

Especificación de  
Requisitos Software

## CASOS DE USO

### Biblioteca Autónoma Inteligente

Yandris Miguel Rivera Torres

Alisson Yamel Reyes Ricardo

Gino Maximiliano Bermúdez Santos

Andy Bryan Alejandro Vera

Fecha: 22 de octubre de 2025

## HISTÓRICO

<i>NOMBRE</i>	<i>FECHA</i>	<i>CAMBIO</i>	<i>PAGINA</i>

## TABLA DE CONTENIDOS

### 1 Contenido

TABLA DE CONTENIDOS .....	3
1. INTRODUCCIÓN .....	4
1.1 Objetivos .....	4
1.2 Ámbito.....	4
2. Especificación de Casos de Uso .....	5
2.1. Característica 1 – Gestión de usuarios .....	5
2.1.1. Caso de uso 01 – Registrar de Cuenta.....	6
2.1.2. Caso de uso 02 – Iniciar Sesión .....	6
2.1.3. Caso de uso 03 – Recuperar Contraseña.....	7
2.1.4. Caso de uso 04 – Cerrar Sesión.....	7
1.3 <a href="#">Característica 2 – Gestión de contactos</a> .....	8
2.1.5. Caso de uso 05 – Buscar Contacto .....	8
2.1.6. Caso de uso 06 – Enviar solicitud de contacto .....	8
2.1.7. Caso de uso 07 – Gestionar Solicitud Entrantes.....	9
2.1.8. Caso de uso 08 – Eliminar Contacto.....	9
2.2. Característica 2 – Mensajería Privada (chat 1 a 1) .....	10
2.2.1. Caso de uso 09 – Iniciar Chat 1 a 1.....	10
2.2.2. Caso de uso 10 – Enviar mensaje.....	11
2.2.3. Caso de uso 11 – Recibir mensaje.....	11
2.2.4. Caso de uso 12 – Actualizar Historial .....	11
2.2.5. Caso de uso 13 – Recibir notificación.....	12
2.3. Característica 3 – Gestión del historial del chat .....	12
2.3.1. Caso de uso 14 – Visualizar historial .....	13
2.3.2. Caso de uso 15 – Actualizar historial .....	13
2.3.3. Caso de uso 16 – Gestionar indicadores de presencia .....	13
1.3.1 Caso de uso 17 – Visualización del estado de presencia .....	14
2.4. Característica 4 – Gestión y uso de Canales .....	15
1.3.2 Caso de uso 18 – Crear nuevo canal .....	15
1.3.3 Caso de uso 19– Unirse a canal existente .....	15
1.3.4 Caso de uso 20 – Enviar Mensaje a Canal .....	16
3. REQUISITOS NO FUNCIONALES .....	16

## 1. INTRODUCCIÓN

### 1.1 Objetivos

- Desarrollar una aplicación de chat en tiempo real aplicando principios de programación funcional para el procesamiento de mensajes y gestión del historial con estructuras persistentes e inmutables tanto en back-end como en el front-end de la web, combinada con programación imperativa u orientada a objetos para la interfaz y comunicación con el servidor.

### 1.2 Ámbito

- El desarrollo se centra en una aplicación web de mensajería en tiempo real que incluye
  - Autenticación y perfil básico: Registro, inicio / cierre de sesión de usuario. Esto es necesario para identificar al usuario que interactúa con la lógica de chat.
  - Gestión de contactos: capacidad de agregar y visualizar usuarios en una lista de contactos. Muestra la gestión de lista de datos inmutables por usuario.
  - Canales de Chat: Permite a los usuarios crear, unirse y salir de salas de chat grupales persistentes.
  - Chat 1 a 1: Comunicación bidireccional en tiempo real entre dos usuarios. Foco de la lógica pura de transformación de mensajes y estado de conversación.
  - Historial y persistencia: Almacenamiento y recuperación de la secuencia de mensajes de una conversación. El historial se trata como una estructura de datos persistentes o inmutables.
  - Indicadores de persistencia: Mostrar si un usuario está conectado o desconectado. Demuestra la capacidad del sistema a reaccionar a eventos de red. Actualizando un estado inmutable de presencia.

