

**UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**DEPARTAMENTO DE LAS CIENCIAS DE LA COMPUTACIÓN**

**APLICACIONES DISTRIBUIDAS**

**INTEGRANTES:**

DE LA CADENA MONCAYO LEONARDO JAVIER

QUISHPE MAIGUA SANTIAGO DAVID

IPIALES ULCUANGO CARLOS JOAO

**NRC:**

15318

### 

[**Introducción 3**](#_52tq7pffhcps)

[**Descripción del problema 3**](#_j06uad9j55gw)

[**Objetivos del proyecto 3**](#_t1p8xernlp7q)

[**Alcance del Proyecto 3**](#_1bgfnjfx7q9r)

[**Requisitos del Proyecto 3**](#_vqs0i9vqmq5f)

[**Características del Sistema 4**](#_r609ffqxlzi)

[**Requerimientos Técnicos 4**](#_e4vrj4evtuol)

[**Metodología Utilizada para el Desarrollo 4**](#_201f6arhexuz)

[**Descripción del Diseño y Arquitectura del Sistema 5**](#_i11nwdw6izse)

[**Requisitos Funcionales 6**](#_ov1ge0640htx)

[**OBJ-01 6**](#_wa2iuca6b5a7)

[**REQ002 7**](#_xqtjt13byvwi)

[**REQ003 8**](#_on6tu05qkoi5)

[**REQ004 9**](#_albzhtcxab0m)

[**OBJ-02 11**](#_rr5mllt05bxs)

[**REQ006 12**](#_5c5z12spsgll)

[**REQ007 13**](#_5jjsqvhc2rsh)

[**Requisitos No Funcionales 15**](#_ndmh82ougrru)

[**Casos de Uso 16**](#_fonmcv37jbsp)

[**Herramientas de Desarrollo 18**](#_xr7towvcvhke)

[**Diseño y arquitectura del Sistema 19**](#_t5t0liek7xma)

[**Diagrama de clases 19**](#_gvsmyewxw0yg)

[**Diagrama de Secuencia 20**](#_jlj7s6nfnabz)

[**Diagrama de Componentes 20**](#_jinjb1pxlac1)

[**Resultados Obtenidos 21**](#_tur6s4kyzj)

[**Interfaces del Aplicativo 21**](#_7sxl1plr5azi)

[**Conclusiones 21**](#_nsths9mrvnp8)

[**Recomendaciones(mejoras futuras) 21**](#_argh325gj2s)

### 

### 

# **Introducción**

El presente informe detalla el desarrollo de un servicio de WebSockets para un chat en vivo, utilizando Spring Boot en el backend y React en el frontend. Este sistema permite que varios usuarios se conecten simultáneamente, se registren y participen en un chat en tiempo real, enviando mensajes globales visibles para todos los usuarios conectados. El proyecto busca mejorar la interacción y comunicación en tiempo real entre usuarios.

# **Descripción del problema**

Actualmente, muchas aplicaciones de chat en vivo enfrentan problemas de latencia, escalabilidad y complejidad en la integración de diversas tecnologías. Este proyecto pretende resolver estos problemas desarrollando un sistema de chat en tiempo real utilizando WebSockets, que ofrece comunicación bidireccional eficiente, y tecnologías modernas como Spring Boot y React para una mejor experiencia de usuario.

# **Objetivos del proyecto**

* Desarrollar un servicio de WebSockets utilizando Spring Boot para permitir la comunicación en tiempo real.
* Crear una interfaz de usuario intuitiva y receptiva utilizando React.
* Permitir el registro de usuarios y la autenticación segura.
* Facilitar la interacción entre usuarios conectados y el envío de mensajes globales en tiempo real.

# **Alcance del Proyecto**

El proyecto abarca el desarrollo de un sistema de chat en tiempo real que incluye:

* Backend con Spring Boot para manejar WebSockets y gestionar la lógica de negocio.
* Frontend con React para permitir el registro de usuarios, autenticación y participación en el chat.
* Integración de una base de datos para almacenar información de usuarios y mensajes.

