universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

RS CHAT: APLICACIÓN WEB DE CHAT PARA COMUNICACIÓN EN TIEMPO REAL ENTRE ESTUDIANTES Y DOCENTES.

RS CHAT: REAL TIME CHAT WEB APPLICATION FOR STUDENTS AND TEACHERS COMMUNICATION.

Autor: Samuel Castrillo Domínguez

Tutor: Eva María Cuervo Fernández

Junio, 2023

UNIVERSIDAD DE LEÓN

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

ALUMNO: Samuel Castrillo Domínguez

TUTOR: Eva María Cuervo Fernández

TÍTULO: RS Chat: Aplicación web de chat para comunicación en tiempo real entre estudiantes y docentes.

TITLE: RS Chat: Real time chat web application for students and teachers communication.

CONVOCATORIA: Junio, 2023

RESUMEN:
El resumen reflejará las ideas principales de cada una de las partes del trabajo, pudiendo incluir un avance de los resultados obtenidos. Constará de un único párrafo y se recomienda una longitud no superior a 300 palabras. En cualquier caso, no deberá superar una página de longitud.

ABSTRACT:
Abstract will reflect the main ideas of each part of the work, including an advance of the results obtained. It will consist of a single paragraph and it is recommended a length not superior to 300 words. In any case, it should not exceed a page of length.

Palabras clave: Lorem, ipsum, dolor, sit, amet.

Firma del alumno:	**VºBº Tutor/es:**

Índice de contenidos

Índice de figuras

Índice de cuadros y tablas

Índice de bloques de código

Índice de diagramas UML

Glosario

Backend Es la parte de la aplicación que se ejecuta en el servidor.

Cluster Es un conjunto de servidores que trabajan juntos para realizar una tarea.

Docker Es una herramienta que permite crear contenedores que ejecutan servicios.

Docker Compose Es una herramienta que permite definir y ejecutar contenedores Docker.

Dockerfile Es un fichero que contiene las instrucciones para crear una imagen Docker.

Frontend Es la parte de la aplicación que se ejecuta en el navegador del usuario.

HTTP HyperText Transfer Protocol.

IDE Integrated Development Environment (Entorno de desarrollo integrado).

Instanciar Es la acción de crear un objeto en memoria principal.

JSON JavaScript Object Notation.

JVM Java Virtual Machine.

Script Es un programa que se ejecuta en un intérprete.

1. Introducción

1.1. MOTIVACIÓN

La idea de la aplicación surgió en el periodo final del curso 2021–2022. Se tenían diferentes opciones para realizar el backend de la aplicación. La primera opción era utilizar *NodeJS* con *Express*, tecnologías que se habían aprendido en la asignatura de *Aplicaciones Web*. Sin embargo, se decidió utilizar *Java* con *Spring Boot* debido a que se consideró que era una mejor forma de conectar con muchas más empresas en un futuro y una forma de aprender una nueva tecnología. Además, se consideró que era una tecnología más robusta y que se podía utilizar en muchos más proyectos. En cuanto a la parte frontend, se tenía claro que se utilizaría *React* debido a que se había aprendido en la misma asignatura y es más fácil de aprender y de emplear que otras tecnologías como *Angular* o *VueJS*. Este proyecto se comenzó en el verano de 2022 como un proyecto personal para aprender *Spring Boot* y se dedicaba el tiempo libre a implementar todas las funcionalidades que se recogen en este documento.

1.2. OBJETIVOS

En un principio, el objetivo de la aplicación era su integración de forma completa con la plataforma de la Universidad de León, para que tanto estudiantes y profesores pudieran acceder a ella. Sin embargo, debido a que era una idea demasiado ambiciosa, se ha optado por generalizarlo más a un chat de mensajes instantáneos para que cualquier persona con acceso a internet pueda utilizarlo (pero manteniendo la temática de estudiantes/profesores, en caso de llegarse a utilizar en un futuro). En cualquier chat, se permite compartir archivos, imágenes, vídeos, etc. con los demás usuarios. Se podrán tener varios chats abiertos a la vez, pudiendo cambiar entre ellos fácilmente (teniendo una disposición en pestañas, mostrando un pequeño icono con el número de mensajes que no han sido leídos todavía). Además, se podrán crear grupos de chat para que varias personas puedan comunicarse entre sí, pudiendo compartir un código de invitación para que otros usuarios se unan al grupo en específico. Estos canales podrán ser públicos o privados, pudiendo ser generados por cualquier usuario registrado en la aplicación. Los canales privados solo podrán ser vistos por los usuarios que tengan acceso a ellos (mediante invitación). Los canales públicos podrán ser vistos por cualquier

persona que tenga acceso a la aplicación. También se podrán enviar mensajes a una sola persona, pudiendo tener una conversación privada.

1.3. TECNOLOGÍAS Y HERRAMIENTAS

A continuación, se presenta una breve descripción de las tecnologías y herramientas más importantes entre todas las que se han utilizado para la elaboración de este trabajo:

- Java: es el lenguaje de programación utilizado para la implementación del backend de la aplicación.
- Spring Boot: es un framework de Java que permite la creación de aplicaciones web.
- React: es una biblioteca de JavaScript que permite la creación de interfaces de usuario mediante componentes.
- Vercel: es un servicio de hosting que permite el despliegue de la parte de frontend de la aplicación web.
- Git: es un sistema de control de versiones que ha facilitado el desarrollo de la aplicación desde diferentes ordenadores.
- GitHub: es una plataforma que permite el almacenamiento de repositorios de Git.
- IntelliJ IDEA: es el IDE que ha permitido la implementación de la aplicación de forma completa. Se ha utilizado para la creación de los proyectos de frontend y backend, así como para la elaboración de este documento mediante el uso de LaTeX.
- Maven: es un gestor de dependencias que facilita la descarga e integración de las librerías utilizadas en la aplicación. El fichero `pom.xml` tiene una etiqueta donde se añaden todas las dependencias a utilizar, proporcionando su nombre, versión e identificador.

1.4. PROBLEMAS

Durante el desarrollo de la aplicación han surgido varios problemas, sobre todo en la parte del despliegue a producción de la aplicación. En un comienzo, la aplicación se ha

desplegado en dos clusters con el plan gratuito de Heroku (uno para el frontend y otro para el backend), pero se ha tenido que desplegar de una manera alternativa debido a que este servicio no es gratuito desde del 28 de noviembre de 2022. Actualmente, el frontend está desplegado en Vercel y en cuanto al backend, se ha tenido una reunión con un encargado de los servicios en la nube de la empresa Platform.sh, pero no se ha llegado a ningún acuerdo para conseguir un plan gratuito para desplegar la aplicación. Debido a esto, se ha optado por desplegar el backend en un pequeño ordenador personal propio, estando siempre encendido y con una dirección IP estática. Esto no es una solución óptima, pero es la mejor que se ha encontrado hasta el momento.