#### Delimitaciones:

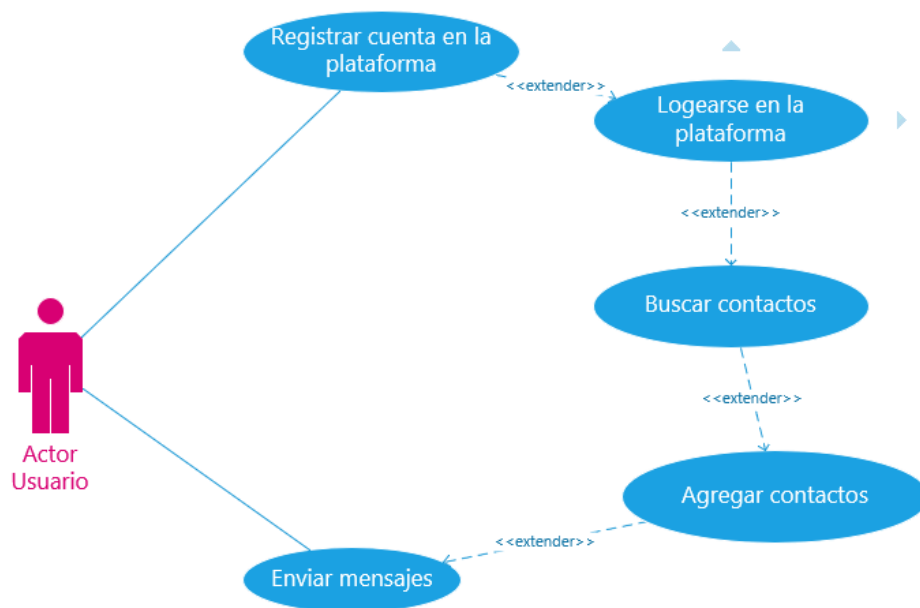
- Lógica central funcional: la lógica de negocio fundamental (procesamiento de mensajes, timestamps, actualización del historial) será implementada exclusivamente con funciones puras. Esto es el objetivo central del proyecto. Se garantiza que el core es predecible y testeable.
- Capa imperativa: la interfaz de usuario (Front-end) y los módulos interacción con el exterior (I/O, WebSockets, base de datos) se implementan bajo paradigmas imperativos u orientados a objetos. Reconoce la necesidad de usar OO/imperativo para manejar los “efectos” del mundo real.

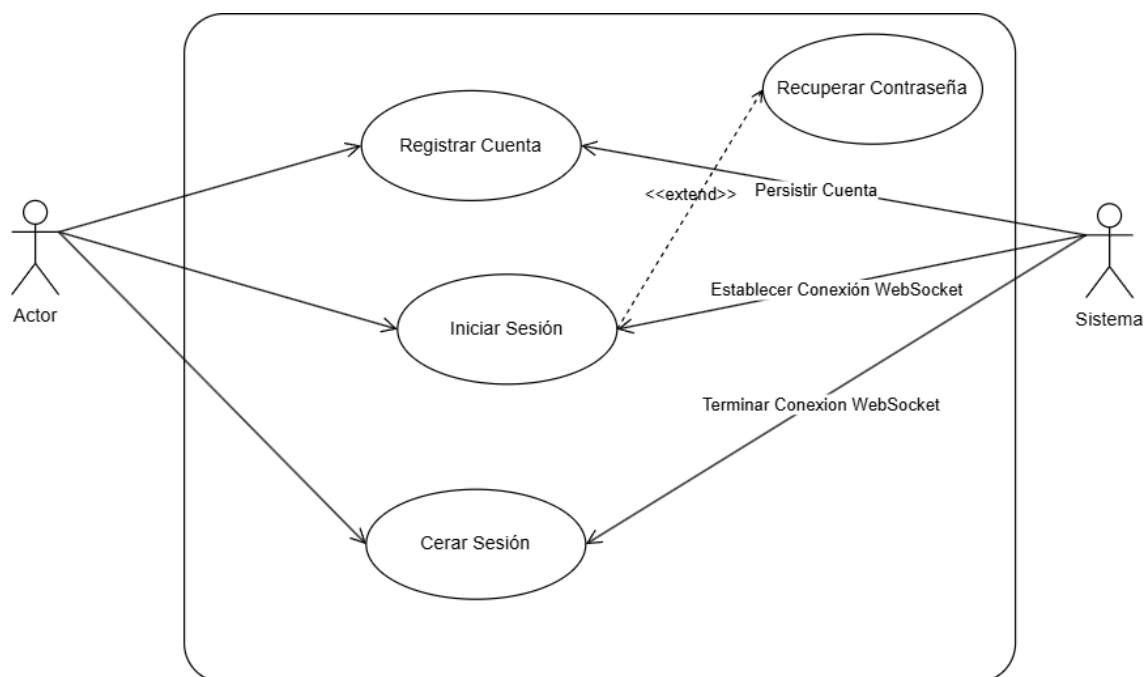
Adicionales:

- Multimedia (Archivos, imágenes voz).
- Cifrado de mensaje End-to-End.
- Notificación push nativas.

## 2. Especificación de Casos de Uso

### 2.1. Característica 1 – Gestión de usuarios





### 2.1.1. Caso de uso 01 – Registrar de Cuenta

Referencia:	CU01
Nombre:	Registrar cuenta
Descripción:	Permite al usuario registrarse para hacer uso de nuestro sistema de mensajería instantánea
Actor:	Usuario
Relaciones:	Ninguna
Precondición	El usuario no debe tener una cuenta registrada previamente.
Flujo básico:	<ol style="list-style-type: none"> <li>1. El usuario entra a la página principal</li> <li>2. El usuario entra a la sección de registro</li> <li>3. El usuario ingresa sus datos</li> <li>4. El usuario crea su cuenta</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>1. Si el usuario al registrarse ya tiene una cuenta existente, el sistema muestra error y solicita corrección.</li> </ol>
Postcondición:	Una vez creada su cuenta el usuario puede usar la plataforma.

### 2.1.2. Caso de uso 02 – Iniciar Sesión

Referencia:	CU02
Nombre:	Iniciar Sesión
Descripción	Permite al usuario iniciar sesión en la plataforma de chat utilizando sus credenciales registradas.
Actor	Usuario

Relaciones	Ninguna
Precondición	El usuario debe tener una cuenta registrada previamente (CU01)
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla de inicio de sesión.</li> <li>2. El usuario ingresa su nombre de usuario y contraseña.</li> <li>3. El sistema valida las credenciales.</li> <li>4. Si la validación es exitosa, el sistema redirige al usuario a la pantalla principal de chat.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>A. Si las credenciales son incorrectas, el sistema muestra un mensaje de error y permite al usuario reintentar el inicio de sesión.</li> </ol>
Postcondición	El usuario está autenticado y tiene acceso a las funcionalidades de chat, como buscar contactos y enviar mensajes.

### 2.1.3. Caso de uso 03 – Recuperar Contraseña

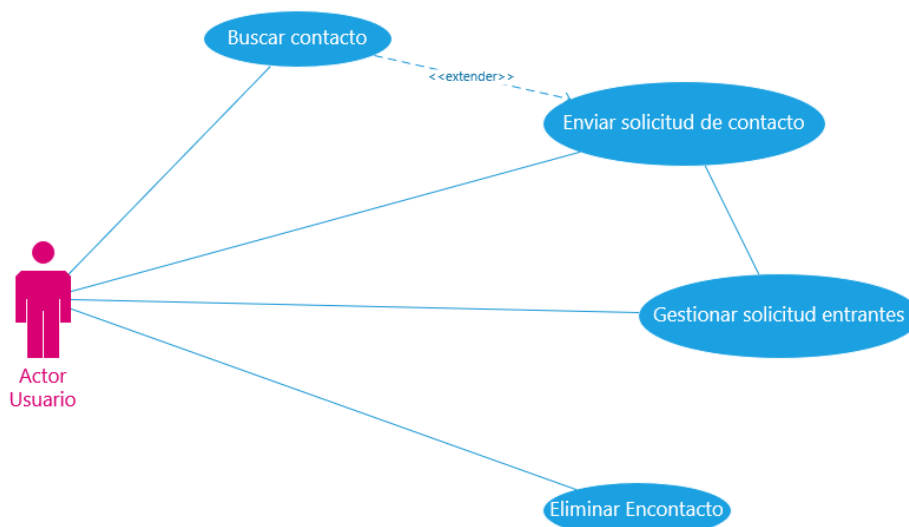
Referencia:	CU03
Nombre:	Recuperar Contraseña
Descripción	Permite al usuario recuperar su contraseña en el caso de que la haya olvidado.
Actor	Usuario
Relaciones	<<Extend>> de Inicio de sesión (CU02).
Precondición	Fallo de autenticación.
Flujo básico	<ol style="list-style-type: none"> <li>1. Usuario solicita restablecimiento.</li> <li>2. El sistema envía un enlace de un solo uso al correo.</li> <li>3. La función cambio de contraseña genera un nuevo hash inmutable.</li> </ol>
Flujo alternativo	A.
Postcondición	La contraseña es restablecida y el usuario puede iniciar sesión.