No se incluye la implementación de características avanzadas como mensajes privados, archivos adjuntos, o cifrado de extremo a extremo en esta fase inicial.

# **Requisitos del Proyecto**

1. Back-End:

* Implementar un servidor utilizando Spring Boot.
* Utilizar WebSockets para la comunicación en tiempo real.
* Manejar las conexiones concurrentes de múltiples usuarios.

1. Front-End:

* Desarrollar una interfaz de usuario utilizando un framework de front-end.
* Integrar la funcionalidad de WebSocket para la comunicación en tiempo real.
* Crear vistas para el registro y la sala de chat.
* Mostrar la lista de usuarios conectados en tiempo real.

# **Características del Sistema**

* Registro de usuarios.
* Enviar y recibir mensajes instantáneamente entre usuarios.
* Notificaciones en tiempo real de nuevos mensajes.
* Visualización de la lista de usuarios conectados en tiempo real.
* Manejo de desconexiones y reconexiones automáticas.

# **Requerimientos Técnicos**

* Backend: Spring Boot, WebSockets, Java 11+, Maven
* Frontend: React, JavaScript (ES6+), HTML5, CSS3
* Base de Datos: MongoDB
* Servidor: localhost
* Control de Versiones: Git, GitHub
* Herramientas de Desarrollo: IntelliJ IDEA, Visual Studio Code

# **Metodología Utilizada para el Desarrollo**

Para el desarrollo de este proyecto, se utilizó la metodología ágil SCRUM. Esta metodología permite una gestión flexible y eficiente del proyecto, asegurando que se cumplan los plazos y se adapten los requerimientos según las necesidades del cliente y del proyecto. A continuación se detallan los apartados específicos de la metodología utilizada:

1. Sprints:

* Descripción: El proyecto se dividió en varios sprints, cada uno de dos semanas de duración. Cada sprint tiene como objetivo entregar un incremento del producto con funcionalidad operativa.
* Planificación: Antes de cada sprint, se realizó una reunión de planificación (Sprint Planning) para definir los objetivos y tareas del sprint.

1. Daily Stand-up:

* Descripción: Se llevaron a cabo reuniones diarias de pie (Daily Stand-up) para revisar el progreso, identificar obstáculos y planificar el trabajo del día.
* Duración: 15 minutos.
* Participantes: Todo el equipo de desarrollo.

1. Sprint Review:

* Descripción: Al final de cada sprint, se realizó una revisión (Sprint Review) para demostrar el trabajo completado y recibir retroalimentación de los interesados.
* Objetivo: Asegurar que el producto cumple con los requisitos y expectativas del cliente.

1. Sprint Retrospective:

* Descripción: Después de cada sprint, se llevó a cabo una retrospectiva (Sprint Retrospective) para discutir qué funcionó bien, qué no, y cómo mejorar en el próximo sprint.
* Objetivo: Mejorar continuamente el proceso de desarrollo.

1. Product Backlog:

* Descripción: Se mantuvo un Product Backlog, que es una lista priorizada de todas las funcionalidades, mejoras y correcciones necesarias para el producto.
* Gestión: El Product Owner fue responsable de gestionar el backlog y priorizar los elementos según el valor para el cliente.

1. Herramientas Utilizadas:

* Gestión de Proyectos: Jira/Trello
* Control de Versiones: Git, GitHub
* Comunicación: Slack, Zoom

# **Descripción del Diseño y Arquitectura del Sistema**

El sistema está diseñado utilizando una arquitectura de microservicios. Esta arquitectura se caracteriza por dividir la aplicación en servicios pequeños e independientes que se comunican entre sí a través de APIs. A continuación se detallan los componentes y la arquitectura del sistema:

1. Componentes del Sistema:

* Servicio de Usuario:

Funcionalidad: Gestión de usuarios (registro, autenticación, edición, eliminación).

Tecnologías: Spring Boot, MongoDB.

* Servicio de Chat:

Funcionalidad: Gestión de chat en tiempo real (envío y recepción de mensajes, lista de usuarios conectados).

Tecnologías: Spring Boot, WebSockets.