2. Estado del arte

2.1. APLICACIONES SIMILARES

Se ha realizado una búsqueda de diferentes sitios web y aplicaciones que permitan la comunicación entre usuarios a través de un chat en tiempo real. Las diferentes páginas web y aplicaciones que se han encontrado se presentan a continuación, analizando sus características principales:

- **Chateagratis.net**: Esta página web permite a los usuarios elegir un apodo (que se escoge antes de iniciar las conversaciones) con el que entrar en el chat, por lo que no es necesaria la creación de una cuenta de usuario (aunque sí que se disponga de esta funcionalidad). La aplicación se divide en diferentes salas, cada una de las cuales tiene una temática diferente y se pueden mantener multiples salas abiertas al mismo tiempo. Los usuarios pueden unirse a las salas existentes en un listado, disponible al acceder a la aplicación o en la pestaña asignada para ello. La página web permite a los usuarios enviar solamente mensajes de texto (con ciertos formatos, como negrita, colores, etc.) y emoticonos, sin la posibilidad de adjuntar imágenes o archivos. Se pueden programar robots para que administren una sala y que envíen mensajes automáticos cada cierto tiempo (como el que se visualiza en las siguientes imágenes, designado con el prefijo @).

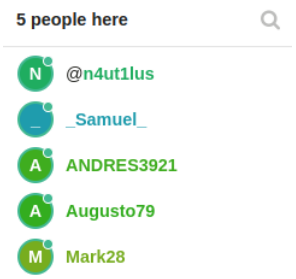

Figura 2.1: Lista de usuarios conectados en una sala de chat de chateagratis.net.

Figura 2.2: Mensajes de información del servidor en una sala de chat de chateagratis.net.

- **Dalechatea.me**: Esta página ofrece las mismas funciones que la anterior, pero con 2 diferencias: se puede modificar el tamaño del chat (ancho y alto) para adaptarlo a la pantalla y necesidades del usuario y se pueden mandar archivos multimedia temporales (imágenes, vídeos y audios), que se borran pasados 15 minutos. Además, se pueden agregar amigos, bloquear usuarios y añadir reacciones a mensajes de otros usuarios (como mandar un saludo o añadir a marcadores), entre otras funciones.

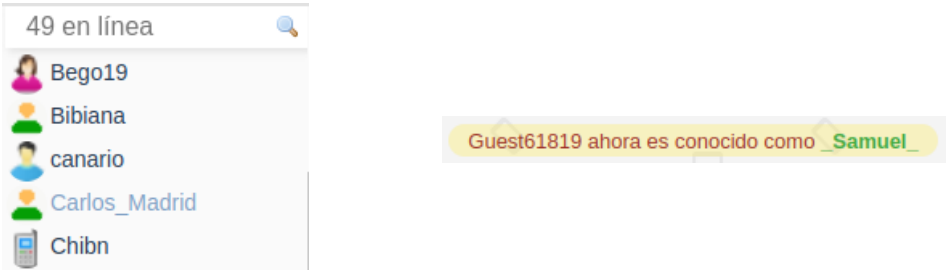

Figura 2.3: Lista de usuarios conectados y mensaje de información en una sala de chat de dalechatea.net.

Ambas páginas tienen un diseño similar, con el chat en la parte central de la pantalla y un listado de usuarios en la parte derecha (ver imagen 2.4). La página web de chateagratis.net tiene un diseño más sencillo y minimalista, mientras que la de dalechatea.me tiene un diseño más moderno y con más funcionalidades. Ambos tienen un sistema de notificaciones y mensajes de información, que se envían por parte del servidor del chat, para informar a todos los usuarios conectados al mismo. Cuando se hace clic en un usuario, se muestra, en la parte derecha de la lista de usuarios, una ventana de información del usuario seleccionado, que muestra su nombre de usuario y ciertos botones para realizar las acciones previamente mencionadas.

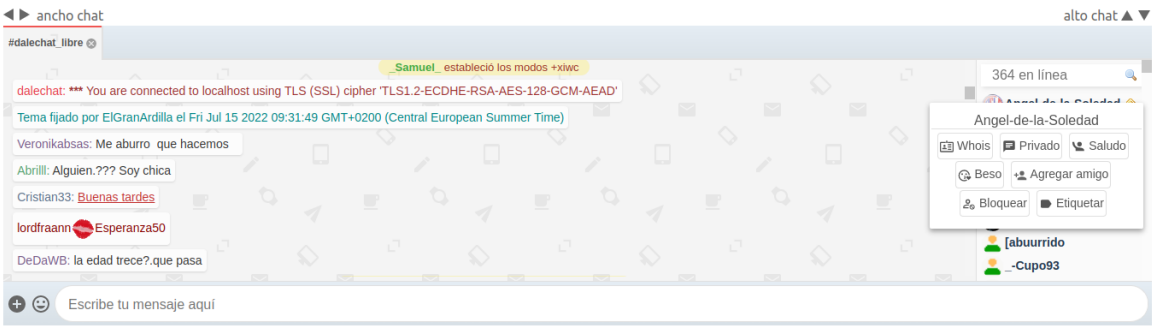

Figura 2.4: Chat completo de la página web de dalechatea.me.

2.2. OPINIÓN SOBRE EL ESTADO DEL ARTE

3. Contenido

3.1. PATRONES DE DISEÑO

Un patrón de diseño es una solución que se puede aplicar a diferentes contextos y que se puede reutilizar en diferentes proyectos.
Hay varias categorías de patrones de diseño, cada una con una finalidad diferente [1]:

- **Patrones de creación**: se utilizan para crear objetos de una forma flexible y reutilizando código existente.
- **Patrones estructurales**: se utilizan para convertir clases y objetos en estructuras más complejas.
- **Patrones de comportamiento**: se utilizan para definir la interacción entre objetos.

En este proyecto se han utilizado varios patrones de diseño, para permitir una mejor escalabilidad, mantenibilidad y reutilización del código. A continuación se detalla la siguiente información de los patrones utilizados:

- Definición / categoría.
- Explicación de cómo se ha implementado en el proyecto.
- Diagrama UML.
- Justificación de su uso en la aplicación.

