

# Plan de Pruebas - Sistema de Chat Global Elephantalk

Fecha: 22 de Noviembre de 2025

Proyecto: Elephantalk - Módulo de Comunicación en Tiempo Real

Versión: 1.0

Elaborado por: Oscar Cornejo

## 1. Información General del Plan

### 1.1 Objetivo

Verificar y validar que el Módulo de Chat Global de la red social Elephantalk cumpla con los requisitos funcionales y no funcionales especificados en las historias de usuario (HU-1, HU-2, HU-3). El objetivo principal es garantizar una experiencia de comunicación fluida, segura (autenticada) y persistente antes del despliegue a producción.

### 1.2 Alcance

Este plan abarca las pruebas para la funcionalidad de chat en vivo, incluyendo:

- **Frontend:** Interfaz de usuario, notificaciones visuales, limpieza de campos y renderizado de listas.
- **Backend:** API de WebSockets (Socket.io), validación de tokens JWT, persistencia en base de datos (MongoDB) y lógica de retransmisión.
- **Infraestructura:** Conectividad y latencia básica.

**Fuera de alcance:** Pruebas de carga masiva (>10,000 usuarios simultáneos) para esta fase inicial.

### 1.3 Estrategia de Pruebas

- **Pruebas Funcionales:** Validación contra los Criterios de Aceptación de cada HU.
- **Pruebas de Seguridad:** Verificación estricta de la autenticación del canal (Handshake).
- **Pruebas de Usabilidad:** Facilidad de acceso y feedback visual al usuario.
- **Entorno:** Las pruebas se ejecutarán en un entorno de *Staging* que replica la producción.

## 2. Casos de Prueba por Historia de Usuario

### Historia de Usuario 1: Chat Global para Elephantalk

Rol: Usuario de la red social | Prioridad: Alta

#### CP-001: Visualización del Acceso al Chat

- **Tipo:** Funcional / Usabilidad

- **Precondiciones:** Usuario logueado en la plataforma.
- **Pasos:**
  1. Cargar la página de inicio (Feed).
  2. Inspeccionar la barra de navegación o menú principal.
  3. Buscar el ícono/burbuja indicativo del chat.
- **Resultado Esperado:** El elemento visual del chat es claramente visible y accesible desde el inicio.

## CP-002: Envío de Mensaje de Texto

- **Tipo:** Funcional
- **Precondiciones:** Ventana de chat abierta.
- **Pasos:**
  1. Hacer clic en el campo de entrada de texto.
  2. Escribir "Hola mundo, prueba de envío".
  3. Presionar el botón "Enviar" o la tecla Enter.
- **Resultado Esperado:**
  - El mensaje se muestra inmediatamente en la lista de mensajes propios.
  - El campo de texto se limpia automáticamente tras el envío.

## Historia de Usuario 2: Renderizado de la Interfaz de Chat

Rol: Usuario de la red social | Prioridad: Alta

## CP-003: Carga de Historial al Conectar

- **Tipo:** Funcional
- **Precondiciones:** Existen mensajes previos en la base de datos.
- **Pasos:**
  1. Refrescar la página o navegar hacia la sección de chat.
  2. Esperar la conexión del socket.
  3. Verificar el área de contenido.
- **Resultado Esperado:** Se renderizan los últimos mensajes (ej. 50) en orden cronológico ascendente. No se muestran mensajes vacíos o corruptos.

## CP-004: Recepción en Tiempo Real (Sincronización)

- **Tipo:** Funcional / Tiempo Real
- **Precondiciones:** Dos navegadores abiertos (Usuario A y Usuario B) en el chat.
- **Pasos:**
  1. Usuario A envía: "Mensaje de prueba sincrónica".
  2. Usuario B observa su pantalla sin recargar.
- **Resultado Esperado:** El mensaje aparece en la pantalla del Usuario B casi instantáneamente (< 500ms), mostrando correctamente el nombre de usuario del emisor (Usuario A) y el contenido.