* Frontend:

Funcionalidad: Interfaz de usuario para registro, login y chat.

Tecnologías: React, JavaScript, HTML, CSS.

1. Comunicación entre Servicios:

* Protocolo: RESTful APIs para la comunicación entre servicios.

WebSockets: Para la comunicación en tiempo real entre el servidor y los clientes.

1. Base de Datos:

Tecnología: MongoDB.

Estructura: Base de datos no relacional para almacenar la información de usuarios y mensajes.

1. Servidor:

Tecnología: Apache Tomcat para desplegar los microservicios de Spring Boot.

Escalabilidad: Capacidad para añadir más instancias de servicios según la demanda.

# **Requisitos Funcionales**

| **OBJ-01** | **Gestionar usuarios** |
| --- | --- |
| **Descripción** | Permitir la gestión completa de los usuarios, incluyendo registro, autenticación, edición y eliminación. |
| **Estabilidad** | Alta |
| **Comentarios** | Representa el microservicio de gestión de usuarios. |

| **REQ001** | **Agregar Usuario** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-01 Gestionar usuarios | |
| **Requisitos asociados** | ● IRQ-01 Crear e ingresar información sobre el usuario | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario se registra proporcionando su nombre y nick. | |
| **Precondición** | El usuario no debe estar registrado previamente. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El usuario accede a la página de login. |
|  | 2 | El usuario llena el formulario de inicio de sesión. |
|  | 3 | El sistema valida los datos ingresados. |
|  | 4 | El sistema guarda los datos en la base de datos. |
|  | 5 | El sistema confirma el registro exitoso. |
| **Postcondición** | El usuario queda registrado en el sistema. | |
| **Excepciones** | **Paso** | **Acción** |
|  | 3 | Si el correo electrónico ya está registrado, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 3 - 4 minutos | |
| **Frecuencia** | 6 vez/semana | |
| **Importancia** | Alta | |
| **Comentarios** | Ninguno. | |

| **REQ002** | **Autenticar usuario** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-01 Gestionar usuarios | |
| **Requisitos asociados** | ● IRQ-01 Validar y autenticar información del usuario | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario proporciona sus credenciales para acceder al sistema. | |
| **Precondición** | El usuario debe estar registrado previamente. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El usuario accede a la página de inicio de sesión. |
|  | 2 | El usuario ingresa su correo electrónico y contraseña. |
| 3 | El sistema valida las credenciales. |
| 4 | El sistema confirma la autenticación. |
| **Postcondición** | El usuario queda autenticado en el sistema. | |
| **Excepciones** | **Paso** | **Acción** |
|  | 3 | Si las credenciales son incorrectas, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 1-2 minutos | |
| **Frecuencia** | 12 veces/semana | |
| **Importancia** | Alta | |
| **Comentarios** | Ninguno | |

| **REQ003** | **Editar información del usuario** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-01 Gestionar usuarios | |
| **Requisitos asociados** | ● IRQ-01 Editar información del usuario | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario modifica los datos registrados en el sistema. | |
| **Precondición** | El usuario debe estar autenticado. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El usuario accede a la sección de perfil. |
|  | 2 | El sistema muestra los datos actuales del usuario. |
| 3 | El usuario modifica los datos. |
| 4 | El usuario guarda los cambios. |
|  | 5 | El sistema confirma los cambios. |
| **Postcondición** | La información del usuario se actualiza en el sistema. | |
| **Excepciones** | **Paso** |  |
|  | 3 | Si los datos no cumplen con los estándares de validación, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 2-3 minutos | |
| **Frecuencia** | 4 veces/semana | |
| **Importancia** | Media | |
| **Comentarios** | Ninguno | |

