
Memoria técnica del proyecto

Tecnologías y modelos para el desarrollo de aplicaciones distribuidas

Samuel Pérez Pedrajas 779333@unizar.es
Hong Christian Lin Jiang 758631@unizar.es

Mayo 2023

1. Introducción

Este documento es la memoria técnica de la aplicación de chat distribuida desarrollada como proyecto de asignatura por Samuel Pérez Pedrajas y Hong Christian Lin Jiang. El objetivo de nota de este proyecto es de notable desarrollando para ello el analizador de tendencias como objetivo adicional.

El proyecto ha sido liderado por Samuel Pérez Pedrajas que ha desarrollado la mayor parte del código, Hong Christian Lin Jiang ha desarrollado algunos módulos siguiendo las indicaciones de Samuel Pérez Pedrajas.

Todo el código del proyecto se puede encontrar en el siguiente [repositorio](#).

La Sección 2 describe en detalle las funcionalidades implementadas, la Sección 3 explica la arquitectura del diseño implementado, la Sección 4 explica los métodos de prueba que se han utilizado durante el proyecto y la 5 expone las conclusiones extraídas del desarrollo.

2. Funcionalidad

Esta sección describe las funcionalidades que se han implementado, indicando que historias de usuario detalladas en los requisitos del proyecto se han completado.

2.1. Historias de usuario implementadas

Para obtener el objetivo de nota se han implementado las historias de usuario 1, 2, 3, 5 y 6.

- **1. Como usuario quiero poder enviar mensajes a otros usuarios para tener conversaciones puntuales.**

El sistema permite a los usuarios enviar mensajes de texto de hasta 500 caracteres a otros usuarios solamente conociendo su nombre, que actúa como identificador único. Los usuarios cuando se conectan reciben de golpe todos los mensajes se les hayan enviado cuando estuvieran desconectados. Cuando los usuarios están conectados reciben los mensajes sin tener que refrescar la página. Los mensajes enviados de esta forma no se almacenan en el sistema una vez recibidos por el usuario final, por lo que no se pueden recuperar.

- **2. Como usuaria quiero poder compartir ficheros de mi equipo con otros usuarios para complementar las conversaciones con información adicional.**

El sistema permite a los usuarios enviar mensajes de tipo fichero a otros usuarios. El sistema permite enviar ficheros de hasta 20 MB que quedan almacenados en uno de los servidores de la aplicación de forma que son accesibles mediante un enlace. Los mensajes de este tipo cumplen los mismos requisitos que los mensajes de la historia 1 pero en vez de contener texto contienen el enlace que da acceso al fichero.

- **3. Como usuario quiero poder crear salas de chat permanentes para tener un registro de las conversaciones pasadas.**

El sistema permite a los usuarios crear salas de chat con un nombre que actúa como identificador único. El usuario que crea la sala es el único que tiene permisos de administración y puede añadir a otros usuarios, eliminarlos o borrar la sala. Los mensajes enviados a las salas de chat cumplen los requisitos detallados para las historias 1 y 2 pero además estos quedan almacenados en el sistema para que se puedan recuperar en un futuro. Las salas de chat son accesibles mediante una URL única que permite a los miembros recuperar todos los mensajes enviados previamente a la sala.

- **5. Como superusuario quiero poder enviar mensajes a todos los usuarios suscritos para anuncios de interés y publicidad de nuevas características del sistema.**

El sistema define una cuenta de administración desde la cual se pueden enviar mensajes de texto que reciben todos los usuarios. Estos mensajes cumplen los mismos requisitos que los de la historia 1.

- **6. Como superusuario quiero saber en tiempo real de qué hablan los usuarios para poder calcular trending topics etc. y vender esa información al mejor postor.**

El sistema cada vez que un usuario envía un mensaje de texto almacena las palabras que no considera un conectoras. Cada 30 segundos calcula las 10 palabras más utilizadas durante la última hora y las envía a todos los usuarios conectados en ese momento al sistema.

