Reporte de Pruebas: Unitarias y de Integración

Proyecto: Sistema de Chat “HablemosYa”

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

1. Introducción

Este documento detalla los resultados obtenidos en las pruebas unitarias y de integración realizadas para el sistema de chat.
Objetivo: Validar que las funciones individuales y las interacciones entre módulos cumplan con los requisitos funcionales y no funcionales.

---

2. Descripción del Sistema

Frontend: Angular
Backend: Node.js
Base de Datos: PostgreSQL
Modelo de Datos:

- `ChatRoomModel`: Representa las salas de chat.
- `UserModel`: Representa los usuarios.
- `ChatMessageModel`: Define los mensajes enviados entre usuarios.

El sistema permite:

1. Crear salas de chat.
2. Enviar y recibir mensajes en tiempo real.

---

3. Resumen de las Pruebas

Tipo de Prueba	Total de Casos	Casos Exitosos	Casos Fallidos	Cobertura (%)
Pruebas Unitarias	25	23	2	92%
Pruebas de Integración	10	9	1	90%
Pruebas de Carga	3	3	0	100%

---

4. Pruebas Unitarias

4.1. Objetivo

Garantizar que las funciones individuales del sistema cumplan con los requisitos y se comporten de manera esperada.

4.2. Herramientas Utilizadas

- **Framework:** Jasmine y Karma para Angular.
- **Cobertura:** Istanbul.

4.3. Resultados por Módulo

4.3.1. Módulo de Servicios: ChatRoomService

Descripción: Implementa la lógica para interactuar con el backend.

ID	Función	Resultado	Observaciones
UT-001	`getChatRooms()`	Aprobado	Recupera correctamente las salas.
UT-002	`getChatMessages(chatRoomId: number)`	Aprobado	Maneja mensajes vacíos sin errores.
UT-003	`createChatRoom(chatRoom: ChatRoomModel)`	Fallido	Error al manejar campos nulos en `chatRoom.name`.

4.3.2. Módulo de Componentes: ChatRoomComponent

Descripción: Renderiza la interfaz de usuario para las salas de chat.

ID	Función	Resultado	Observaciones
UT-004	Renderización inicial de salas	Aprobado	Muestra correctamente las salas.
UT-005	Manejo de errores en carga de mensajes	Fallido	No muestra mensaje de error al fallar.

4.3.3. Módulo de Modelos

Descripción: Verifica la integridad de los datos en las interfaces.

ID	Función	Resultado	Observaciones
UT-006	Validación del modelo `ChatMessageModel`	Aprobado	Maneja campos opcionales sin errores.

---

5. Pruebas de Integración

5.1. Objetivo

Asegurar que las interacciones entre los módulos y servicios funcionan de manera correcta.

5.2. Herramientas Utilizadas

- **Framework:** Postman para pruebas API.
- **Simulación:** MockServiceWorker para simular peticiones.

5.3. Resultados por Caso de Uso

5.3.1. Caso de Uso: Enviar Mensajes

ID	Escenario	Resultado	Observaciones
IN-001	Crear una sala y enviar un mensaje	Aprobado	El mensaje se almacena en PostgreSQL.
IN-002	Enviar múltiples mensajes en paralelo	Aprobado	Tiempos de respuesta < 200 ms.

IN-003	Enviar mensaje con datos inválidos	Aprobado	API responde con código 400 (Bad Request).

5.3.2. Caso de Uso: Carga Masiva de Mensajes

ID	Escenario	Resultado	Observaciones
IN-004	Simular 500 usuarios enviando mensajes	Aprobado	Sin errores, base de datos consistente.

---

6. Pruebas de Carga

6.1. Escenarios Cubiertos

1. **Simulación de carga media:** 500 mensajes/s.
2. **Simulación de carga alta:** 10,000 mensajes/s.

6.2. Herramientas Utilizadas

- **K6:** Simulación de tráfico.
- **Grafana + Prometheus:** Monitoreo de métricas en tiempo real.

6.3. Métricas Obtenidas

Métrica	Escenario 1 (Media)	Escenario 2 (Alta)	Umbral	Estado
Tiempo promedio de respuesta	150 ms	180 ms	≤ 200 ms	Aprobado
Errores HTTP	0%	0.3%	≤ 0.5%	Aprobado
Uso promedio de CPU	65%	78%	≤ 80%	Aprobado
Uso promedio de memoria	70%	85%	≤ 90%	Aprobado

---

7. Análisis de Resultados

7.1. Éxitos

- Las pruebas unitarias demostraron que la mayoría de las funciones trabajan de manera adecuada.

- Las pruebas de carga confirmaron que el sistema puede manejar hasta 10,000 mensajes/s.

7.2. Fallos

1. **Pruebas Unitarias:**
    - El servicio `createChatRoom` no valida correctamente los campos nulos.
    - La interfaz no muestra un mensaje de error al fallar la carga de mensajes.
2. **Pruebas de Integración:**
    - Ningún fallo detectado en las interacciones entre módulos.

---

8. Recomendaciones

1. **Corrección de Errores:**
    - Actualizar la validación en el método `createChatRoom`.
    - Mejorar el manejo de errores en el componente `ChatRoomComponent`.
2. **Optimización:**
    - Revisar la estrategia de indexación en PostgreSQL para mejorar tiempos de consulta bajo carga extrema.
    - Implementar balanceo de carga en el backend para manejar picos mayores a 10,000 mensajes/s.
3. **Pruebas Futuras:**
    - Ampliar las pruebas de carga con más de 50,000 mensajes/s para evaluar la escalabilidad.
    - Realizar pruebas de resistencia durante 24 horas continuas.

---

9. Conclusión

El sistema de chat cumple con la mayoría de los requisitos funcionales y no funcionales en pruebas unitarias, de integración y de carga. Sin embargo, se identificaron áreas de mejora en la validación de datos y en el manejo de errores del frontend.

Este reporte será utilizado como base para priorizar las correcciones y preparar el sistema para un entorno de producción estable.