| **REQ004** | **Eliminar usuario** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-01 Gestionar usuarios | |
| **Requisitos asociados** | ● IRQ-01 Eliminar información del usuario | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario puede eliminar su cuenta del sistema. | |
| **Precondición** | El usuario debe estar autenticado. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El usuario accede a la sección de perfil. |
|  | 2 | El usuario selecciona la opción de eliminar cuenta. |
| 3 | El sistema solicita confirmación. |
| 4 | El usuario confirma la eliminación. |
| 5 | El sistema elimina la cuenta del usuario. |
| **Postcondición** | La cuenta del usuario se elimina del sistema. | |
| **Excepciones** | **Paso** | **Acción** |
|  | 4 | Si el usuario no confirma la eliminación, la cuenta no se elimina. |
| **Rendimiento** | **Cota de tiempo** | |
| 1-2 minutos | |
| **Frecuencia** | 1 vez/mes | |
| **Importancia** | Media | |
| **Comentarios** | Ninguno | |

**Objetivo del chat en vivo**

| **OBJ-02** | **Gestionar chat en vivo** |
| --- | --- |
| **Descripción** | Permitir la gestión completa del chat en vivo, incluyendo conexión de usuarios, envío y recepción de mensajes, y visualización de usuarios conectados. |
| **Estabilidad** | Alta |
| **Comentarios** | Representa el microservicio de gestión de chat. |

| **REQ005** | **Conectar usuario al chat** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-02 Gestionar chat en vivo | |
| **Requisitos asociados** | ● IRQ-01 Conectar usuario al chat | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario se conecta al chat en vivo. | |
| **Precondición** | El usuario debe estar autenticado. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El usuario accede a la página de chat. |
|  | 2 | El sistema establece una conexión WebSocket. |
|  | 3 | El usuario puede ver los mensajes globales y la lista de usuarios conectados. |
| **Postcondición** | El usuario está conectado al chat en tiempo real. | |
| **Excepciones** | **Paso** | **Acción** |
|  | 2 | Si la conexión WebSocket falla, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 1-2 minutos | |
| **Frecuencia** | 12 veces/semana | |
| **Importancia** | Alta | |
| **Comentarios** | Ninguno | |

| **REQ006** | **Recibir mensajes en tiempo real** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-02 Gestionar chat en vivo | |
| **Requisitos asociados** | ● IRQ-01 Recibir mensajes en tiempo real | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario recibe mensajes enviados por otros usuarios en tiempo real. | |
| **Precondición** | El usuario debe estar conectado al chat. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | Un usuario envía un mensaje. |
|  | 2 | El sistema transmite el mensaje a través de WebSocket. |
| 3 | El sistema recibe y muestra el mensaje en la interfaz de usuario. |
| **Postcondición** | El mensaje se muestra en el chat para el usuario. | |
| **Excepciones** | **Paso** |  |
|  | 2 | Si la conexión WebSocket se pierde, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 1-2 segundos | |
| **Frecuencia** | 60 veces/semana | |
| **Importancia** | Alta | |
| **Comentarios** | Ninguno | |

| **REQ007** | **Mostrar lista de usuarios conectados** | |
| --- | --- | --- |
| **Objetivos asociados** | ● OBJ-02 Gestionar chat en vivo | |
| **Requisitos asociados** | ● IRQ-01 Mostrar lista de usuarios conectados | |
| **Actor** | ● ACT-01 | |
| **Descripción** | El usuario ve la lista de todos los usuarios conectados al chat en tiempo real. | |
| **Precondición** | El usuario debe estar conectado al chat. | |
| **Secuencia normal** | **Paso** | **Acción** |
|  | 1 | El sistema actualiza la lista de usuarios conectados en tiempo real. |
|  | 2 | El usuario ve la lista actualizada de usuarios conectados. |
| **Postcondición** | La lista de usuarios conectados se muestra correctamente. | |
| **Excepciones** | **Paso** | **Acción** |
|  | 1 | Si la conexión WebSocket se pierde, el sistema muestra un mensaje de error. |
| **Rendimiento** | **Cota de tiempo** | |
| 1-2 segundos | |
| **Frecuencia** | 30 veces/semana | |
| **Importancia** | Alta | |
| **Comentarios** | Ninguno | |

# 

# **Requisitos No Funcionales**

1. Rendimiento

* Descripción: El sistema debe ser capaz de manejar múltiples usuarios simultáneamente sin una degradación significativa del rendimiento.
* Métrica: El tiempo de respuesta para el envío y recepción de mensajes no debe exceder los 500 ms bajo condiciones de carga máxima.