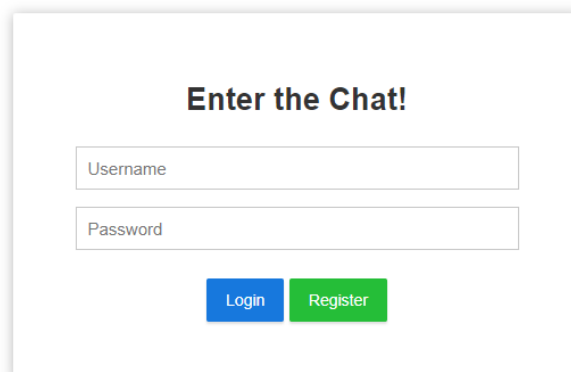
2.2. Requisitos no funcionales implementados

El sistema se despliega sobre 6 máquinas, 2 de ellas en proveedores cloud y utiliza comunicaciones seguras sobre TLS siempre que la comunicación atraviesa redes públicas. Uno de los componentes del sistema es un broker de mensajería.

2.3. Interfaz de usuario

Se ha implementado una aplicación web sencilla que actúa como interfaz de usuario.

La Figura 1 muestra el formulario de inicio de sesión o registro que aparece como página inicial cuando se accede mediante HTTPS a la URL del servicio.



Enter the Chat!

Username

Password

Login Register

Figura 1: Formulario de inicio de sesión.

Cuando se pulsa alguno de los 2 botones la aplicación realiza una petición al servidor para generar un token de sesión. Si recibe una respuesta válida almacena dicho token en el almacén local de la

página para utilizarlo en caso de accesos futuros. Tras ello la aplicación muestra la página que se muestra en la Figura 2.

The screenshot displays a web interface for a user's chat. At the top, a white box contains a red 'Logout' button on the left and the text 'Welcome to the Chat!' in the center. Below this is a thin white bar with the text 'Hello!'. The next section features two input fields labeled 'Group...' and 'User...', followed by a 'Create' button with a dropdown arrow. A green 'Send' button is positioned below these fields. The following section has a 'Destination...' input field and a 'Message...' text area. Below the text area is a file selection interface with a button labeled 'Seleccionar archivo' and the text 'Ninguno archivo selec.'. A green 'Send' button and a 'User' dropdown are at the bottom of this section. The final part of the interface is a large white box titled 'paco3's Inbox' with a vertical scrollbar on the right side.

Figura 2: Página del chat de un usuario.

La página cuenta con un primer panel en la parte superior que permite cerrar la sesión de usuario abierta, borrando los tokens de acceso del almacenamiento local y recargando la página.

Luego la página cuenta con un segundo panel que muestra las notificaciones recibidas del servidor. Las Figuras 3, 4 y 5 muestran diferentes estados de este panel.

The screenshot shows a single notification message in a white box. The text 'pacos created' is displayed in a blue, italicized font.

Figura 3: Notificación genérica.

A horizontal bar with a light gray background and a thin border. It contains the text "TRENDING: hello! | universidad" in a dark gray, sans-serif font.

Figura 4: Notificación de tendencias.

A horizontal bar with a light red background and a thin border. It contains the text "Error sending message to paco5" in a dark red, sans-serif font.

Figura 5: Notificación de error.

A continuación la aplicación muestra un panel que permite enviar diferentes operaciones relacionadas con las salas de chat. Este panel contiene un formulario en el que se debe introducir el nombre de la sala y en caso de que la operación involucre a algún usuario su nombre. La Figura 6 muestra dicho panel con las opciones de tipo de mensaje desplegadas.

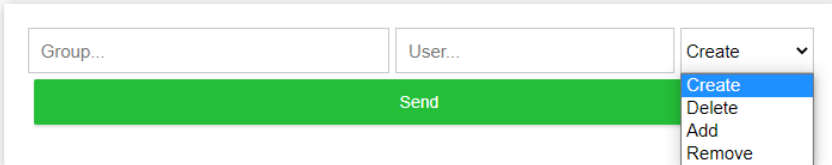
A web form for sending chat messages. It has two input fields: "Group..." and "User...". Below these is a large green "Send" button. To the right of the "User..." field is a dropdown menu currently showing "Create". The dropdown menu is open, showing options: "Create" (highlighted in blue), "Delete", "Add", and "Remove".