### 2.1.4. Caso de uso 04 – Cerrar Sesión

Referencia:	CU04
Nombre:	Cerrar Sesión
Descripción	Permite al usuario cerrar su sesión de manera manual, o si el usuario supera el tiempo de inactividad se cerrará la sesión automáticamente.
Actor	Sistema
Relaciones	
Precondición	Tener una sesión activa (CU02)
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario cierra sesión.</li> <li>2. La conexión WebSocket se termina.</li> <li>3. Actualiza la presencia o estado.</li> </ol>
Flujo alternativo	A.
Postcondición	Usuario desconectado estado de presencia "offline".

### 1.3 Característica 2 – Gestión de contactos

#### Característica 2 – Gestión de Contactos



#### 2.1.5. Caso de uso 05 – Buscar Contacto

Referencia:	CU05
Nombre:	Buscar contactos
Descripción	Permite a un buscar a otros usuarios por su nombre de usuario
Actor	Usuario
Relaciones	
Precondición	Usuario autenticado (CU02)
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario ingresa criterio de búsqueda.</li> <li>2. El sistema muestra los resultados.</li> <li>3. El estado inmutable de la búsqueda se carga en el frontend.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>A. Si no se encuentra nada se muestra mensaje de “contacto no encontrado”</li> </ol>
Postcondición	Permite al usuario encontrar otro usuario dentro de la aplicación.

#### 2.1.6. Caso de uso 06 – Enviar solicitud de contacto

Referencia:	CU06
Nombre:	Enviar solicitud de contacto
Descripción	El usuario mandar solicitud al usuario destinatario.
Actor	Usuario
Relaciones	
Precondición	Un usuario destinatario debe existir.
Flujo básico	<ol style="list-style-type: none"> <li>1. Usuario selecciona destinatario y envía solicitud.</li> <li>2. Genera un nuevo estado inmutable de “solicitudes pendientes” para ambos usuarios.</li> <li>3. El sistema persiste el estado.</li> </ol>
Flujo alternativo	

Postcondición	La solicitud queda registrada en el sistema.
---------------	--

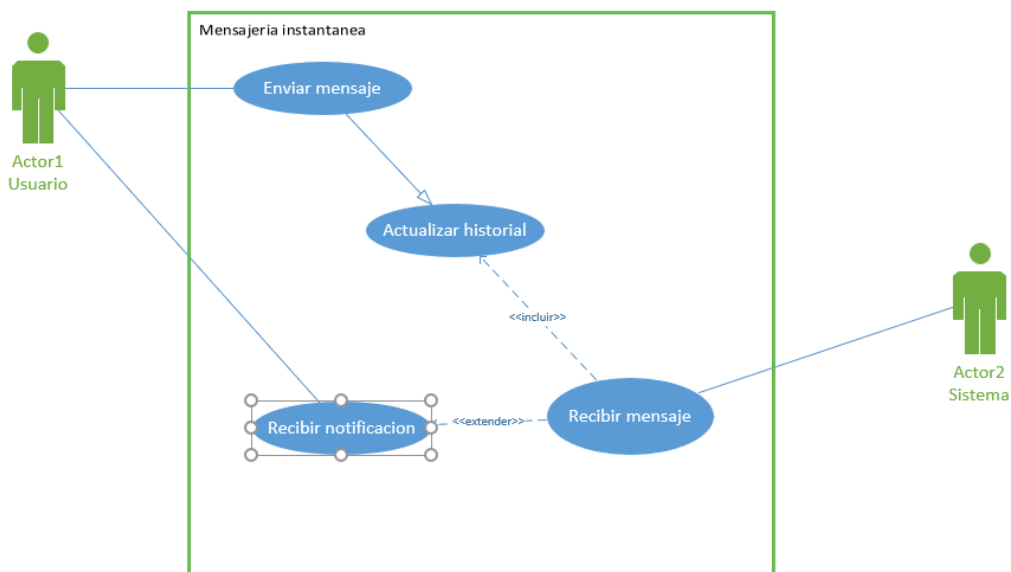
### 2.1.7. Caso de uso 07 – Gestionar Solicitud Entrantes