3.1.1. BUILDER

Es un patrón de **creación** que permite instanciar objetos complejos de una forma sencilla. Se ha creado una interfaz (`Builder`) genérica para su reutilización en caso de ser necesaria para cualquier otra clase. Esta interfaz define el método `build()` que devolverá la instancia de un objeto con los datos establecidos previamente. El diagrama UML del patrón implementado es el siguiente:

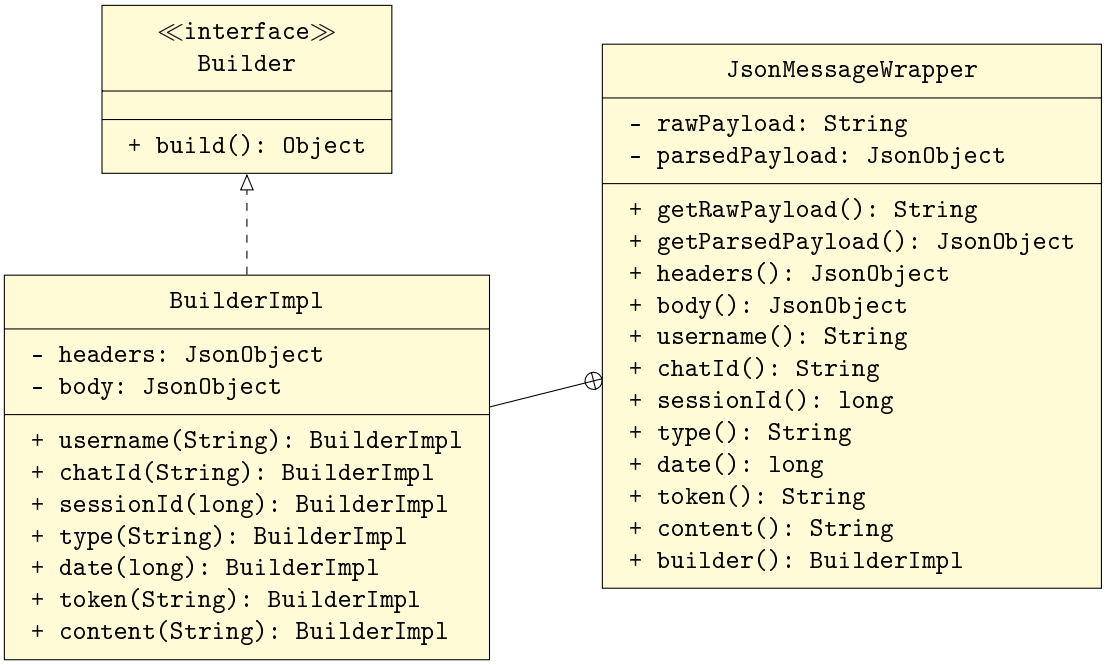

Diagrama UML 3.1: Patrón Builder empleado en la aplicación.

El patrón builder se ha usado en la aplicación para simplificar la creación de objetos de tipo `JsonMessageWrapper`, por el momento. Esta clase es la encargada de encapsular los mensajes que se envían a través de la red en formato JSON.

3.1.2. SINGLETON

También es un patrón de **creación** y se emplea para garantizar que una clase concreta tenga una única instancia y proporciona un punto de acceso global a ella [2]. En el contexto de esta aplicación, se utiliza en ciertas clases de utilidad y en las clases que asocian rutas a un método HTTP (por ejemplo `/login` con el método POST). Estas últimas son clases internas de `Routes.java` y los nombres dependen del método HTTP que se debe utilizar para realizar una petición a una ruta específica.

Cuadro 3.1: Relación entre método HTTP y ruta.

Método HTTP	Clase de la ruta
GET	`GetRoute`
POST	`PostRoute`
PUT	`PutRoute`
DELETE	`DeleteRoute`

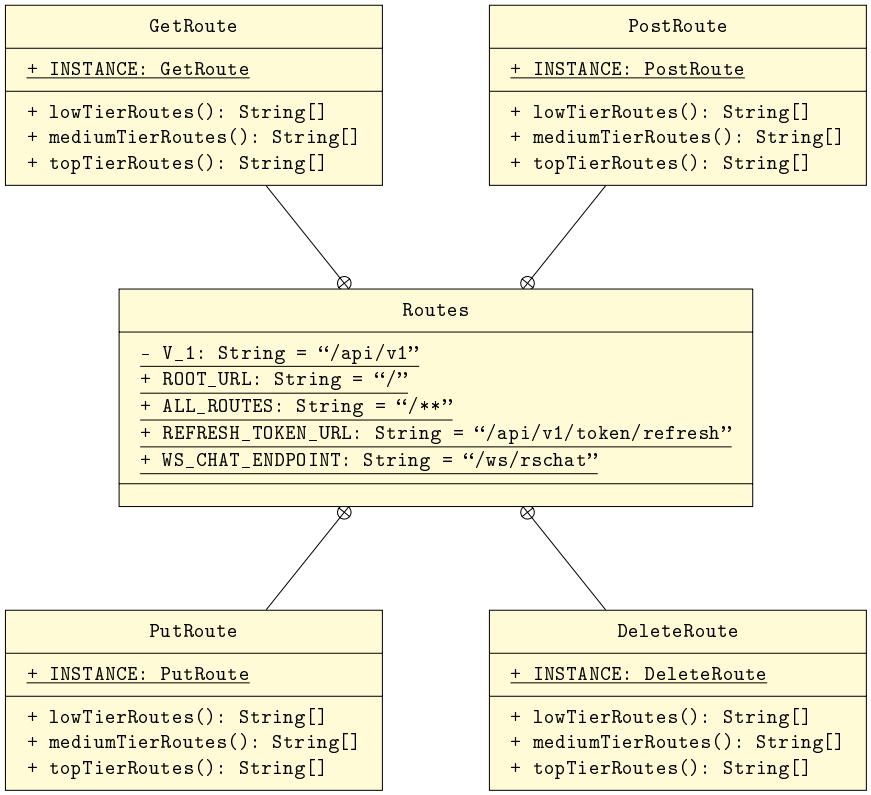

Diagrama UML 3.2: Patrón Singleton empleado en la aplicación.

Se han utilizado diferentes formas de acceso a la instancia de las clases. En el caso de las clases que asocian rutas a métodos HTTP, el modificador de acceso a la instancia es público. En otros casos, se provee un método estático para obtener la instancia de la clase.

3.1.3. STRATEGY

Este patrón de **comportamiento** se utiliza para encapsular un algoritmo dentro de una clase, de forma que pueda ser intercambiado por otro algoritmo en tiempo de ejecución. En la aplicación, se emplea para encapsular el proceso de manejo de los mensajes que se envían entre los usuarios. Existen varias clases que se encargan de realizarlo, por lo que todas ellas implementan la interfaz `MessageStrategy`, que define el método `handle(...)`, encargado de realizar el procesamiento del mensaje y recibe 3 parámetros:

- El mensaje que se debe manejar.
- La lista con todos los chats para determinar al que se debe enviar el mensaje.

- Otros datos que pueden ser necesarios para el manejo del mensaje. Dependiendo de la clase, este parámetro puede contener más o menos datos.