Figura 6: Panel para enviar mensajes relacionados con las operaciones de salas de chat.

La operación *Create* permite crear una sala y la operación *Delete* eliminarla. La operación *Add* permite añadir un usuario a una sala y la operación *Remove* expulsarlo.

La aplicación cuenta con un siguiente panel que permite enviar mensajes a usuarios o a salas de chat. Dicho panel permite elegir el destino del mensaje mediante el nombre del usuario o sala e introducir el texto o añadir el fichero que se quiera enviar. La Figura 7 muestra el panel con las opciones de tipo de destinatario desplegadas.

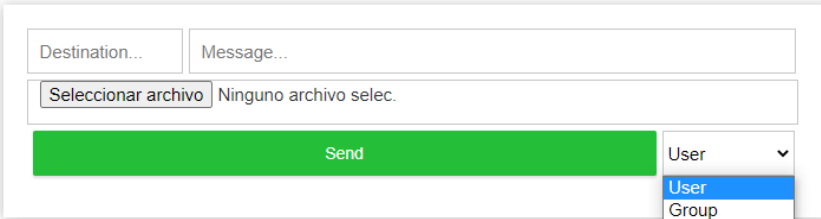
A web form for sending messages to destinations. It has two input fields: "Destination..." and "Message...". Below these is a button labeled "Seleccionar archivo" and the text "Ninguno archivo selec.". Below that is a large green "Send" button. To the right of the "Message..." field is a dropdown menu currently showing "User". The dropdown menu is open, showing options: "User" (highlighted in blue) and "Group".

Figura 7: Panel para enviar mensajes a destinatarios.

Por último la aplicación cuenta con un panel que muestra todos los mensajes que se reciben y se envían. La Figura 8 muestra un ejemplo de dicho panel con mensajes de texto punto a punto, a salas y ficheros. Los mensajes de tipo fichero se muestran como un botón que cuando se accionan descargan el archivo.

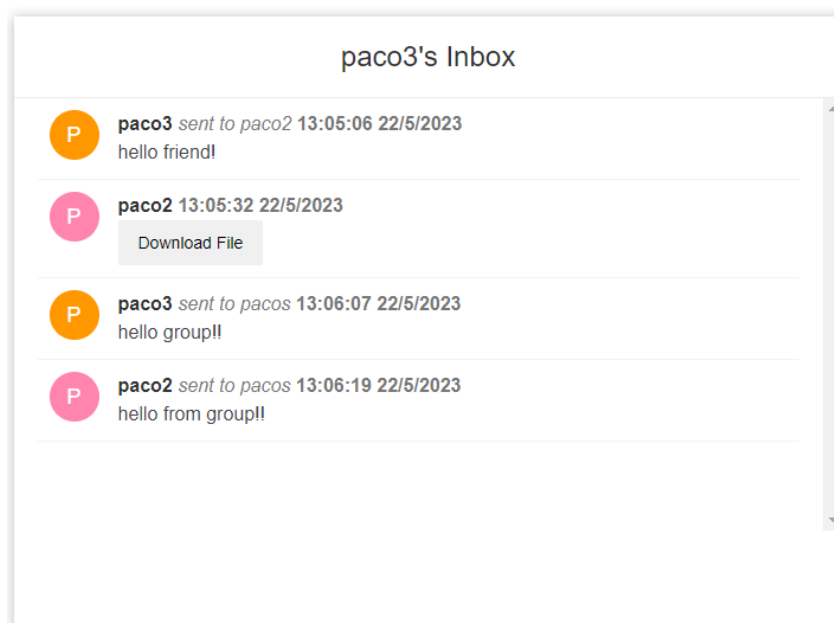


Figura 8: Panel que muestra todos los mensajes que ha recibido el usuario.

La aplicación web cuenta con otra funcionalidad, si en vez de acceder a la raíz de la dirección del servicio (<https://server/>) se accede al servicio de la siguiente forma <https://server/rooms/roomname> la aplicación solicitará los mensajes de la sala `roomname` al servidor y los mostrará como si los acabara recibir.

3. Arquitectura

En esta sección se explican las decisiones de diseño que se han tomado con respecto a la arquitectura del sistema junto con diagramas explicativos.

