Información del Caso de Prueba 02

ID: TCP-CARGA-002

Versión: 1.0
Fecha: 2 Diciembre 2024
Autor: Equipo QA
Prioridad: Alta

1. Descripción General

Este caso de prueba evalúa la capacidad del sistema para manejar múltiples envíos de mensajes de chat simultáneamente en diferentes salas, verificando tiempos de respuesta, consumo de recursos, manejo de errores, y la consistencia de los datos almacenados en PostgreSQL. Además, se evalúa el impacto del tráfico en el frontend desarrollado en Angular.

2. Precondiciones

- Ambiente de pruebas configurado:
  - API de backend desplegada y accesible en un entorno aislado.
  - Base de datos PostgreSQL configurada con acceso permitido desde el backend.
  - Angular frontend desplegado y accesible en modo producción.
- Herramientas configuradas:
  - Locust o K6 para simular múltiples envíos de mensajes.
  - Grafana + Prometheus para monitoreo en tiempo real del sistema.
- Configuración de prueba:
  - Máximo de mensajes concurrentes: 10,000.
  - Incremento gradual de mensajes por segundo.

3. Datos de Prueba

```
const chatMessages = [
   {
      id: 1,
      chatRoom: { id: 1, name: 'Sala de Pruebas', createdAt: new Date() },
      user: { id: 1, name: 'Usuario1', email: 'usuario1@ejemplo.com' },
      message: 'Mensaje de prueba 1',
      createdAt: new Date(),
      unread: false
   },
   {
      id: 2,
      chatRoom: { id: 1, name: 'Sala de Pruebas', createdAt: new Date() },
      user: { id: 2, name: 'Usuario2', email: 'usuario2@ejemplo.com' },
      message: 'Mensaje de prueba 2',
      createdAt: new Date(),
      unread: false
   }
   // Repetir con usuarios y salas de prueba adicionales
];
```

4. Script de Automatización

Usando K6 para Pruebas de Carga

```
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  stages: [
    { duration: '1m', target: 100 }, // 100 mensajes/s
    { duration: '5m', target: 500 }, // Incremento a 500 mensajes/s
    { duration: '2m', target: 1000 } // Pico de 1000 mensajes/s
  ],
};

export default function () {
  const url = 'https://api.chat-system.test/api/chat-rooms/1/messages';
  const payload = JSON.stringify({
    id: Math.floor(Math.random() * 100000),
    chatRoom: { id: 1 },
    user: { id: Math.floor(Math.random() * 1000), name: 'UsuarioPrueba' },
    message: `Mensaje de prueba ${Math.random()}`,
    createdAt: new Date().toISOString()
  });

  const params = { headers: { 'Content-Type': 'application/json' } };

  const res = http.post(url, payload, params);
  check(res, { 'is status 200': (r) => r.status === 200 });
  sleep(1);
}
```

Ejecución del script: k6 run prueba_carga_mensajes.js

5. Pasos de Prueba Manual

1. Acceder al frontend en Angular y autenticar a varios usuarios en diferentes navegadores.
2. Enviar mensajes desde cada usuario en intervalos de 1 segundo.
3. Monitorear la interfaz para detectar problemas como retrasos, duplicación de mensajes o errores.
4. Verificar la base de datos PostgreSQL para asegurar que los mensajes se almacenaron correctamente y sin inconsistencias.
5. Observar los logs del backend y el rendimiento del servidor.

6. Métricas de Prueba

Criterio	Métrica	Estado
Tiempo de respuesta máximo	≤ 200 ms por mensaje	[ ]
Errores HTTP	≤ 0.5% en solicitudes	[ ]
Mensajes procesados	≥ 5,000 mensajes/s en promedio	[ ]
Consistencia de datos	Sin duplicados ni omisiones	[ ]
Uso de recursos	CPU, memoria, y disco ≤ 80%	[ ]

7. Herramientas Recomendadas

1. K6

- **Uso:** Pruebas de carga con scripts basados en JavaScript.
- **Métricas:** Tiempos de respuesta, tasas de error, concurrencia.

2. Locust

- **Uso:** Simular usuarios concurrentes que envían mensajes y observan respuestas.
- **Métricas:** Solicitudes por segundo, tiempos de respuesta promedio, tasa de errores.

3. Grafana + Prometheus

- **Uso:** Monitoreo en tiempo real del sistema y base de datos PostgreSQL.
- **Métricas:** Uso de CPU, memoria, y disco; consultas por segundo en PostgreSQL.