La relación entre el tipo de mensaje y las clases encargadas de procesarlo, que implementan esta interfaz, son las siguientes:

Cuadro 3.2: Relación Mensaje - Estrategia

Mensaje	**Clase de la estrategia**
`USER_JOINED`	`UserJoinedStrategy`
`USER_LEFT`	`UserLeftStrategy`
`TEXT_MESSAGE`	`TextMessageStrategy`
`IMAGE_MESSAGE`	`ImageMessageStrategy`
`AUDIO_MESSAGE`	`AudioMessageStrategy`
`VIDEO_MESSAGE`	`VideoMessageStrategy`
`ACTIVE_USERS_MESSAGE`	`ActiveUsersStrategy`
`GET_HISTORY_MESSAGE`	`GetHistoryStrategy`
`ERROR_MESSAGE`	`ErrorMessageStrategy`
`PING_MESSAGE`	`PingStrategy`

A continuación se muestra el diagrama UML de la interfaz `MessageStrategy`:

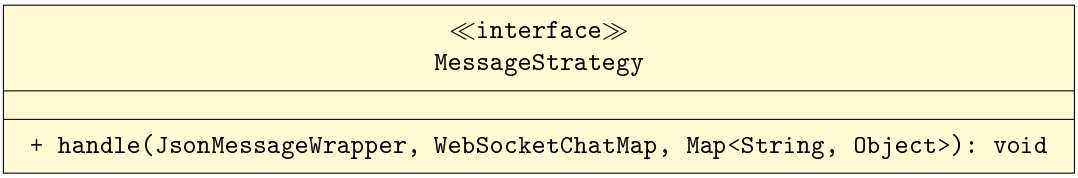

Diagrama UML 3.3: Interfaz MessageStrategy.

Este patrón se ha utilizado para simplificar el manejo de los mensajes recibidos en el servidor y para que sea más fácil de extender. En el caso de que se desee agregar un nuevo tipo de mensaje, se debe crear una nueva clase que implemente la interfaz `MessageStrategy` y agregarla a la lista de estrategias que se encuentran en la clase `WebSocketHandler`. Esta clase es la encargada de determinar qué estrategia se debe utilizar para manejar el mensaje. Para esto, se utiliza el método `decideStrategy(receivedMessageType: WSMessage)` que recibe como parámetro el tipo mensaje y devuelve la estrategia que se debe utilizar para manejar el mensaje.

3.2. PROCESAMIENTO DE LOS MENSAJES

Como hemos visto en la sección anterior, los mensajes que se reciben en el servidor pueden ser de diferentes tipos, cambiando la forma en que se procesan. A continuación se muestra una lista con las acciones que se realizan para cada tipo de mensaje que se trata:

- **Text, Image, Video y Audio**: se envían al resto de usuarios del chat en el mismo formato en que llegaron al servidor. Estos cuatro tipos de mensaje contienen texto exclusivamente, siendo un mensaje o el enlace a un archivo almacenado en el bucket de S3.
- **ActiveUsers**: se envía solo al cliente que lo ha solicitado, y contiene una lista con los usuarios que están conectados en ese momento al chat, ordenados alfabéticamente.
- **GetHistory**: se envía al usuario que lo solicita, y contiene una lista con los últimos 65 mensajes que se han enviado al chat. Se realiza una lectura del fichero de texto que contiene el historial de mensajes (almacenado en disco) y un filtrado de los mensajes de actividad (los 2 siguientes) del solicitante, ya que no son relevantes.
- **UserJoined y UserLeft**: se notifica a las personas conectadas el nombre del usuario que se ha unido o ha salido del chat,
- **Ping**: se envía un mensaje con un breve texto. Se utiliza exclusivamente para mantener la conexión WebSocket abierta.

3.3. CICLO DE VIDA DE LA CONEXIÓN DE USUARIOS

Cuando un usuario accede a un chat de la aplicación, se inicia una conexión entre el cliente y el servidor a través del protocolo de comunicación bidireccional **WebSocket** [3]. Esta conexión se mantiene abierta mientras el usuario esté en el chat, y se cierra cuando el usuario lo abandona. La secuencia de eventos que ocurren durante la conexión de un usuario al chat es la siguiente:

3.3.1. FRONTEND

1. Se realiza una solicitud de conexión WebSocket al servidor.
2. Se realiza una petición HTTP para comprobar que el usuario puede acceder al chat. Esto se realiza para que, en caso de que el usuario introduzca la URL de un chat al que no tiene acceso de forma manual, se le redirija a la página de inicio de la aplicación.
3. Si se confirma que el usuario **puede acceder** al chat:
   3.1. Se establece la conexión WebSocket.
   3.2. Se envía un mensaje de tipo `USER_JOINED` al servidor.
   3.3. Se consultan los últimos mensajes del historial de mensajes del chat con el mensaje de tipo `GET_HISTORY_MESSAGE`.
   3.4. Se realiza una petición de la lista de usuarios activos con un mensaje de tipo `ACTIVE_USERS_MESSAGE`.
   3.5. Se configura un temporizador para mandar un mensaje de tipo `PING_MESSAGE` cada 30 segundos. Esto se realiza para que el servidor no cierre la conexión por inactividad.
4. Si el usuario **no puede acceder** al chat:
   4.1. Se cierra la conexión WebSocket, en caso de llegarse a abrir.
   4.2. Se redirige al usuario a la página principal.

3.3.2. BACKEND

Cuando comienza la ejecución del programa, se indica a Spring Boot que el manejador de mensajes a través de WebSocket del servidor es una instancia de la clase `WebSocketHandler`, en la ruta '/ws/rschat'. Al instanciar esta clase, se crea un objeto **chatMap** de tipo `WebSocketChatMap`, que contiene el atributo `chats` (ver Diagrama UML 3.4). Es una tabla de dispersión que contiene los chats activos en la aplicación. Cada entrada asocia a una cadena de texto (identificador del chat) la instancia de un objeto de tipo `Chat`. El proceso de **conexión** al servidor sigue el siguiente flujo de eventos:

1. Se establece la conexión WebSocket con el usuario. Esto ocurre de forma transparente al programador debido a que la implementación se realiza en el framework de Spring Boot.

2. Se recibe el mensaje `USER_JOINED` del cliente y se crea un objeto de tipo `WSClient` (formado por la instancia de `WebSocketSession` y el `WSClientID` del usuario). Este nuevo objeto se añade a la lista de usuarios conectados al chat.

    2.1. Si el usuario es el primero que se ha conectado al chat, se crea uno nuevo, guardándose en la lista de chats.

    2.2. Si no, se añade al chat correspondiente de la lista de chats.

3. Se recibe el mensaje `GET_HISTORY_MESSAGE` del cliente y se envían como respuesta los últimos 65 mensajes del historial de mensajes del chat.

4. Se recibe el mensaje `ACTIVE_USERS_MESSAGE` del cliente y se devuelve la lista con los usuarios activos en el chat.

Y el proceso de **desconexión** se realiza como sigue:

1. Se recibe el mensaje `USER_LEFT` del cliente.

2. Se informa al resto de los usuarios del chat de la desconexión del usuario.

3. Al eliminar un usuario del chat se pueden dar 2 casos:

    3.1. Si el usuario es el último que se ha desconectado del chat, se elimina el chat de la tabla de dispersión `chats`. Cuando esto ocurre, el historial de mensajes del chat que se haya registrado desde que se inició, se envía al almacenamiento en la nube.