La arquitectura del sistema sigue el patrón N-tier, con un primer nivel Cliente, un segundo nivel Web, un tercer nivel de Dominio y un cuarto nivel de Datos.

3.1. Vista de Componentes y Conectores

La Figura 9 muestra un diagrama de componentes y conectores del sistema en ejecución.

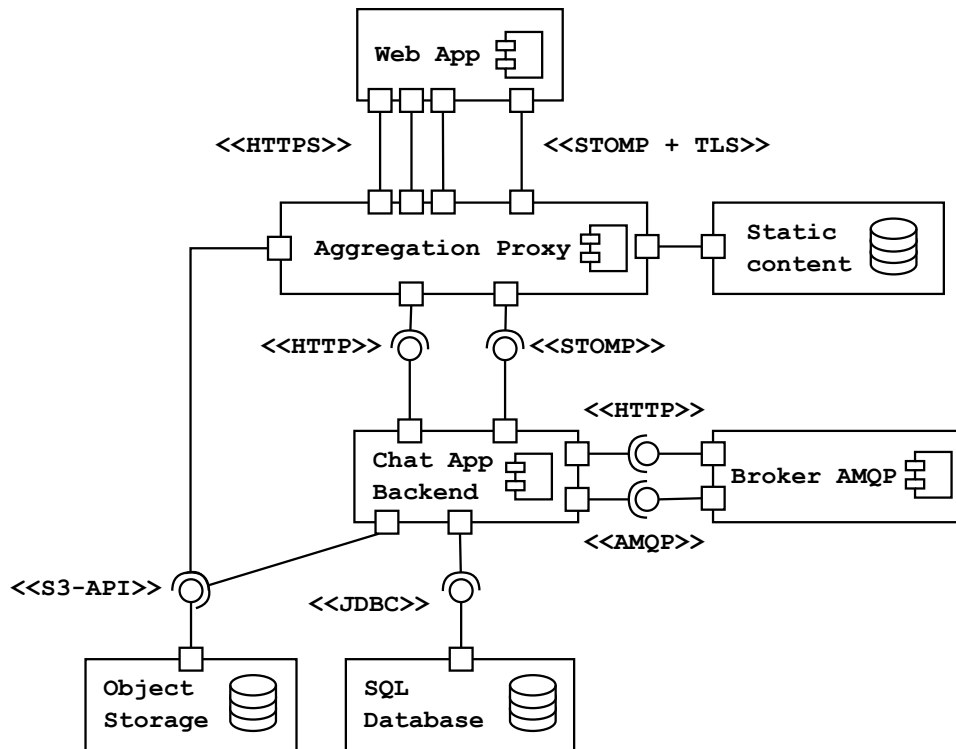


Figura 9: Diagrama de componentes y conectores del sistema en ejecución.

El componente Web App se ha explicado en la Sección 2.

El componente Aggregation Proxy se encarga de dar un punto de acceso único y protegido mediante TLS al resto de servicios del sistema. El componente redirige todas las peticiones HTTP a HTTPS.

Dependiendo de la URL de la petición que reciba, el componente actúa de la siguiente manera:

- /api/websocket: Permite el acceso al punto de conexión mediante Websockets del componente Chat App Backend.
- /api: Permite el acceso a la API HTTP ofrecida por el componente Chat App Backend.
- /minio: Permite el acceso a la API ofrecida por el componente Object Storage.
- /: El componente sirve contenido estático necesario (HTML, CSS y JavaScript) para que el cliente pueda ejecutar el componente Web App en su navegador.

El componente Chat App Backend implementa la mayoría de lógica de dominio de aplicación. Este componente ofrece una API sobre el protocolo HTTP y otra API sobre el protocolo de mensajería STOMP, el cual, funciona sobre Websockets.