4. Postgres Exporter

- **Uso:** Complemento para monitorear métricas de PostgreSQL como latencia de consultas y conexiones activas.
- **Métricas:** Latencia de escrituras, bloqueos en tablas, transacciones activas.

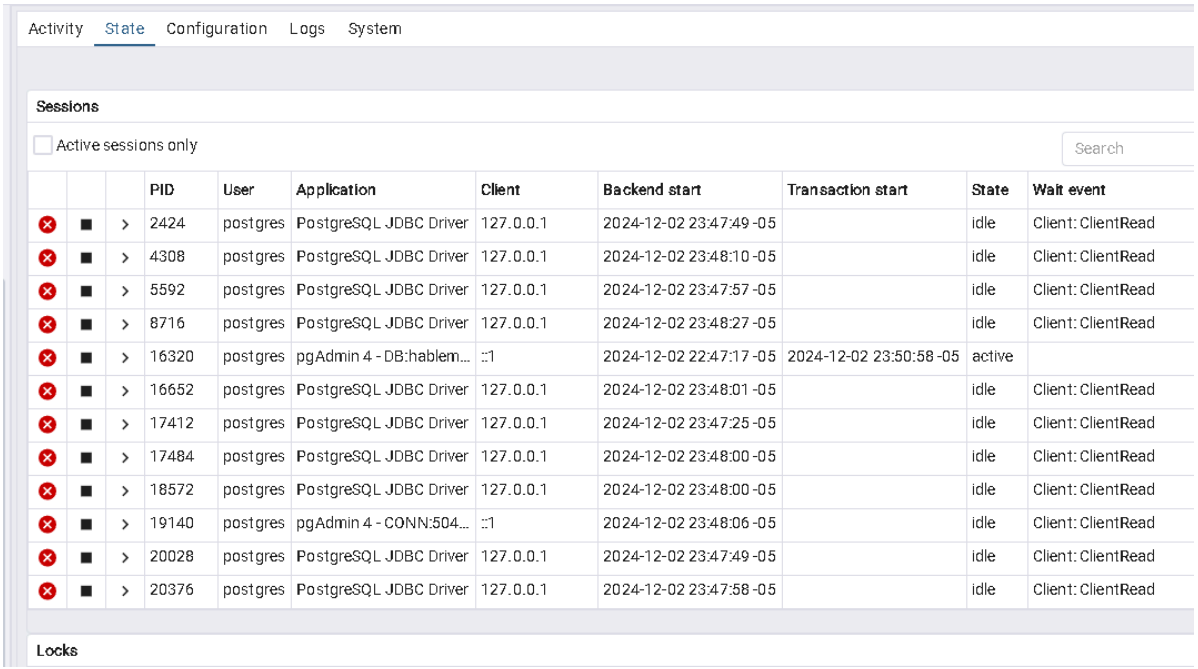
8. Resultados Esperados

Carga máxima de 10,000 mensajes/s:

- Tiempos de respuesta ≤ 200 ms.
- Consistencia de datos: todos los mensajes deben almacenarse correctamente.

Impacto en PostgreSQL:

- Sin bloqueos ni tiempos de espera prolongados.
- Consultas concurrentes ≤ 80% del límite máximo permitido.



			PID	User	Application	Client	Backend start	Transaction start	State	Wait event
			2424	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:49 -05		idle	Client: ClientRead
			4308	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:10 -05		idle	Client: ClientRead
			5592	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:57 -05		idle	Client: ClientRead
			8716	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:27 -05		idle	Client: ClientRead
			16320	postgres	pgAdmin 4 - DB:hablem...	::1	2024-12-02 22:47:17 -05	2024-12-02 23:50:58 -05	active	
			16652	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:01 -05		idle	Client: ClientRead
			17412	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:25 -05		idle	Client: ClientRead
			17484	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:00 -05		idle	Client: ClientRead
			18572	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:00 -05		idle	Client: ClientRead
			19140	postgres	pgAdmin 4 - CONN:504...	::1	2024-12-02 23:48:06 -05		idle	Client: ClientRead
			20028	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:49 -05		idle	Client: ClientRead
			20376	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:58 -05		idle	Client: ClientRead

9. Observaciones y Logs

```
import { writeFile } from 'fs';

const logMessage = (message: string) => {
    const timestamp = new Date().toISOString();
    writeFile('carga_test.log', `${timestamp} - ${message}\n`, { flag: 'a' }, (err) => {
        if (err) console.error('Error al escribir el log:', err);
    });
};

// Ejemplo de uso:
logMessage('Inicio de prueba de carga con 100 usuarios.');
logMessage('Tiempos de respuesta: promedio de 150 ms.');
```