    3.2. Si no, se elimina el usuario de la lista de usuarios del chat.

4. Se cierra la conexión WebSocket con el usuario, de forma transparente al programador (al igual que la conexión).

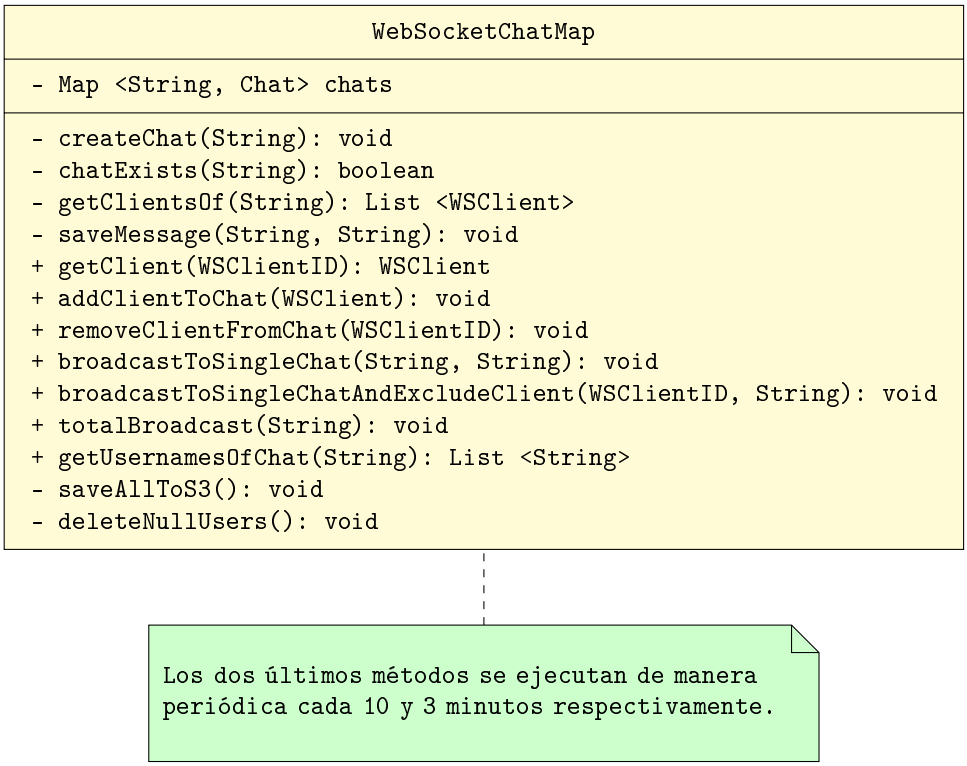

Diagrama UML 3.4: Clase `Chat` para almacenar los usuarios activos

3.4. DOCKER

Docker es una plataforma software que permite el desarrollo, prueba y ejecución de aplicaciones de forma rápida y cómoda, separando la aplicación de la infraestructura permitiendo ejecutar múltiples aplicaciones en un mismo servidor (ver imagen 3.1). Esto se realiza empaquetando la aplicación, sus dependencias y otras herramientas necesarias para su ejecución en lo que se denominan **contenedores**, que son instancias ejecutables de una imagen de Docker. Las imágenes son ficheros de solo lectura que contienen las instrucciones necesarias para crear un contenedor y normalmente se basan en otras imágenes añadiendo o modificando configuraciones [4].

En el desarrollo de la aplicación se ha utilizado Docker para la creación de una imagen que contenga todo el código de la aplicación así como las variables de entorno necesarias para su correcto funcionamiento. Se ha utilizado la herramienta `docker-compose`, que permite definir y ejecutar contenedores Docker de manera sencilla. Para la configuración de los contenedores, se dispone del fichero `docker-compose.yaml` que se encuentra en la raíz del proyecto y define los servicios que se ejecutarán en los contenedores, así como las dependencias entre ellos.

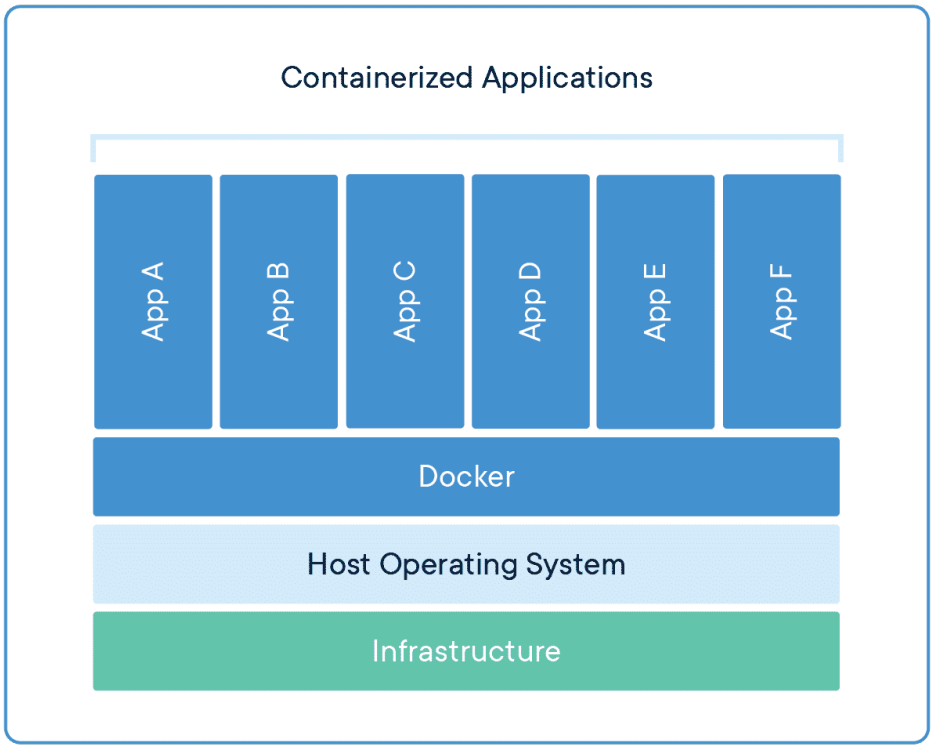

Figura 3.1: Infrastructura docker. Fuente: `https://www.docker.com/resources/what-container`

Debido a que cada vez que se elimina un contenedor (al reiniciar el sistema o al generar de nuevo la imagen de la aplicación para incluir cambios del código) se pierden los datos, la persistencia de los mismos y de sus configuraciones es necesaria. Para lograr esto, se usan **volúmenes**, que son directorios creados y gestionados por Docker que se almacenan en el sistema de archivos del host, y que se montan en los contenedores cuando se inicializan [5]. De esta forma, los datos no se pierden al eliminar el contenedor. Los volúmenes se definen en el mismo fichero `docker-compose.yaml` mediante la opción `volumes` y son los siguientes:

- `rschat-db`: contiene los datos y script inicial de la base de datos.
- `rschat-logs`: contiene los ficheros de registro de la aplicación.
- `grafana-storage`: contiene la configuración de Grafana.