Referencia:	CU07
Nombre:	Gestionar solicitudes entrantes
Descripción	Gestiona la solicitud pendiente entre usuarios.
Actor	Usuario
Relaciones	Tener una solicitud activa (CU06)
Precondición	Un usuario destinatario debe existir.
Flujo básico	<ol style="list-style-type: none"> <li>1. Usuario consulta solicitudes.</li> <li>2. Usuario selecciona “Aceptar” o “Rechazar”.</li> <li>3. Función gestionarSolicitud() genera un nuevo estado inmutable, eliminando la solicitud y, si es aceptada creando una conexión mutua.</li> </ol>
Flujo alternativo	
Postcondición	Si es aceptada, los usuarios son contactos mutuos; si es rechazada, la solicitud se elimina.

### 2.1.8. Caso de uso 08 – Eliminar Contacto

Referencia:	CU08
Nombre:	Eliminar contacto
Descripción	El usuario puede eliminar uno de sus contactos.
Actor	Usuario
Relaciones	<<Include>> Buscar contactos existentes (CU03)
Precondición	El usuario a eliminar debe existir.
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario selecciona contacto y “eliminar”.</li> <li>2. Función eliminarContacto() genera un nuevo estado inmutable de la lista de contactos.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>A. Si no existe el usuario a eliminar, mostrar mensaje de “usuario no encontrado”.</li> </ol>
Postcondición	El usuario elimina un contacto de su lista.

## 2.2. Característica 3 – Mensajería Privada (chat 1 a 1)



### 2.2.1. Caso de uso 09 – Iniciar Chat 1 a 1

Referencia:	CU09
Nombre:	Iniciar chat
Descripción	El usuario selecciona un contacto para abrir o continuar una conversación privada.
Actor	Usuario
Relaciones	Ninguna
Precondición	Tener una sesión abierta o estar conectado (CU02) y la conversación con el contacto debe estar abierta.
Flujo básico	<ol style="list-style-type: none"> <li>1. Se escoge el usuario.</li> <li>2. El usuario escribe el mensaje y presiona enviar.</li> <li>3. Se procesa el mensaje, se valida y genera el estado inmutable.</li> <li>4. Persiste el nuevo estado inmutable y envía el mensaje a través del WebSocket al destinatario.</li> <li>5. La interfaz de usuario se actualiza para mostrar el nuevo mensaje.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>A. La función debe validar, detecta el mensaje esta vacío o contiene contenido no permitido, el flujo se termina y se notifica al usuario.</li> </ol>
Postcondición	El mensaje se encuentra persistido en el historial de la conversación.

**2.2.2. Caso de uso 10 - Enviar mensaje**

Referencia:	CU10
Nombre:	Enviar un mensaje
Descripción	Permite al usuario enviar mensajes instantáneos a otros usuarios conectados en la aplicación.
Actor	Usuario
Relaciones	Ninguna
Precondición	Tener una sesión abierta o estar conectado (CU02).
Flujo básico	<ol style="list-style-type: none"> <li>6. El usuario escribe un mensaje en la caja de texto.</li> <li>7. El usuario presiona el botón enviar</li> <li>8. El sistema procesa el mensaje (limpia el texto, normaliza y agrega timestamp)</li> <li>9. El mensaje se envía al servidor y se distribuye a los demás usuarios conectados.</li> </ol>
Flujo alternativo	A. Si no hay conexión con el servidor, el sistema muestra un mensaje de error.
Postcondición	El mensaje se muestra en la ventana del chat y se actualiza el historial.

