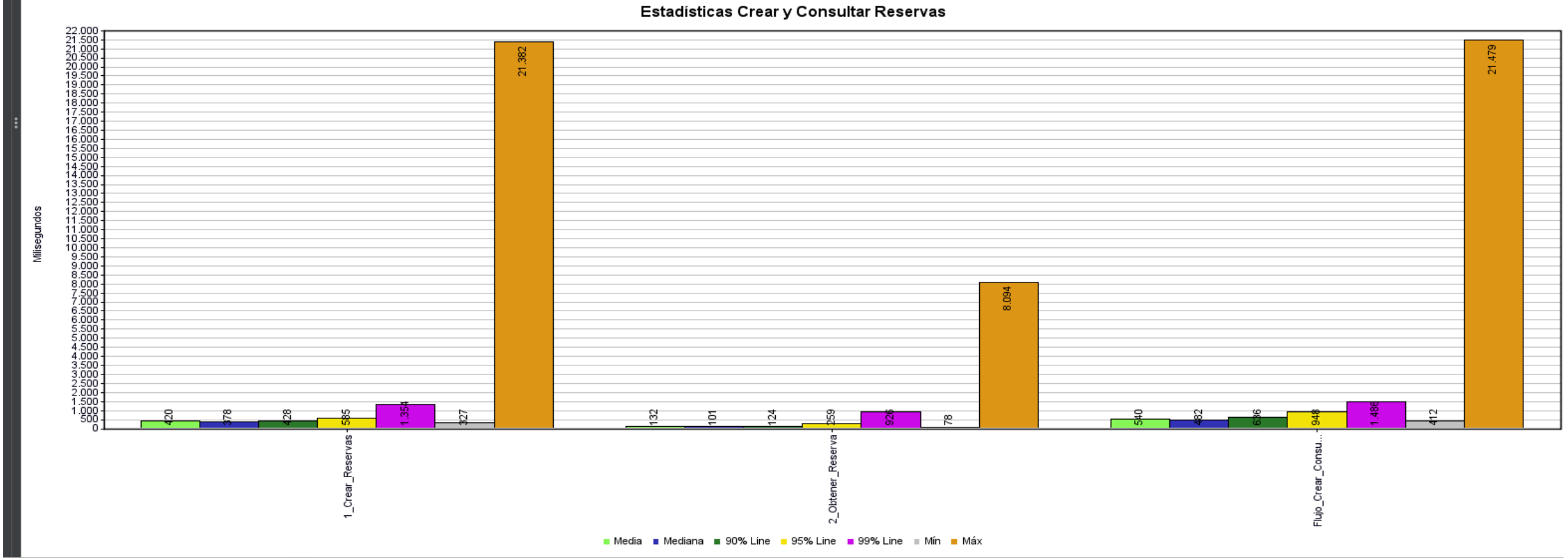
- ✓ #Samples (Muestras): 18099, este volumen elevado de transacciones simula una carga máxima o pico en el sistema.
- ✓ Average (Promedio): 540 ms, el tiempo promedio de respuesta aumentó en comparación con los otros escenarios, pero sigue siendo razonable considerando la alta concurrencia.
- ✓ Median (Mediana): 482 ms, la mediana muestra que la mayoría de las solicitudes se procesan rápidamente. Sin embargo, existe una mayor variación en los tiempos de respuesta bajo esta carga.
- ✓ Percentiles (90%, 95%, 99% Line):90% Line: 636 ms, 95% Line: 948 ms, 99% Line: 1486 ms. Se observa un incremento significativo en los tiempos de respuesta para las solicitudes más lentas, especialmente en el percentil 95 y 99. Esto podría indicar saturación del sistema o cuellos de botella bajo alta concurrencia.
- ✓ Min: 412 ms, el tiempo mínimo se mantiene estable, lo que demuestra que el sistema puede procesar algunas solicitudes rápidamente incluso en condiciones de alta carga.
- ✓ Maximum: 21479 ms, un outlier extremo que sugiere problemas serios bajo alta concurrencia, como saturación de recursos o tiempos de espera excesivos.
- ✓ Error %: 0.46%, aunque bajo, este porcentaje refleja errores que deben analizarse. Los errores probablemente se incrementen debido a la saturación del sistema o a la imposibilidad de procesar algunas solicitudes.
- ✓ Throughput: 29.8 transacciones/segundo, el sistema muestra una buena capacidad para manejar solicitudes, pero el aumento en los tiempos de respuesta y errores indica que posiblemente está llegando a su límite.
- ✓ Received KB/sec & Sent KB/sec: 55.39 / 19.64 KB, el tráfico de datos es alto, lo cual es esperado bajo alta concurrencia.