La API HTTP ofrece los siguientes servicios:

- **POST /users/register:** Permite registrar a un usuario con un nombre y contraseña que el cliente debe enviar en formato JSON. El servidor devuelve un token JWT que se utilizará para identificar al usuario a lo largo de la sesión.
- **POST /users/login:** Permite iniciar la sesión de un usuario con un nombre y una contraseña que el cliente debe enviar en formato JSON. El servidor devuelve un token JWT que se utilizará para identificar al usuario a lo largo de la sesión.
- **GET /users/secured:** Permite comprobar la validez de un token de sesión, el cliente puede realizar una petición con un token en la cabecera de autenticación para comprobar si es válido. El servidor devolverá una respuesta 200 en caso de serlo.
- **GET /files/\$object:** Permite obtener un fichero con identificador \$object almacenado en el componente Object Storage. Si la petición cuenta con una cabecera de autenticación válida el servidor obtendrá el fichero del componente Object Storage y se lo enviará al cliente.
- **POST /admin/send:** Permite al administrador enviar un mensaje a todos los usuarios. El administrador debe enviar el contenido del mensaje en formato JSON. Si la petición cuenta con una cabecera de autenticación válida con el rol de administrador el servidor enviará el mensaje recibido al componente Broker AMQP.

La API sobre el protocolo de mensajería STOMP es la siguiente:

- **Establecer conexión:** Cuando un cliente establezca conexión mediante Websocket con el componente este deberá enviar una cabecera con un token de autenticación valido. Tras verificar el token la sesión de Websocket quedará validada.
- **Suscripciones a temas:** Una vez se haya establecido la conexión de formar correcta con el servidor el cliente deberá suscribirse a los siguientes temas:
 - **/topic/chat/\$username:** Los mensajes que reciba \$username serán publicados en este tema por el servidor.
 - **/topic/system/notifications/\$username:** Las notifica que reciba \$username serán publicadas en este tema por el servidor.
 - **/topic/system/trends/:** El servidor publicará las tendencias de forma periódica en este tema.
- **chat.start:** Indica al servidor que debe iniciar un proceso que consuma los mensajes pendientes para el usuario almacenados en el componente Broker AMQP.
- **chat.send:** Un cliente puede enviar un mensaje en formato JSON que el servidor enviará al destinatario mediante el componente Broker AMQP.
- **chat.group.get:** Permite a un cliente solicitar la lista de salas a las que pertenece. El servidor publicará la lista en el tema de notificaciones del usuario.

- `chat.group.create`: Permite a un cliente solicitar la creación de una sala. El cliente debe enviar en formato JSON el nombre deseado para la sala. El servidor enviará el resultado de la operación al usuario mediante su tema de notificaciones.
- `chat.group.delete`: Permite a un cliente propietario de una sala solicitar el borrado de la misma. El cliente debe enviar en formato JSON el nombre de la sala a borrar. El servidor enviará el resultado de la operación al usuario mediante su tema de notificaciones.
- `chat.group.add`: Permite a un cliente añadir a un usuario a una sala de la que es administrador. El cliente debe enviar en formato JSON el nombre de la sala y el nombre del usuario. El servidor enviará el resultado de la operación al usuario mediante su tema de notificaciones y también notificará al otro usuario.
- `chat.group.remove`: Permite a un cliente expulsar a un usuario de una sala de la que es administrador. El cliente debe enviar en formato JSON el nombre de la sala y el nombre del usuario. El servidor enviará el resultado de la operación al usuario mediante su tema de notificaciones y también notificará al otro usuario.

El componente Broker AMQP se encarga de mantener la persistencia de los mensajes en vuelo entre los usuarios. El componente mantiene una cola persistente por cada usuario dado de alta en el sistema. El componente también mantiene la lista de los miembros de cada sala de forma indirecta ya que cada sala se almacena como una centralita de tipo Fanout y la pertenencia a las mismas como un enlace entre la cola del usuario y la centralita de la sala. La Figura 10 muestra un diagrama explicativo.