## Historia de Usuario 3: Autenticación y Backend

Rol: Desarrollador (Backend) | Prioridad: Crítica

### CP-005: Seguridad - Conexión sin Token

- **Tipo:** Seguridad (Negativa)
- **Precondiciones:** Cliente de pruebas (ej. Postman/Socket.io Client) sin token.
- **Pasos:**
  1. Intentar establecer conexión WebSocket (ws://...) sin enviar el header de autorización o handshake auth.
- **Resultado Esperado:** El servidor rechaza la conexión inmediatamente (Evento disconnect o error 401/403). No se permite unir al canal global.

### CP-006: Seguridad - Conexión con Token Válido

- **Tipo:** Seguridad (Positiva)
- **Precondiciones:** Token JWT válido generado por el endpoint de login.
- **Pasos:**
  1. Establecer conexión WebSocket enviando el token en el handshake.
- **Resultado Esperado:** Conexión exitosa (Evento connect). El servidor asocia el Socket ID con el User ID en memoria.

### CP-007: Persistencia y Retransmisión (Broadcast)

- **Tipo:** Backend / Integración
- **Pasos:**
  1. Enviar evento send\_message con payload válido.
  2. Verificar inmediatamente la colección de Messages en la base de datos (MongoDB).
  3. Verificar logs del servidor para confirmación de retransmisión.
- **Resultado Esperado:**
  - El mensaje existe en la BD con los campos correctos (content, senderId, timestamp).
  - El servidor emite el evento new\_message a todos los clientes conectados (Broadcast).

## 3. Matriz de Trazabilidad

ID Caso	Historia de Usuario Asociada	Tipo	Prioridad	Estado
CP-001	HU-1: Chat Global	Usabilidad	Alta	Pendiente
CP-002	HU-1: Chat	Funcional	Alta	Pendiente

	Global			
CP-003	HU-2: Renderizado	Funcional	Media	Pendiente
CP-004	HU-2: Renderizado	Funcional	Crítica	Pendiente
CP-005	HU-3: Autenticación	Seguridad	Crítica	Pendiente
CP-006	HU-3: Autenticación	Seguridad	Alta	Pendiente
CP-007	HU-3: Backend	Backend	Alta	Pendiente

## 4. Criterios de Aceptación del Plan

### Criterios de Entrada

- Código del módulo de chat desplegado en ambiente de pruebas (Staging).
- Base de datos de prueba operativa y accesible.
- Usuarios de prueba creados (mínimo 2 usuarios activos).

### Criterios de Salida (Definition of Done)

- 100% de los casos de prueba críticos (CP-004, CP-005, CP-007) aprobados.
- Tasa de éxito global > 90%.
- Ningún bug de severidad "Crítica" o "Alta" abierto.

## 5. Recursos y Herramientas

### Herramientas de Prueba

- **Interfaz:** Google Chrome, Mozilla Firefox (últimas versiones).
- **WebSockets:** Postman (soporte Socket.io) o Firecamp.
- **Base de Datos:** MongoDB Compass (para verificar persistencia).

### Ambiente de Pruebas

- **Frontend:** React + Vite (localhost / Vercel Staging).
- **Backend:** NestJS + Socket.io Gateway.
- **Red:** Conexión estable > 5 Mbps.

## 6. Gestión de Defectos

## Clasificación de Severidad

- Crítico:** El chat no conecta o los mensajes no se entregan (Bloqueante).
- Alto:** La persistencia falla o la seguridad es vulnerable (ej. conectar sin token).
- Medio:** Problemas visuales menores (ej. el scroll no baja automáticamente).
- Bajo:** Errores estéticos (colores, espaciado).

## 7. Riesgos y Mitigación

Riesgo	Impacto	Mitigación
<b>Desconexiones frecuentes</b>	Alto	Implementar lógica de reconexión automática en el cliente (Frontend).
<b>Latencia alta</b>	Medio	Optimizar payload de mensajes (JSON ligero) y usar índices en MongoDB.
<b>Fallo en Base de Datos</b>	Crítico	Implementar manejo de errores try-catch en el Gateway para no tumbar el servidor.

## 8. Conclusiones

Este plan garantiza que el módulo de chat de Elephantalk no solo funcione visualmente, sino que cumpla con los estándares de seguridad y persistencia requeridos por la arquitectura del sistema. La ejecución exitosa de este plan es mandatoria para el paso a producción.