**2.2.3. Caso de uso 11 - Recibir mensaje**

Referencia:	CU11
Nombre:	Recibir mensaje
Descripción	Permite al usuario visualizar mensajes enviados por otros usuarios en tiempo real
Actor	Sistema
Relaciones	<<Include>> Actualizar Historial (CU07), <<Extend>> Recibir Mensaje (CU08)
Precondición	El usuario debe estar conectado al servidor (CU02) y ser miembro de la sala o chat.
Flujo básico	<ol style="list-style-type: none"> <li>1. El sistema detecta un nuevo mensaje recibido.</li> <li>2. El sistema muestra el mensaje en la interfaz del usuario.</li> </ol>
Flujo alternativo	A. Si se pierde la conexión, el sistema intentar reconectar automáticamente.
Postcondición	El usuario visualiza los mensajes nuevos sin necesidad de recargar la página.

**2.2.4. Caso de uso 12 - Actualizar Historial**

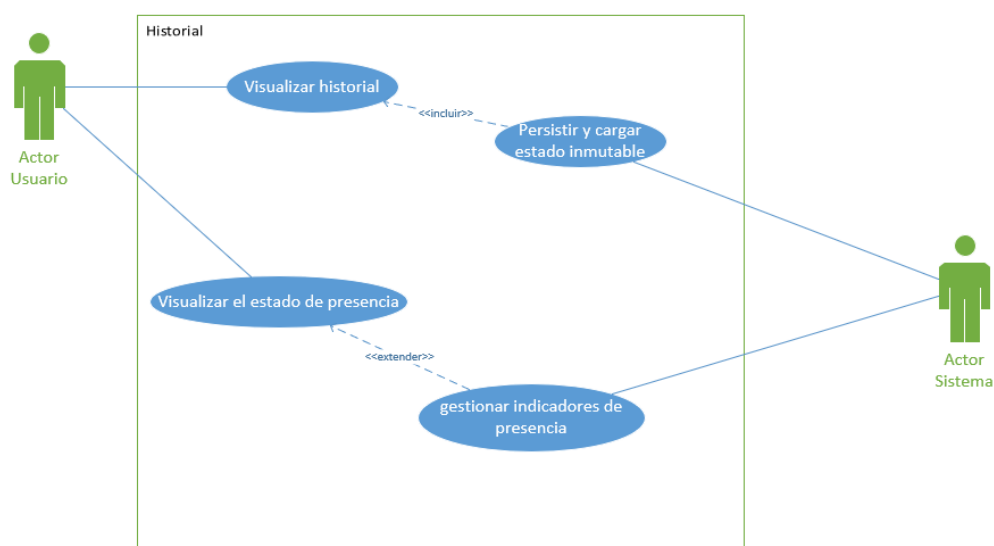
Referencia:	CU12
Nombre:	Actualizar Historial (Función Pura)
Descripción	Función responsable de generar un nuevo historial a partir de un estado anterior y nuevo evento, sin mutaciones.
Actor	Sistema
Relaciones	<<Include>> Recibir Mensaje (CU05) y enviar Mensaje (CU04)
Precondición	Se ha generado un nuevo evento de chat.
Flujo básico	<ol style="list-style-type: none"> <li>1. Se recibe el estado del historial anterior</li> <li>2. Se aplica la lógica de anexar el nuevo evento.</li> <li>3. Se retorna una copia totalmente nueva de la estructura.</li> </ol>
Flujo alternativo	

Postcondición	Se genera una nueva versión del historial, lista para ser mostrada.
---------------	---

### 2.2.5. Caso de uso 13 – Recibir notificación

Referencia:	CU13
Nombre:	Recibir notificación
Descripción	Alertar al usuario sobre nuevos mensajes cuando la ventana no está enfocada.
Actor	Sistema (Extensión del CU05)
Relaciones	
Precondición	Se recibe un mensaje (CU05) y se detecta que no esté enfocada la ventana de chat.
Flujo básico	<ol style="list-style-type: none"> <li>1. Verifica el foco en la ventana.</li> <li>2. Si no hay foco, se muestra una notificación.</li> <li>3. El usuario puede hacer clic en la notificación para enfocar en la ventana.</li> </ol>
Flujo alternativo	
Postcondición	El usuario ha sido alertado del nuevo mensaje fuera de la interfaz principal.