Resultados y Análisis

Gráficos escenario 3: Alta concurrencia:

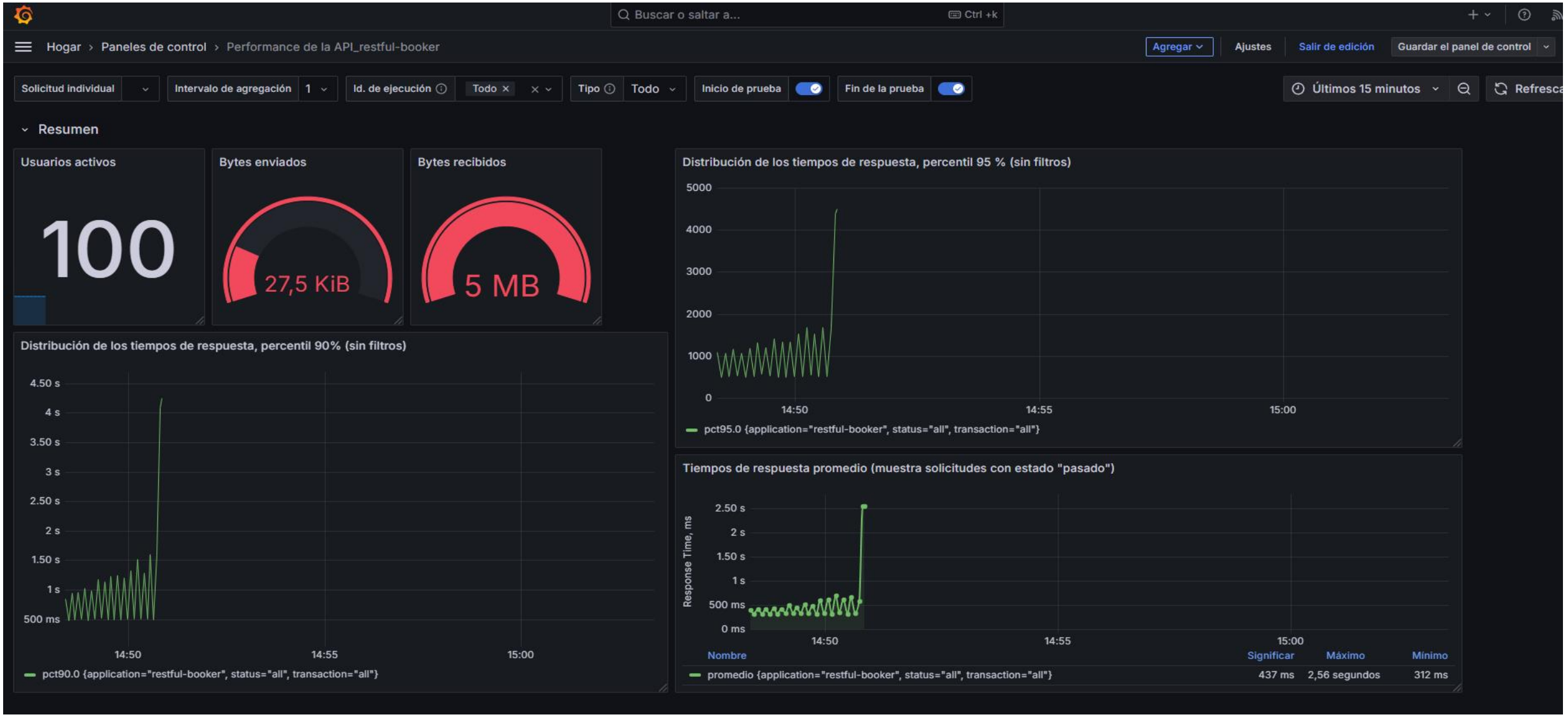
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
1_Crear_Reservas	18188	420	378	428	585	1354	327	21382	0,20%	30,1/sec	28,47	14,09
2_Obtener_Reserva	18062	132	101	124	259	926	78	8094	0,26%	29,8/sec	27,29	5,72
Flujo_Crear_Consultar_Reservas	18099	540	482	636	948	1486	412	21479	0,46%	29,8/sec	55,39	19,64
Total	54349	364	381	519	687	1340	78	21479	0,31%	89,5/sec	110,92	39,35