A continuación, se mencionan todos los contenedores utilizados para la aplicación junto con una breve descripción de cada uno de ellos:

Nota: los marcados con el símbolo * se describirán en detalle más adelante.

- `rschat`: Contenedor que ejecuta el backend de la aplicación.
- `rschat-db`: Contenedor que ejecuta la base de datos MySQL.
- `prometheus`*: Contenedor que ejecuta el servidor de métricas de Prometheus.
- `grafana`*: Contenedor que ejecuta el panel de observabilidad de Grafana.
- `loki`*: Contenedor que ejecuta el servidor de logs Loki.
- `promtail`*: Contenedor que ejecuta el agente de logs Promtail.

3.4.1. PROMETHEUS (MÉTRICAS)

Prometheus es un conjunto de herramientas de código abierto para la **monitorización** de sistemas y **alertas** que recoge y almacena las métricas con la marca de tiempo cuando se producen, incluyendo etiquetas (clave-valor) de manera opcional [6]. Estas métricas se utilizan para determinar el funcionamiento y estado de la aplicación en tiempo real, permitiendo diagnosticar problemas de rendimiento o recursos de una manera rápida y sencilla. Para habilitar la exportación de las métricas (de manera automática) por parte de la aplicación hay que realizar ciertas configuraciones, que se detallan a continuación:

- Añadir 2 librerías en el fichero `pom.xml` [7]:
  - spring-boot-starter-actuator
  - micrometer-registry-prometheus
- Para exponer el 'endpoint' que permite ver las métricas, hay que añadir una propiedad en el fichero `application.properties`:
  - `management.endpoints.web.exposure.include=prometheus`
- Permitir las peticiones de tipo GET a `/actuator/prometheus`: esto se configura de manera programática estableciendo la ruta como pública (permitiendo peticiones GET sin necesidad de estar autenticado).

Las métricas que más se utilizarán son las relacionadas con el rendimiento de la aplicación y el uso de recursos del sistema, aunque se han añadido algunas personalizadas para determinar el uso que se hace de determinadas partes de la aplicación. La lista con las métricas más usadas es la siguiente:

- jvm_memory_used_bytes: bytes usados por la JVM.
- jvm_memory_committed_bytes: bytes que se han reservado para su uso por la JVM.
- system_cpu_usage: uso de CPU del sistema donde se ejecuta la aplicación.
- process_cpu_usage: uso de CPU del proceso de la JVM.
- system_cpu_count: número de procesadores disponibles para la JVM.
- logback_events_total: número de eventos de log de un determinado nivel (info, debug, warn, error).
- http_server_requests_seconds_count: número de peticiones HTTP realizadas a la aplicación.
- jvm_threads_live_threads: número total de hilos en ejecución.
- jvm_threads_daemon_threads: número de hilos en ejecución (en segundo plano).
- jvm_threads_peak_threads: número máximo de hilos activos desde que se inició la JVM.

3.4.2. GRAFANA (PANEL DE OBSERVABILIDAD)

Grafana es una solución de código abierto para mostrar las métricas que se recogen de las aplicaciones de una manera amigable en paneles personalizables [8]. Permite estudiar, analizar y monitorizar aplicaciones a lo largo del tiempo gracias a las marcas de tiempo proporcionadas junto con los datos que se recolectan. Se puede conectar con muchas fuentes de datos, como Graphite, Prometheus, ElasticSearch, MySQL, etc. Una ventaja de Grafana es que ofrece una solución 'on-premise', que permite desplegar una instancia propia en el mismo servidor donde se ejecuta la aplicación. De esta manera,

se garantiza la seguridad y protección de los datos, ya que no se exponen a Internet de manera directa.

Para la configuración de Grafana en el entorno de producción, se necesita un contenedor Docker que utilice la imagen `grafana/grafana-oss` y exponga un puerto para permitir conexiones (el 4046 en este caso). A continuación, se presenta una configuración básica para ejecutar una instancia de Grafana:

```yaml
version: '3.7'
services:
  grafana:
    image: grafana/grafana-oss:9.3.1
    ports:
      - "4046:3000"  # Mapeo de puertos (HOST:CONTENEDOR)
    volumes:
      - grafana-storage:/var/lib/grafana  # Se define el volumen 'grafana-storage'
    env_file:
      - GF_SECURITY_ADMIN_USER=<usuario>
      - GF_SECURITY_ADMIN_PASSWORD=<contraseña>
    networks:
      - rschat-net  # Se asigna la red interna para comunicarse con otros servicios
```

Código 3.1: Configuración mínima para ejecutar un contenedor con Grafana

En este proyecto se ha utilizado un panel importado desde las plantillas de la comunidad de Grafana para su utilización con Spring Boot y Prometheus [9] al que se le han añadido más paneles (para los logs y otras métricas). Algunos de los tipos de paneles que se pueden añadir son de tipo numérico o texto, gráficas, histogramas, mapas de calor, tablas, entre otros [10].

Figura 3.2: Visualización de métricas relacionadas con los recursos del sistema utilizados y tiempo de actividad

Figura 3.3: Visualización de métricas relacionadas con los logs y peticiones HTTP

Para que Grafana muestre toda esta información, se deben configurar 2 fuentes de datos de las cuales extraer las métricas. Para establecer estos ajustes, accedemos a `Configuration > Data Sources` y añadimos un 'data source' para Prometheus y otro para Loki, con la URL que provee las métricas (el nombre del contenedor con el puerto interno, ya que se usa la misma red para todos los servicios):

- Prometheus: `http://prometheus:9090`
- Loki: `http://loki:3100`

3.4.3. LOKI (AGREGACIÓN DE LOGS)

Loki es una herramienta de agregación de logs, que permite almacenar, indexar y consultar logs de manera eficiente, ya que se basa en el uso de etiquetas (para el agrupamiento y filtrado de logs) dejando el mensaje original sin modificar. Para el funcionamiento de Loki es necesario disponer de un agente que se encargue de enviar los logs a la instancia de Loki, y en este proyecto se ha utilizado Promtail, explicado en la siguiente sección. La configuración necesaria para que Grafana pueda recoger los logs de Loki es sencilla, ya que se añade un fichero `yaml` con la configuración de la fuente de datos de Loki [11] y el servicio en el fichero `docker-compose.yaml`:

```yaml
loki:
  image: grafana/loki:2.7.0
  container_name: loki
  ports:
    - "4045:3100"
  volumes:
    # Habilitar la lectura del fichero de configuración desde el contenedor
    - ./config/loki.yaml:/etc/loki/loki-config.yaml
  command: -config.file=/etc/loki/loki-config.yaml
  depends_on:
    - promtail
  networks:
    - rschat-net
```

Código 3.2: Servicio de Loki para el registro de logs (fichero `docker-compose.yaml`)