1. Escalabilidad

* Descripción: El sistema debe ser escalable para soportar un creciente número de usuarios sin necesidad de rediseño significativo.
* Métrica: El sistema debe soportar la adición de más nodos en la infraestructura backend para manejar un aumento en el tráfico de usuarios.

1. Confiabilidad

* Descripción: El sistema debe tener un tiempo de actividad (uptime) de al menos 99.9%.
* Métrica: El tiempo de inactividad no planificado no debe exceder los 8.76 horas al año.

1. Seguridad

* Descripción: El sistema debe asegurar la privacidad y protección de los datos de los usuarios.
* Métrica: Implementación de autenticación segura (por ejemplo, OAuth 2.0) y cifrado de datos en tránsito y en reposo.

1. Usabilidad

* Descripción: La interfaz de usuario debe ser intuitiva y fácil de usar para todos los usuarios, independientemente de su nivel técnico.
* Métrica: Realizar pruebas de usabilidad con al menos 10 usuarios y obtener una puntuación de satisfacción de al menos 4 de 5.

1. Compatibilidad

* Descripción: El sistema debe ser compatible con los navegadores web modernos y dispositivos móviles.
* Métrica: El sistema debe funcionar correctamente en las últimas versiones de Chrome, Firefox, Safari y Edge, así como en dispositivos iOS y Android.

1. Mantenibilidad

* Descripción: El código del sistema debe ser modular y seguir las mejores prácticas de desarrollo para facilitar su mantenimiento y actualización.
* Métrica: Implementación de pruebas unitarias con una cobertura de al menos el 80% del código.

1. Disponibilidad

* Descripción: El sistema debe estar disponible 24/7, excepto durante períodos programados de mantenimiento.
* Métrica: Programar mantenimientos fuera de las horas pico y notificar a los usuarios con al menos 24 horas de anticipación.

1. Documentación

* Descripción: El sistema debe estar bien documentado para facilitar su uso y mantenimiento.
* Métrica: Proveer documentación completa para desarrolladores y usuarios, incluyendo guías de inicio rápido, API docs y manuales de usuario.

1. Localización e Internacionalización

* Descripción: El sistema debe soportar múltiples idiomas y ser adaptable a diferentes regiones.
* Métrica: Implementar soporte para al menos dos idiomas adicionales (por ejemplo, español e inglés) y permitir la fácil adición de más idiomas en el futuro.

# **Casos de Uso**

1. Caso de Uso 1: Registrar nuevo usuario

Actor: Usuario

Descripción: El usuario se registra proporcionando su nombre, correo electrónico y contraseña.

Precondición: El usuario no debe estar registrado previamente.

Secuencia Normal:

* El usuario accede a la página de registro.
* El usuario llena el formulario de registro.
* El sistema valida los datos ingresados.
* El sistema guarda los datos en la base de datos.
* El sistema confirma el registro exitoso.

Postcondición: El usuario queda registrado en el sistema.

Excepciones:

* Si el correo electrónico ya está registrado, el sistema muestra un mensaje de error.

1. Caso de Uso 2: Conectar usuario al chat

Actor: Usuario

Descripción: El usuario se conecta al chat en vivo.

Precondición: El usuario debe estar autenticado.

Secuencia Normal:

* El usuario accede a la página de chat.
* El sistema establece una conexión WebSocket.
* El usuario puede ver los mensajes globales y la lista de usuarios conectados.

Postcondición: El usuario está conectado al chat en tiempo real.

Excepciones:

* Si la conexión WebSocket falla, el sistema muestra un mensaje de error.

1. Caso de Uso 3: Enviar mensaje global

Actor: Usuario

Descripción: El usuario envía un mensaje que será visible para todos los usuarios conectados.

Precondición: El usuario debe estar conectado al chat.