Settings Graph



Resultados y Análisis

Gráficos escenario 3: Alta concurrencia:



Resultados y Análisis

Resultados obtenidos desde K6:

Escenarios de prueba	Resultados desde K6						
	#. Peticiones	Throughput (Rendimiento)	Máximos (Max Response Time)	Percentil 90	Percentil 95	Data received	Data sent
Escenario1: Poca Concurrencia: 10 Usuarios	5290	14,6/sec	402,95 ms	104,45ms	108,39ms	15 kb/s	4.4 kb/s
Escenario2: Carga Moderada: 50 Usuarios	26304	72,81/sec	1,39 ms	131,28ms	173,61ms	76 kb/s	22 kb/s
Escenario3: Alta Concurrencia: 100 Usuarios	73136	121,63/sec	9,1 s	218,34ms	534,53ms	127 kb/s	37 kb/s

Observaciones: En cuanto a la diferencia de K6 en el número de Peticiones y Throughput la cual fue mas elevada que las de Jmeter, cabe tener en cuenta lo siguiente:

- ✓ **Eficiencia del motor de ejecución de K6 frente a Jmeter:** K6 es conocido por su motor de ejecución altamente optimizado y ligero. Está diseñado para maximizar el uso del hardware disponible, con un enfoque especial en el rendimiento, utilizando Go como lenguaje base. Esto permite a K6 manejar solicitudes HTTP de forma más eficiente, generando un mayor número de solicitudes por segundo en comparación con JMeter, que funciona sobre la JVM (Java Virtual Machine). La JVM consume más memoria y recursos debido a su naturaleza de ejecución, lo que puede limitar el número de solicitudes procesadas en un entorno similar. Por eso, K6 tiende a registrar más solicitudes y un mayor throughput bajo las mismas condiciones.
- ✓ **Diferencia en la medición del Throughput:**
  - **En JMeter:** El Throughput se calcula como el número de solicitudes completadas en un período de tiempo, basado en la duración total del escenario, incluyendo los tiempos de espera (think time) y la latencia de red entre solicitudes.
  - **En K6:** El Throughput (http_reqs/s) es más directo y optimizado. K6 tiende a aprovechar al máximo la capacidad de los usuarios y no incluye tiempos muertos entre iteraciones, a menos que se especifiquen explícitamente con sleep(). Esto puede dar lugar a un mayor número de solicitudes por segundo.
- ✓ **Recursos del sistema y hardware:** Aunque los scripts son idénticos, el rendimiento puede variar según el hardware y los recursos del sistema utilizados:
  - K6 es más ligero y aprovecha mejor los recursos de la CPU y memoria.
  - JMeter, al ejecutarse en la JVM, puede requerir más memoria y procesamiento, lo que puede limitar el número de solicitudes enviadas.
  - Si se ejecutaron ambos scripts en la misma máquina, es probable que K6 haya utilizado los recursos de manera más eficiente, permitiendo un mayor número de solicitudes.
- ✓ **Conclusión General:** La diferencia en el número de solicitudes y throughput entre K6 y JMeter se debe principalmente a la eficiencia del motor de ejecución de K6 y su capacidad para manejar conexiones HTTP y tiempos de espera de manera más optimizada. Aunque ambos scripts están configurados de manera idéntica, K6 logra un mayor rendimiento gracias a su arquitectura ligera basada en eventos y su uso eficiente del hardware. Esto no significa que los resultados de JMeter sean incorrectos, sino que reflejan las limitaciones inherentes de su arquitectura basada en threads y la JVM. Si el objetivo es comparar el rendimiento real del sistema bajo prueba, es importante interpretar los resultados de cada herramienta teniendo en cuenta estas diferencias en la metodología de ejecución y medición.