## 2.3. Característica 4 – Gestión del historial del chat



**2.3.1. Caso de uso 14 – Visualizar historial**

Referencia:	CU14
Nombre:	Visualizar historial
Descripción	Permite al usuario ver la secuencia completa de mensajes de cualquier conversación.
Actor	Usuario.
Relaciones	<<Include>> Cargar estado inmutable (CU07)
Precondición	El usuario debe haber iniciado sesión (CU02) y seleccionado una conversación.
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario accede a la interfaz principal del chat.</li> <li>2. El sistema carga el historial inmutable de mensajes. (CU07)</li> <li>3. El sistema muestra los mensajes anteriores en la pantalla.</li> </ol>
Flujo alternativo	A. Historial vacío: si CU09 devuelve un estado inmutable vacío, la interfaz muestra un mensaje de “inicia la conversación”.
Postcondición	El usuario puede leer los mensajes enviados y recibidos anteriormente.

**2.3.2. Caso de uso 15 – Actualizar historial**

Referencia:	CU15
Nombre:	Persistir y Cargar Estado inmutable
Descripción	Gestiona la lectura y escritura del estado inmutable del historial en la base de datos.
Actor	Sistema
Relaciones	<<Include>> en Visualizar Historial (CU08)
Precondición	<ol style="list-style-type: none"> <li>1. Para persistir: Existe un nuevo estado inmutable.</li> <li>2. Para cargar: se solicita el inicio de una conversación.</li> </ol>
Flujo básico	<ol style="list-style-type: none"> <li>1. El sistema recibe un nuevo estado, proporcionado por enviar mensaje (CU04).</li> <li>2. El sistema tiene registrado la estructura de datos inmutables.</li> <li>3. El sistema retorna esa estructura para mostrarla.</li> <li>4. El sistema muestra los mensajes anteriores en la pantalla.</li> </ol>
Flujo alternativo	
Postcondición	La base de datos refleja el estado actual del historial, se muestra en la estructura inmutable.

**2.3.3. Caso de uso 16 – Gestionar indicadores de presencia**

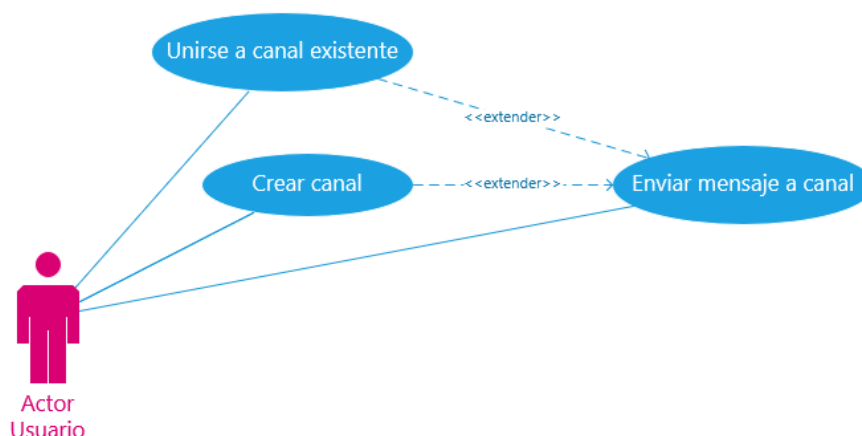
Referencia:	CU16
Nombre:	Gestionar indicadores de presencia
Descripción	Actualiza y distribuye el estado de conexión (online / offline) de los usuarios.
Actor	Sistema
Relaciones	<<Extends>> iniciar sesión (CU02)
Precondición	1. Ocurre un evento de conexión o desconexión del usuario.
Flujo básico	<ol style="list-style-type: none"> <li>1. La capa de WebSocket detecta un evento de red.</li> <li>2. El sistema ejecuta una función para actualizar el estado de presencia.</li> <li>3. El servidor distribuye estos estados a los usuarios.</li> <li>4. El frontend actualiza la lista de la presencia de forma inmutable.</li> </ol>

Flujo alternativo	
Postcondición	El usuario puede leer los mensajes enviados y recibidos anteriormente.