Secuencia Normal:

* El usuario escribe un mensaje en el cuadro de texto.
* El usuario presiona el botón de enviar.
* El sistema envía el mensaje a todos los usuarios conectados mediante WebSocket.
* Los usuarios conectados reciben y visualizan el mensaje en tiempo real.

Postcondición: El mensaje se muestra en el chat para todos los usuarios.

Excepciones:

Si la conexión WebSocket se pierde, el sistema muestra un mensaje de error.

1. Caso de Uso 4: Recibir mensajes en tiempo real

Actor: Usuario

Descripción: El usuario recibe mensajes enviados por otros usuarios en tiempo real.

Precondición: El usuario debe estar conectado al chat.

Secuencia Normal:

* Un usuario envía un mensaje.
* El sistema transmite el mensaje a través de WebSocket.
* El sistema recibe y muestra el mensaje en la interfaz de usuario.

Postcondición: El mensaje se muestra en el chat para el usuario.

Excepciones:

* Si la conexión WebSocket se pierde, el sistema muestra un mensaje de error.

1. Caso de Uso 5: Mostrar lista de usuarios conectados

Actor: Usuario

Descripción: El usuario ve la lista de todos los usuarios conectados al chat en tiempo real.

Precondición: El usuario debe estar conectado al chat.

Secuencia Normal:

* El sistema actualiza la lista de usuarios conectados en tiempo real.
* El usuario ve la lista actualizada de usuarios conectados.

Postcondición: La lista de usuarios conectados se muestra correctamente.

Excepciones:

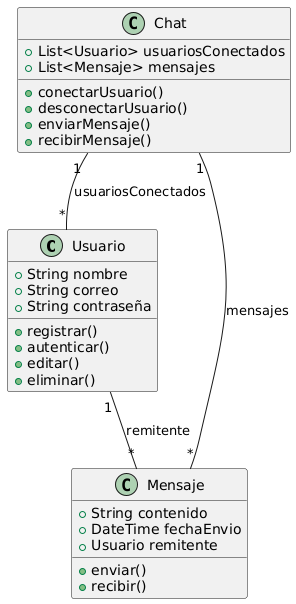
* Si la conexión WebSocket se pierde, el sistema muestra un mensaje de error.

# **Herramientas de Desarrollo**

* Backend: Spring Boot, WebSockets, Java
* Frontend: React, JavaScript, HTML, CSS
* Base de Datos: MongoDB
* IDE: IntelliJ IDEA, Visual Studio Code
* Control de Versiones: Git, GitHub

# **Diseño y arquitectura del Sistema**

# Diagrama de clases



# Diagrama de Secuencia

# 

# Diagrama de Componentes

# 

# 

# **Resultados Obtenidos**

Al finalizar el proyecto, se logró desarrollar un sistema de chat en tiempo real que permite a los usuarios registrarse, autenticarse y participar en conversaciones globales. El sistema utiliza WebSockets para asegurar una comunicación eficiente y en tiempo real, y la arquitectura basada en microservicios facilita la escalabilidad y mantenimiento del sistema.

# **Interfaces del Aplicativo**

* Pantalla de Registro: Formulario para que los usuarios se registren.
* Pantalla de Login: Formulario para que los usuarios se autentiquen.
* Pantalla de Chat: Interfaz principal del chat, mostrando mensajes y usuarios conectados.

# **Conclusiones**

El proyecto demostró ser exitoso en la implementación de un sistema de chat en tiempo real utilizando tecnologías modernas como Spring Boot y React. La utilización de WebSockets permitió una comunicación eficiente y en tiempo real entre los usuarios. Se lograron todos los objetivos planteados, y se establecieron las bases para futuras mejoras y expansiones del sistema.

# **Recomendaciones(mejoras futuras)**

* Implementar mensajes privados entre usuarios.
* Añadir soporte para envío de archivos.
* Mejorar la seguridad del sistema mediante cifrado de extremo a extremo.
* Optimizar el rendimiento del sistema para manejar un mayor número de usuarios concurrentes.

Link repositorio Github:

<https://github.com/SantiagoQuishpe312/MicroServicios_SegundoParcial.git>