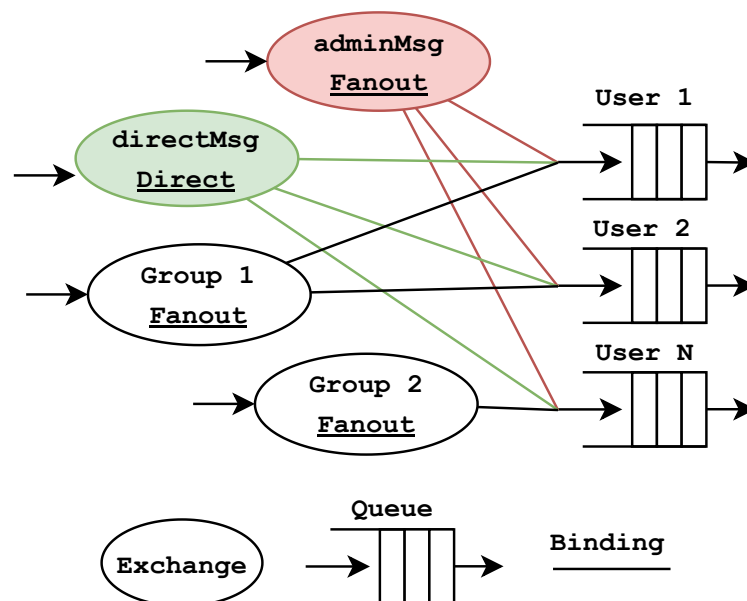


Figura 10: Diagrama de centralitas y colas del Broker AMQP.

Todas las colas de los usuarios se enlazan con una centralita de tipo Fanout a la que se envían los mensajes del administrador y con una centralita de tipo Direct que utiliza como clave de enrutado el nombre del usuario.

El componente Chat App Backend envía los mensajes que recibe de los usuarios como cadenas de texto en formato JSON a la centralita correspondiente del componente Broker AMQP mediante el protocolo de mensajería AMQP, en caso de que sea un mensaje directo punto a punto a la centralita `directMsg`, en caso de que vaya a un grupo a la centralita del grupo y en caso de que sea un mensaje

del administrador a la centralita adminMsg.

El componente Broker AMQP también ofrece una API HTTP que permite al componente Chat App Backend consultar la lista de centralitas, enlaces y colas activas.

El componente Object Storage se encarga de almacenar ficheros con un identificador único. Este componente ofrece una API compatible con la API S3 de Amazon, de forma que se puede acceder a los objetos almacenados de forma sencilla mediante peticiones HTTP. Los clientes pueden acceder directamente a la PI de este componente con enlaces pre-firmados por el componente Chat App Backend para almacenar objetos.

El componente SQL Database almacena los datos relacionados con las cuentas de usuario, salas, mensajes persistentes y tendencias. El componente Chat App Backend se comunica con este componente mediante el protocolo JDBC. La Figura 11 muestra las tablas de la base de datos.

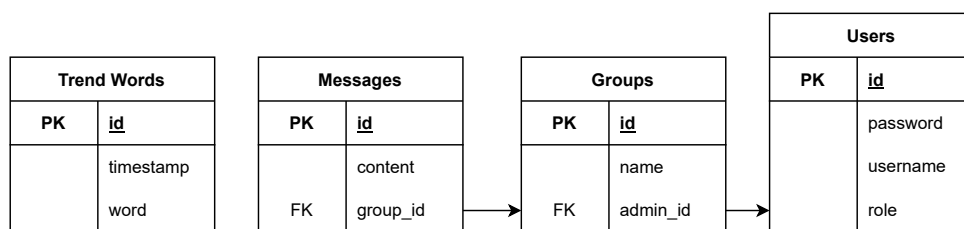


Figura 11: Estructura de tablas del componente SQL Database.

3.2. Vista de Distribución y Despliegue

La Figura 12 muestra un diagrama de despliegue. Se muestran las tecnologías utilizadas mediante iconos.