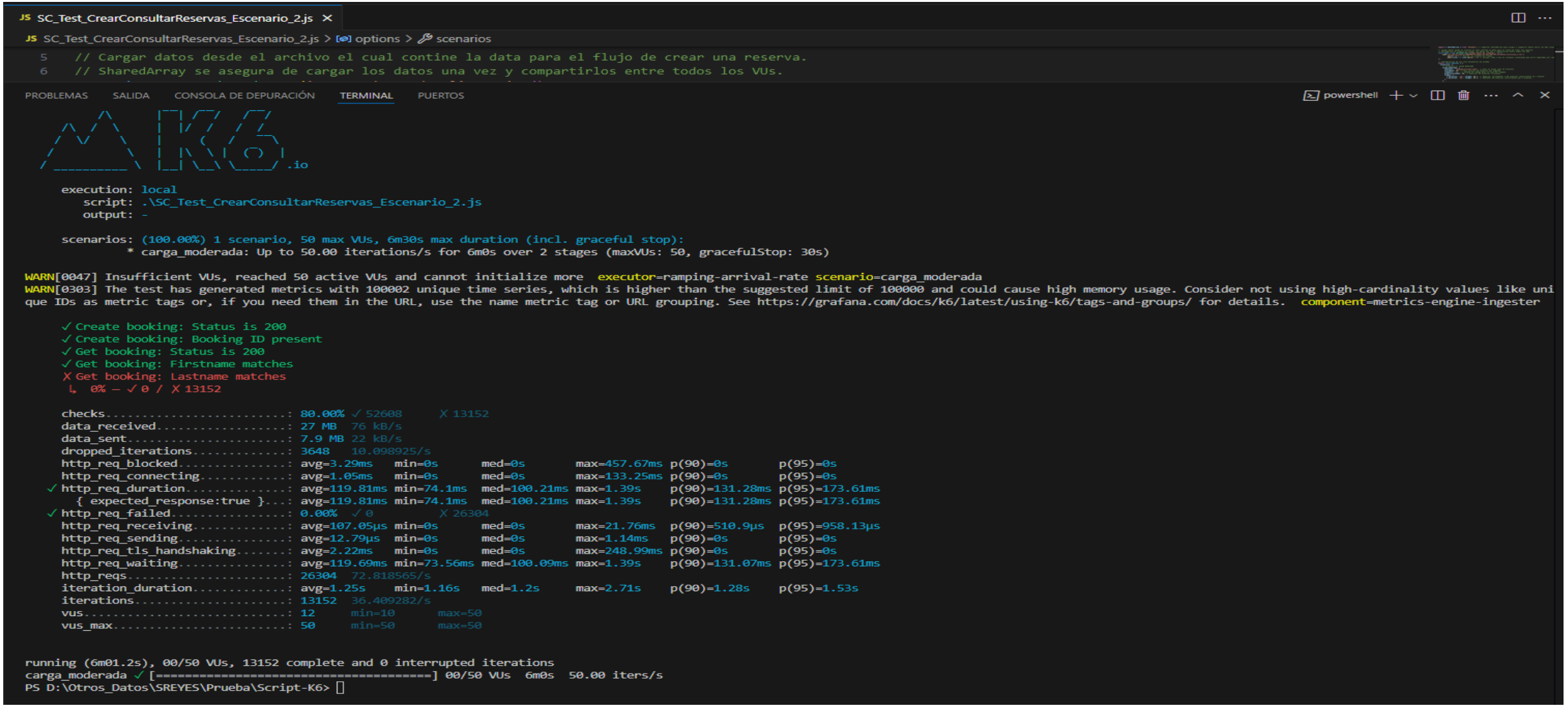
Resultados y Análisis

Resultados desde K6, Escenario1: Poca Concurrencia: 10 Usuarios:

```
          /\      |‾‾| /‾‾/   /‾‾/
     /\  /  \     |  |/  /   /  /
    /  \/    \    |     (   /   ‾‾\
   /          \   |  |\  \ |  (‾)  |
  / __________ \  |__| \__\ \_____/ .io

     execution: local
        script: .\Copia-SC_Test_crearConsultarReservas.js
        output: -

     scenarios: (100.00%) 1 scenario, 10 max VUs, 6m30s max duration (incl. graceful stop):
              * poca_concurrencia: Up to 10.00 iterations/s for 6m0s over 2 stages (maxVUs: 10, gracefulStop: 30s)

WARN[0048] Insufficient VUs, reached 10 active VUs and cannot initialize more  executor=ramping-arrival-rate scenario=poca_concurrencia

     ✓ Create booking: Status is 200
     ✓ Create booking: Booking ID present
     ✓ Get booking: Status is 200
     ✓ Get booking: Firstname matches
     ✗ Get booking: Lastname matches
      ↳  0% — ✓ 0 / ✗ 2645

     checks.....................: 80.00% ✓ 10580      ✗ 2645
     data_received..............: 5.5 MB 15 kB/s
     data_sent..................: 1.6 MB 4.4 kB/s
     dropped_iterations.........: 715    1.979344/s
     http_req_blocked...........: avg=3.22ms   min=0s      med=0s      max=465.11ms p(90)=0s       p(95)=0s
     http_req_connecting........: avg=1.04ms   min=0s      med=0s      max=111.27ms p(90)=0s       p(95)=0s
   ✓ http_req_duration..........: avg=96.4ms   min=80.38ms med=94.66ms max=402.95ms p(90)=104.46ms p(95)=108.39ms
       { expected_response:true }: avg=96.4ms   min=80.38ms med=94.66ms max=402.95ms p(90)=104.46ms p(95)=108.39ms
   ✓ http_req_failed............: 0.00%  ✓ 0          ✗ 5290
     http_req_receiving.........: avg=115.73µs min=0s      med=0s      max=1.58ms   p(90)=530.62µs p(95)=999.5µs
     http_req_sending...........: avg=25.36µs  min=0s      med=0s      max=1.25ms   p(90)=0s       p(95)=103.59µs
     http_req_tls_handshaking...: avg=2.13ms   min=0s      med=0s      max=219.38ms p(90)=0s       p(95)=0s
     http_req_waiting...........: avg=96.26ms  min=80.38ms med=94.57ms max=402.95ms p(90)=104.36ms p(95)=108.28ms
     http_reqs..................: 5290   14.644377/s
     iteration_duration.........: avg=1.2s     min=1.17s   med=1.19s   max=1.69s    p(90)=1.21s    p(95)=1.23s
     iterations.................: 2645   7.322188/s
     vus........................: 3      min=2        max=10
     vus_max....................: 10     min=10       max=10


running (6m01.2s), 00/10 VUs, 2645 complete and 0 interrupted iterations
poca_concurrencia ✓ [======================================] 00/10 VUs  6m0s  10.00 iters/s
PS D:\Otros_Datos\SREYES\Prueba\Script-K6>
```

Resultados y Análisis

Resultados desde K6, Escenario2: Carga Moderada: 50 Usuarios:



Resultados y Análisis

Resultados desde K6, Escenario3: Alta Concurrencia: 100 Usuarios:

```
JS SC_Test_CrearConsultarReservas_Escenario_3.js
JS SC_Test_CrearConsultarReservas_Escenario_3.js > options
35  export default function () {