### 1.3.1 Caso de uso 17 – Visualización del estado de presencia

Referencia:	CU17
Nombre:	Visualización del estado de presencia
Descripción	Se muestra el estado de presencia de un usuario (conectado/ausente) a sus contactos
Actor	Sistema
Relaciones	<<Extends>> iniciar sesión (CU02)
Precondición	El sistema debe haber detectado que un usuario (o varios) han cambiado su estado de conexión (conectado, desconectado, ausente).
Flujo básico	<ol style="list-style-type: none"> <li>1. El sistema (servidor WebSocket) detecta un evento de red relevante para la presencia</li> <li>2. El sistema actualiza el estado de presencia en su base de datos en tiempo real o estructura en memoria.</li> <li>3. El sistema distribuye los cambios de estado de presencia a los clientes conectados, especialmente a los contactos del usuario que cambió de estado.</li> <li>4. El frontend de cada usuario actualiza la lista de contactos/usuarios en tiempo real, mostrando el estado de presencia</li> <li>5. El usuario visualiza el estado de presencia de las personas con las que mantiene una conversación.</li> </ol>
Flujo alternativo	
Postcondición	El usuario puede ver el estado de la persona con la que mantiene una conversacion

## 2.4. Característica 5 – Gestión y uso de Canales



### 1.3.2 Caso de uso 18 – Crear nuevo canal

Referencia:	CU18
Nombre:	Crear nuevo canal
Descripción	Permite crear una nueva sala de chat grupal persistente.
Actor	Usuario.
Relaciones	Ninguna
Precondición	Inicio de sesión de usuario (CU02)
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario ingresa el nombre del canal.</li> <li>2. Se genera un estado global inmutable con la nueva entidad crearCanal().</li> <li>3. El sistema persiste este estado.</li> </ol>
Flujo alternativo	
Postcondición	El canal existe y el usuario es su primer miembro.

### 1.3.3 Caso de uso 19– Unirse a canal existente

Referencia:	CU19
Nombre:	Unirse a canal existente
Descripción	Permite a un usuario convertirse en miembro de un canal
Actor	Usuario
Relaciones	
Precondición	El canal debe existir (CU13) y el usuario estar autenticado (CU02)
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario selecciona “unirse”.</li> <li>2. Se ejecuta la función uniserAcanal() devuelve el nuevo estado inmutable del canal con la nueva suscripción.</li> <li>3. El sistema persiste el nuevo estado.</li> </ol>
Flujo alternativo	
Postcondición	El usuario es miembro activo del canal.

### 1.3.4 Caso de uso 20 – Enviar Mensaje a Canal

Referencia:	CU20
Nombre:	Enviar mensaje a canal.
Descripción	Se envía mensaje a todos los miembros activos del canal.
Actor	Usuario
Relaciones	
Precondición	El usuario debe ser miembro del canal.
Flujo básico	<ol style="list-style-type: none"> <li>1. El usuario envía el mensaje.</li> <li>2. Se ejecuta la función procesarMensajeCanal() para generar el nuevo estado inmutable del historial del canal.</li> <li>3. El sistema persiste en su estado y distribuirlo.</li> </ol>
Flujo alternativo	
Postcondición	El historial inmutable del canal se ha actualizado y distribuido.

## 3. REQUISITOS NO FUNCIONALES

RNF01. Usabilidad: La interfaz debe ser simple, intuitiva y accesible desde navegadores modernos.

RNF02. Rendimiento: Las operaciones de envío y recepción deben ejecutarse con baja

RNF03. Mantenibilidad: El código debe estar dividido en módulos funcionales y de presentación, facilitando su lectura y modificación.

RNF04. Escalabilidad: La arquitectura debe permitir agregar más usuarios o salas sin modificar la lógica base del chat.

RNF05. Compatibilidad: Debe funcionar en los principales navegadores.

RNF06. Confiabilidad: El sistema debe mantener la consistencia de los mensajes y el historial aunque se produzcan desconexiones temporales.