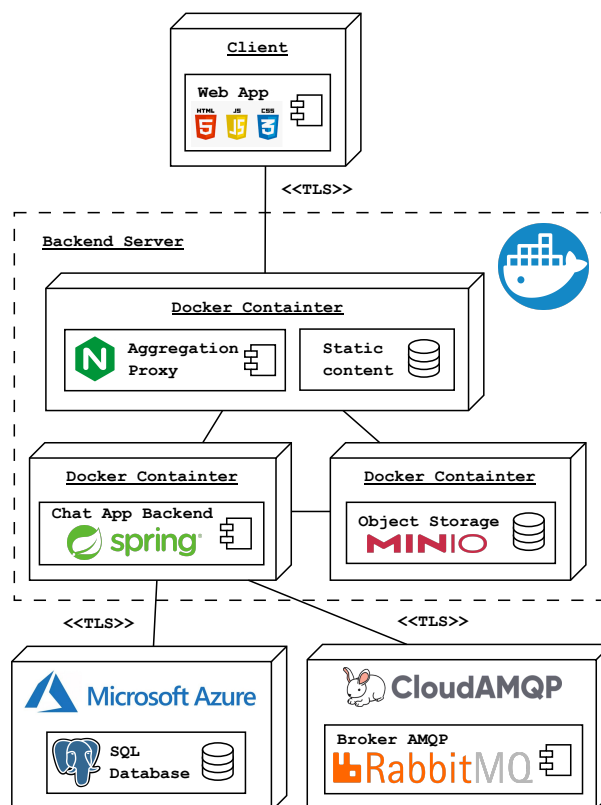


Figura 12: Diagrama de despliegue del sistema.

El despliegue final del sistema cuenta con 4 máquinas físicas, la máquina del cliente, un servidor que virtualiza mediante contenedores de Docker 3 componentes y 2 servicios cloud. Se han inyectado los datos pertinentes a todos los componentes mediante variables de entorno.

Para implementar el componente *Aggregation Proxy* se ha utilizado el servidor web Nginx debido a su elevado uso en la industria y la gran cantidad de documentación y ejemplos disponibles.

Para desarrollar el componente *Chat App Backend* se ha utilizado Spring siguiendo los requisitos del proyecto.

Para implementar el componente *Object Storage* se ha utilizado MINIO, debido a que es sencillo de desplegar mediante docker y cuenta con librerías de Java que facilitan su uso desde el componente *Chat App Backend*.

Se ha elegido PostgreSQL como gestor de base de datos para el componente *SQL Database* debido a la familiaridad de uso con el de los miembros del equipo. Se ha utilizado Microsoft Azure como proveedor cloud para desplegar este componente por que se dispone de crédito gratuito en la plataforma.

Se ha utilizado RabbitMQ como *Broker AMQP* por que se habían mostrado ejemplos de uso en clase, cuenta con una librería bien documentada para Java y se podía desplegar de forma gratuita en la plataforma CloudAMQP.

3.3. Patrones EIP

Se han utilizado los siguientes patrones EIP en el proyecto:

- **Publicación-Suscripción:** La comunicación sobre el protocolo STOMP enviada desde el servidor a los clientes está basada en este patrón.
- **Entrega garantizada:** Se han configurado las colas de los usuarios dentro del broker de mensajería de forma persistente de forma que se comporten siguiendo este patrón.
- **Mensajes comando:** La comunicación desde el cliente al servidor mediante el protocolo STOMP se realiza mediante mensajes comando.
- **Mensaje evento:** Se ha seguido este patrón para implementar las notificaciones enviadas desde el servidor a los clientes.
- **Petición-Respuesta:** Este patrón se da en el caso de las operaciones relacionadas con las salas de chat. El cliente solicita una operación mediante un mensaje comando y recibe la respuesta mediante su tema de notificaciones.
- **Lista de receptores:** Se ha implementado este patrón para conseguir la comunicación en las salas de chat, cuando un mensaje se envía a una centralita de grupo el broker de mensajería distribuye entre todas las colas enlazadas con la centralita el mensaje que se configuran de forma dinámica.
- **Comprobante de reclamación:** Se ha implementado este patrón para el envío de los ficheros por parte de los usuarios. Cuando el usuario envía un fichero el servidor le devuelve una URL pre-firmada para que lo almacene en el sistema de objetos y el servidor envía como mensaje a los destinatarios la URL de lectura de donde dicho fichero va a ser almacenado.

4. Pruebas

Para verificar que el sistema funciona correctamente se han ido realizando pruebas de las funciones conforme se iban implementando. Estas pruebas se han realizado de forma manual mediante la aplicación web desarrollada. A continuación se muestra una lista de pruebas para verificar que se cumplen los requisitos del sistema.