54    // Definir los headers para las solicitudes HTTP.
55    const headers = {
56      'Content-Type': 'application/json', // Tipo de contenido en JSON.
57      Accept: 'application/json', // Especificar que se espera una respuesta en JSON.
58    };
59
60    // Solicitud POST para crear una reserva.
61    const createResponse = http.post(
62      'https://restful-booker.herokuapp.com/booking', // URL del endpoint para crear una reserva.

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS                    powershell

running at parse (native)
alta_conat Get booking: Firstname matches (file:///D:/Otros_Datos/SREYES/Prueba/Script-K6/SC_Test_CrearConsultarReservas_Escenario_3.js:87:17(5))
        at go.k6.io/k6/js/modules/k6.(*K6).Check-fm (native)
        at file:///D:/Otros_Datos/SREYES/Prueba/Script-K6/SC_Test_CrearConsultarReservas_Escenario_3.js:84:8(103)  executor=ramping-arrival-rate scenario=alta_concurrencia source=stacktrace

     X Create booking: Status is 200
      ↳  99% — ✓ 36560 / X 16
     X Create booking: Booking ID present
      ↳  99% — ✓ 36560 / X 16
     X Get booking: Status is 200
      ↳  99% — ✓ 36541 / X 19
     X Get booking: Firstname matches
      ↳  99% — ✓ 36541 / X 19
     X Get booking: Lastname matches
      ↳  0% — ✓ 0 / X 36541

     checks.....................: 79.97% ✓ 146202      X 36611
     data_received..............: 76 MB  127 kB/s
     data_sent..................: 22 MB  37 kB/s
     dropped_iterations.........: 11423  18.997226/s
     http_req_blocked...........: avg=3.08ms   min=0s       med=0s       max=460.35ms p(90)=0s       p(95)=0s
     http_req_connecting........: avg=973.39µs min=0s       med=0s       max=119.89ms p(90)=0s       p(95)=0s
   ✓ http_req_duration..........: avg=181.01ms min=73.84ms  med=102.08ms max=9.1s     p(90)=218.34ms p(95)=534.53ms
       { expected_response:true }: avg=180.16ms min=73.84ms  med=102.07ms max=6.1s     p(90)=217.13ms p(95)=532.83ms
   ✓ http_req_failed............: 0.04%  ✓ 35          X 73101
     http_req_receiving.........: avg=116.35µs min=0s       med=0s       max=19.91ms  p(90)=526.2µs  p(95)=957.3µs
     http_req_sending...........: avg=8.77µs   min=0s       med=0s       max=1.53ms   p(90)=0s       p(95)=0s
     http_req_tls_handshaking...: avg=2.1ms    min=0s       med=0s       max=243.99ms p(90)=0s       p(95)=0s
     http_req_waiting...........: avg=180.88ms min=73.84ms  med=101.95ms max=9.1s     p(90)=218.23ms p(95)=534.33ms
     http_reqs..................: 73136  121.630142/s
     iteration_duration.........: avg=1.37s    min=143.92ms med=1.21s    max=9.5s     p(90)=1.53s    p(95)=2.43s
     iterations.................: 36576  60.828375/s
     vus........................: 35     min=19       max=100
     vus_max....................: 100    min=100      max=100


running (10m01.3s), 000/100 VUs, 36576 complete and 0 interrupted iterations
alta_concurrencia ✓ [======================================] 000/100 VUs  10m0s  100.00 iters/s
PS D:\Otros_Datos\SREYES\Prueba\Script-K6>
```

Conclusiones

- **Conclusiones en cuanto a los resultados de la prueba:**
  - ✓ **Poca Concurrencia:** El sistema funciona correctamente, pero los tiempos promedio y máximos indican que existen ciertos problemas que necesitan ser abordados, como la gestión de outliers.
  - ✓ **Carga Moderada:** El sistema es más estable y maneja esta carga sin errores ni problemas significativos. Este escenario muestra el rendimiento óptimo del sistema.
  - ✓ **Alta Concurrencia:** Aunque el sistema maneja una cantidad significativa de solicitudes, los tiempos de respuesta más altos y el aumento en los errores indican la necesidad de optimizar el rendimiento y la capacidad del sistema bajo alta carga.
  - ✓ **Throughput (Rendimiento**): El throughput aumenta proporcionalmente con la carga:
    - Poca Concurrencia: 3.7/sec.
    - Carga Moderada: 18.6/sec.
    - Alta Concurrencia: 29.8/sec.

    Este comportamiento sugiere que el sistema está bien dimensionado para manejar cargas crecientes, pero se observan limitaciones en alta concurrencia, especialmente con los tiempos de respuesta más altos y errores.
  - ✓ **Máximos (Max Response Time):** Los tiempos máximos aumentan significativamente con la carga:
    - Poca Concurrencia: 9493 ms (outlier (Valor Atípico) notable).
    - Carga Moderada: 898 ms (muy controlado).
    - Alta Concurrencia: 21479 ms (indicativo de saturación severa).

    Los tiempos máximos altos en poca y alta concurrencia indican que el sistema tiene dificultades intermitentes al procesar algunas transacciones, posiblemente por saturación de recursos en alta carga o por problemas de diseño en cargas bajas.
  - ✓ **Percentiles (90%, 95%, 99%):** Tendencia
    - Poca Concurrencia: Los percentiles 90% y 95% están cerca del promedio (510 ms y 524 ms), pero el 99% muestra un incremento significativo (678 ms), lo que sugiere pocas solicitudes problemáticas.
    - Carga Moderada: Los percentiles son más consistentes (521 ms, 539 ms, 635 ms), mostrando un sistema estable.
    - Alta Concurrencia: Los percentiles se disparan (636 ms, 948 ms, 1486 ms), lo que indica un comportamiento menos predecible a medida que la carga aumenta.

    Esto indica que el sistema es más estable bajo carga moderada y tiene comportamientos inesperados bajo alta concurrencia.
  - ✓ **Promedio vs Mediana:**
    - La mediana en los tres escenarios está consistentemente cerca del promedio, lo que indica una distribución de tiempos de respuesta mayoritariamente uniforme, excepto en los outliers.
    - En alta concurrencia, sin embargo, la diferencia entre promedio (540 ms) y percentil 99% (1486 ms) sugiere que los tiempos de respuesta de las solicitudes más lentas tienen un impacto más significativo.