1 Prueba de registro de Usuario [General]

Verificar que se puede crear una cuenta correctamente y se puede cerrar e iniciar sesión con la misma. Verificar que no se puede iniciar sesión con credenciales erróneas.

2 Prueba de envío de mensajes de texto entre 2 usuarios [HU 1]

Con sesiones de 2 usuarios abiertas a la vez enviar mensajes de texto punto a punto entre ambos y verificar que se reciben de forma correcta.

Con la sesión de un usuario abierta y la del otro cerrada enviar mensajes al usuario con la sesión cerrada. Verificar que se reciben todos en cuando el usuario inicia la sesión.

Verificar que el usuario no recibe mensajes que no van dirigidos a él.

Comprobar que no se pueden enviar mensajes de texto de más de 500 caracteres.

3 Prueba de envío de ficheros entre 2 usuarios [HU 2]

Repetir todas las pruebas realizadas en 2 pero utilizando ficheros.

Comprobar que no se pueden enviar ficheros de más de 20 MB.

4 Prueba de salas de chat [HU 3]

Crear una sala de chat con un usuario y comprobar que el usuario creador puede enviar y recibir mensajes de la misma.

Añadir a otro usuario y verificar que recibe los mensajes enviados a la sala, repitiendo las pruebas 2 y 3.

Verificar que un miembro de la sala puede consultar la lista completa de mensajes.

Verificar que si se expulsa al usuario de sala deja de recibir los mensajes y que los usuarios no miembros no los reciben.

5 Prueba de permisos en salas de chat [HU 3]

Crear una sala y verificar que solo el administrador puede añadir nuevos usuarios a la sala y expulsarlos.

Verificar que solo el administrador puede eliminar la sala creada.

6 Prueba de mensajes del administrador [HU 5]

Enviar un mensaje a todos los usuarios desde la cuenta de administrador y verificar que todos lo reciben. Se debe probar que los usuarios con la sesión abierta lo reciben directamente y los usuarios con la sesión cerrada en cuanto la inician.

7 Verificación del comportamiento correcto de la base de datos [General]

Repetir las pruebas **1**, **4** y **5** verificando que las operaciones se reflejan correctamente en la base de datos.

- Comprobar que aparecen las entradas de los usuarios nuevos.
- Comprobar que aparecen las entradas de las salas nuevas.
- Comprobar que los mensajes enviados a la sala se almacenan correctamente.

8 Prueba de tendencias [HU 6]

Enviar mensajes y verificar que las palabras utilizadas se reflejan en las tendencias.

Esperar una hora y verificar con nuevos mensajes que las tendencias cambian.

9 Prueba con múltiples usuarios simultáneos [General]

Probar que el sistema soporta sin fallos el uso simultáneo de 5 usuarios dejándoles libertad total de actuación.

5. Conclusiones

El resultado final de este proyecto es una aplicación de chat distribuida en 6 componentes, que se pueden desplegar tanto localmente utilizando contenedores de Docker como distribuidos utilizando servicios cloud. La aplicación implementada cumple con los objetivos necesarios para alcanzar el objetivo de nota planteado. Las pruebas realizadas verifican que se cumplen los requisitos funcionales y no funcionales detallados en las historias de usuario. El resto de requisitos no funcionales se pueden verificar con los informes de despliegue incluidos previamente en este documento.

A. Informe de dedicación de horas

La siguiente tabla recoge las horas dedicadas al proyecto por Samuel Pérez Pedrajas.

Actividad	Horas Dedicadas
Reuniones	4:00:00
Redacción de documentos	14:45:00
Lectura de documentación	10:35:00
Pruebas	14:30:00
Controladores Spring	8:00:00
Registro de Usuarios y Seguridad	12:00:00
Integración MINIO	10:10:00
Integración RabbitMQ	7:30:00
Base de datos	8:30:00
Docker y despliegue	7:30:00
Frontend Web	9:00:00
TOTAL	106:30:00

La siguiente tabla recoge las horas dedicadas al proyecto por Hong Christian Lin Jiang.

Actividad	Horas Dedicadas
Reuniones	4:00:00
Prueba inicial de RabbitMQ	4:00:00
Análisis de tendencias	8:00:00
TOTAL	16:00:00