Recomendaciones

- **Recomendaciones generales:**
  - Los resultados en los tiempos de respuesta pueden variar respecto a la cantidad de solicitudes en este flujo.
  - Se sugiere mantener un monitoreo constante, especialmente en entornos de producción, para garantizar un rendimiento continuo y detectar cualquier problema potencial a medida que evoluciona la carga.
  - Asegúrese de realizar una revisión continua del rendimiento, especialmente cuando se introduzcan cambios en el sistema o se incrementen las cargas esperadas. Además, considere la implementación de medidas proactivas, como el escalado automático de recursos, para mantener un rendimiento óptimo en entornos dinámicos.
- **Recomendaciones especificas:**

✓ **Poca Concurrencia:**
- ➢ **Análisis**: Los tiempos máximos altos (9493 ms) y un porcentaje de error del 0.45% indican que el sistema podría no estar completamente optimizado incluso en baja carga.
- ➢ **Recomendaciones**:
  - ❖ Revisar las transacciones con tiempos de respuesta máximos para identificar posibles problemas de diseño o consultas ineficientes.
  - ❖ Optimizar las consultas o servicios que generen solicitudes con tiempos de respuesta altos.
  - ❖ Analizar las solicitudes que fallaron para evitar el error del 0.45%.

✓ **Carga Moderada:**
- ➢ **Análisis**: El sistema muestra un rendimiento óptimo en este escenario con tiempos de respuesta consistentes y sin errores.
- ➢ **Recomendaciones**:
- ➢ Usar este escenario como referencia para establecer el nivel óptimo de carga del sistema.
- ➢ Continuar monitoreando y optimizando para mantener la estabilidad ante picos de carga moderada.

✓ **Alta Concurrencia:**
- ➢ **Análisis**: El sistema experimenta saturación, con tiempos máximos elevados (21479 ms), un aumento en los tiempos de respuesta de los percentiles 90%-99%, y un 0.46% de errores.
- ➢ **Recomendaciones**:
  - ❖ Identificar cuellos de botella en los recursos del servidor, como CPU, memoria o base de datos (Optimización de consultas).
  - ❖ Revisar las solicitudes que fallaron (0.46%) para reducir el impacto de errores.
  - ❖ Realizar ajustes en los servicios más críticos, priorizando la reducción de tiempos en el percentil 99%.

✓ **Optimización de Tiempos:** Analizar las transacciones que están tomando más tiempo del esperado y optimizar el código o la lógica de negocio asociada.

✓ **Monitoreo Continuo:** Implementar monitoreo proactivo para identificar y diagnosticar posibles problemas antes de que afecten al usuario final. Herramientas como APM (Application Performance Monitoring) podrían ser útiles para rastrear solicitudes lentas en tiempo real.

✓ **Capacidad del Sistema:** Evaluar si el rendimiento actual es suficiente para los requisitos del sistema. Puede ser necesario escalar recursos o realizar ajustes en la arquitectura.

✓ **Pruebas con mayor concurrencia:** Aumentar la cantidad de usuarios concurrentes en pruebas futuras para evaluar el comportamiento del sistema bajo cargas mayores. Esto ayudará a identificar puntos de saturación y a garantizar que el rendimiento se mantenga estable.

✓ **Margen de Mejora:** Si bien el sistema cumple, siempre es beneficioso tener un margen de mejora. Podrías considerar optimizaciones adicionales para aumentar el throughput y proporcionar un margen adicional de capacidad.

Anexos

- ✓ **Scripts de JMeter** (SC_CrearConsultarReservas.jmx) **y de K6** (SC_Test_crearConsultarReservas_Escenario_1.js, SC_Test_crearConsultarReservas_Escenario_2.js, SC_Test_crearConsultarReservas_Escenario_3.js y SC_Test_CrearConsultarReservas_TodosEscenarios.js).
- ✓ **Capturas de pantallas de resultados** (Carpeta Resultados/Desde_K6 y Desde_JMeter)**.**
- ✓ **Captura de pantalla de la concurrencia** (Carpeta Resultados/Desde_JMeter), para las pruebas en K6 se utilizo la misma concurrencia en cada escenario**.**
- ✓ **Logs .csv de resultados desde JMeter** (Carpeta Resultados/Desde_JMeter)**.**