3.4.4. PROMTAIL (AGENTE DE LOGS)

Promtail es un agente (o cliente) de logs, que los recoge y los convierte en flujos que puede enviar a Loki (ver imagen 3.4) a través de una API HTTP [12]. La configuración de docker para Promtail es similar a la de Loki, ya que se añade un fichero `yaml` con la configuración de la fuente de datos de Promtail y el servicio en el fichero `docker-compose.yaml`:

```yaml
promtail:
  image: grafana/promtail:2.7.0
  container_name: promtail
  restart: unless-stopped
  ports:
    - "4044:9080"
  volumes:
    - /var/log:/var/log
    # Habilitar la lectura del fichero de configuración desde el contenedor
    - ./config/promtail.yaml:/etc/promtail/promtail.yaml
    # Habilitar acceso a los logs de los contenedores
    - /var/lib/docker/containers:/var/lib/docker/containers
  command: -config.file=/etc/promtail/promtail.yaml
  depends_on:
    - rschat
  networks:
    - rschat-net
```

Código 3.3: Servicio de Promtail para la recolección de logs (fichero `docker-compose.yaml`)

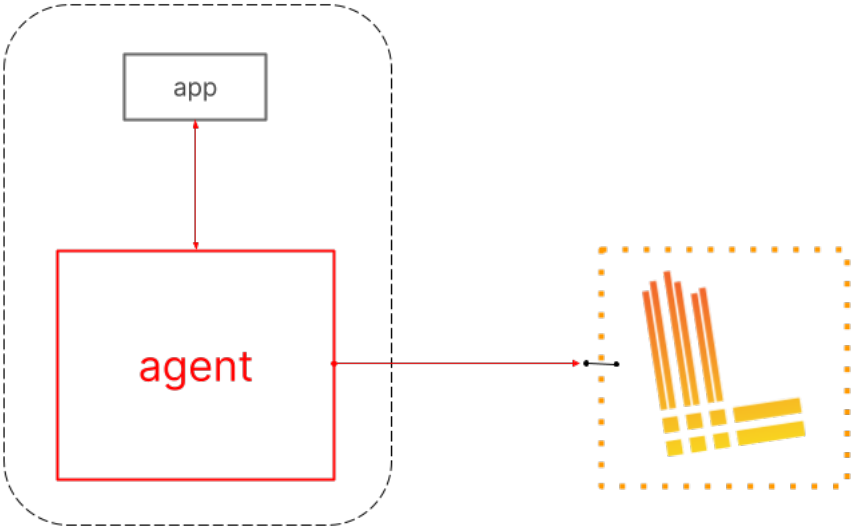

Figura 3.4: Flujo de logs entre Promtail y Loki

El formato de las trazas que exporta Promtail se ha ajustado para que se permita la visualización tanto por fichero, como por el nombre del contenedor que genera el log. Para ello, el fichero `promtail.yaml` contiene una sección `pipeline_stages` con la configuración de los 'stages' que se aplican a las trazas antes de enviarlas a Loki. La más importante es la que se encarga de añadir las etiquetas `container_name` con una expresión regular que extrae el nombre del contenedor a partir de una etiqueta asociada a cada traza:

```
pipeline_stages:
  # ...
  - regex:  # Nombre del contenedor en el valor de "tag"
      expression: (?P<container_name>(?:[^|]*[^|]))
      source: tag
  # ...
```

Código 3.4: Stage para obtener el nombre del contenedor con una expresión regular a partir de la etiqueta `tag` (fichero `promtail.yaml`)

3.5. BASE DE DATOS

El sistema gestor de base de datos utilizado para la aplicación es `MySQL`. Las bases de datos de MySQL son relacionales, lo que significa que los datos se almacenan en tablas, que están formadas por filas (campos) y columnas (registros). Las tablas se relacionan entre sí mediante claves primarias y claves foráneas, que son campos que identifican a

cada registro de la tabla. Este tipo de base de datos es muy empleado en aplicaciones web, por lo que es sencillo encontrar información sobre cómo usar MySQL. Como el lenguaje de programación con el que se ha desarrollado el backend de la aplicación es Java, se ha utilizado el ORM `Hibernate`, que permite realizar un mapeado de la base de datos a objetos de Java, de forma que se puede trabajar con ella de una forma transparente y cómoda para el programador. Para realizar las operaciones de inserción, actualización, consulta y borrado de los datos, `Spring Boot Data JPA` es la mejor opción, ya que se encarga de realizar estas operaciones de forma transparente para el desarrollador, dependiendo del nombre que reciba un método. También, mediante el uso de anotaciones (expresiones precedidas por el símbolo @, como por ejemplo `@Query`), se pueden personalizar de las consultas que se ejecutarán en base de datos. El diagrama de las tablas de la base de datos de la aplicación se muestra en la siguiente figura:

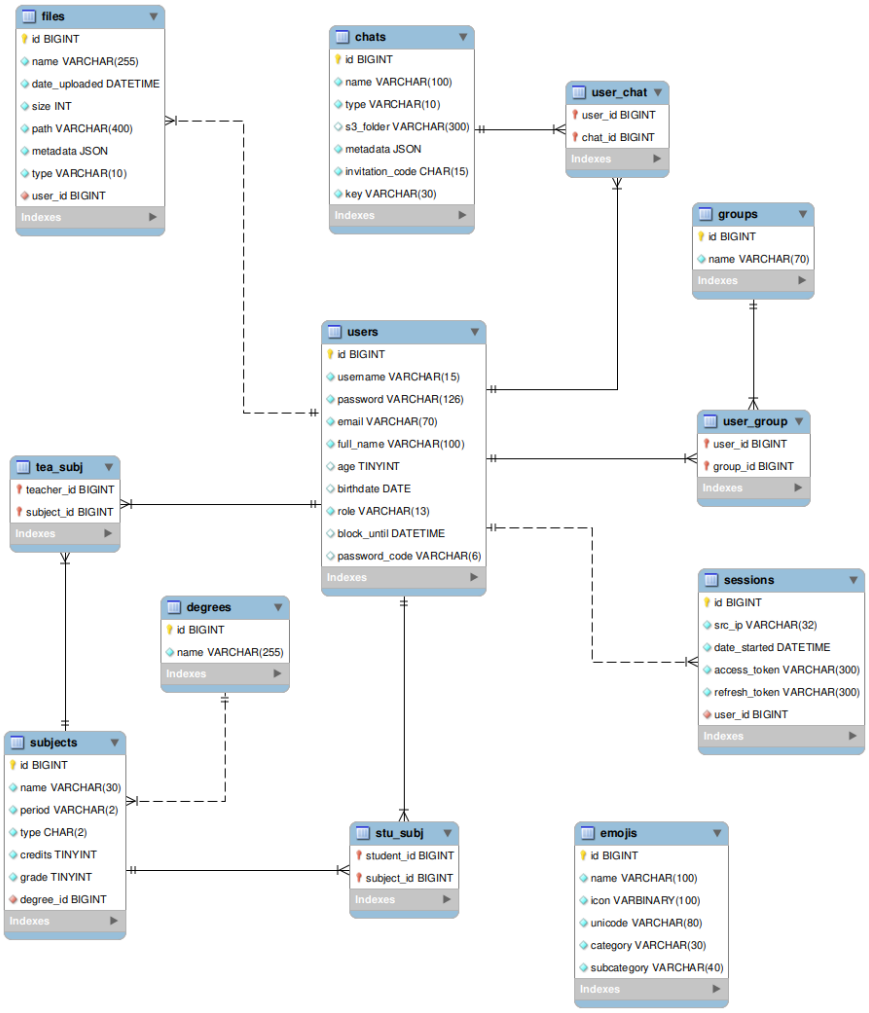

Figura 3.5: Diagrama de las tablas de la base de datos

3.6. SEGURIDAD

Para la prevención de ciberataques al servidor por parte de usuarios malintencionados, se han establecido las medidas de seguridad que se detallan a continuación:

- **Cifrado de la comunicación entre el servidor y los clientes**: Para evitar que los datos de los usuarios puedan ser interceptados por terceros, se ha establecido un cifrado de las comunicaciones. Para ello, se ha empleado el protocolo TLS, que permite cifrar la comunicación utilizando el certificado SSL que proporciona Let's Encrypt. Este certificado se ha generado automáticamente mediante su servicio gratuito de certificados y, utilizando la aplicación `Certbot`, se renueva automáticamente cada 90 días. El certificado se almacena en el servidor en el directorio `/etc/letsencrypt/live/`.

- **Configuración del servidor ssh**: Para evitar que los usuarios puedan acceder al servidor mediante ssh, se ha configurado para que únicamente se pueda acceder mediante claves ssh (no permitiendo usuario y contraseña). Estas claves se obtienen mediante el comando `ssh-keygen`, que genera un par de claves (pública y privada), y se ha copiado la clave pública del cliente en el fichero `/home/user/.ssh/authorized_keys` del servidor. De esta forma, únicamente se puede acceder al servidor mediante la clave privada del cliente.

- **Configuración del servidor web**: Se ha limitado la redirección de puertos desde el router hasta el servidor web, de forma que únicamente se puede acceder al servidor desde el puerto 22 (ssh), 4040 (aplicación en entorno de producción). Además, se ha configurado el servidor web para que únicamente se pueda acceder mediante el protocolo https, y no mediante el protocolo http, haciendo que los usuarios sean redirigidos automáticamente a https.

- **Configuración de la base de datos**: La base de datos solo es accesible desde el propio servidor web, por lo que el acceso está restringido a la red interna.

3.7. PRUEBAS

3.8. PROBLEMAS EN EL DESARROLLO

Uno de los problemas más significativos que han surgido en el desarrollo de la aplicación se ha dado en la parte del despliegue en producción de la misma. En un comienzo, la aplicación se había desplegado en dos clusters con el plan gratuito de Heroku (uno para el frontend y otro para el backend), pero en el momento en que esta plataforma publicó que se dejaría de dar soporte a las aplicaciones gratuitas, se empezaron a buscar alternativas para alojar la aplicación.

3.8.1. ALTERNATIVAS A HEROKU

Las opciones que se consideraron fueron las que se describen a continuación, ordenadas según la fecha de consideración.

- Para la parte de **frontend**, la elección fue muy sencilla. **Vercel** fue la primera opción a considerar, siendo una plataforma muy conocida por su facilidad para desplegar aplicaciones web. Cada vez que se realiza un cambio en el código fuente de la aplicación, se inicia un despliegue automático en la plataforma (esto ocurre si se tiene enlazado el repositorio de Git).

- Para la parte de **backend**, se consideraron las siguientes opciones:

  - **Alojamiento en la nube**: se consideró la posibilidad de alojar la aplicación en la nube utilizando otros servicios. Se llegó a concretar una reunión con un *Cloud Consultant* de **Platform.sh**, que fue la primera opción a considerar, pero no se llegó a ningún acuerdo para conseguir un plan gratuito para desplegar la aplicación, por tanto, se descartó esta opción.

  - **Raspberry Pi 4**: es un ordenador de tamaño muy reducido, que se puede configurar para que actúe como un servidor. Es una buena opción en caso de se desee un consumo de energía bajo. Sin embargo, no es una opción viable para el proyecto a largo plazo, ya que el almacenamiento se realiza en una tarjeta de memoria, cuya velocidad de lectura/escritura es más baja en comparación con los discos duros sólidos de los ordenadores convencionales.

- **Servidor propio**: es la opción más viable, ya que se puede aprovechar para realizar otras tareas como almacenamiento de gran cantidad datos o desplegar otras aplicaciones más potentes que con la Raspberry. Esta ha sido la **opción elegida** finalmente, ya que se ha encontrado un pequeño ordenador que cumple con los requisitos necesarios para alojar la aplicación.

3.9. PREPARACIÓN DEL SERVIDOR

Cuando se ha recibido el ordenador, se han seguido una serie de pasos para configurarlo de la manera más segura posible, ya que se va a utilizar para alojar el backend una aplicación web. Estos pasos se describen a continuación.

3.9.1. INSTALACIÓN DEL SISTEMA OPERATIVO

El sistema operativo que se ha instalado en el ordenador es **Ubuntu Server 20.04 LTS**. Se ha escogido esta distribución de Linux debido a que es una de las más populares y con una gran comunidad de usuarios, lo que hace que sea fácil encontrar información sobre cómo configurar el sistema. Además, se ha elegido la versión LTS para tener soporte durante un periodo de tiempo más largo (en el caso de esta versión, hasta 2025 y con actualizaciones de seguridad hasta 2030). Se han instalado todos los paquetes necesarios para administrarlo de una manera más cómoda y para garantizar la seguridad e integridad del sistema. Algunos de estos paquetes son los siguientes:

- **OpenSSH**: el protocolo **ssh** permite establecer conexiones seguras entre dos ordenadores. Se ha configurado esta herramienta para que se pueda acceder a él mediante un par de claves (pública/privada), eliminando la posibilidad de acceder mediante una contraseña. Esto último se realiza para evitar que se pueda establecer una conexión con él de forma no autorizada. De esta manera, los únicos usuarios que pueden acceder al servidor son aquellos que tienen la clave privada asociada a la clave pública registrada en el servidor. Para generar estas claves, se ha utilizado el comando **ssh-keygen** (en el cliente) y se ha guardado la clave pública en el archivo **authorized_keys** del servidor.

- **certbot**: es un servicio que permite obtener certificados SSL y configurarlos automáticamente en el servidor web.

3.9.2. CONFIGURACIÓN DE LA RED

El ordenador se ha conectado a la red mediante un cable Ethernet para una mejor comunicación entre sistemas. En la configuración del router, se ha asignado una dirección IP estática (dentro de la red local) al ordenador para que siempre tenga la misma dirección IP. Se ha contactado con el proveedor de Internet para consultar si sería posible obtener una dirección IP estática para el ordenador, pero no se ha podido conseguir, ya que utilizan la tecnología CG-NAT. Esto no es un problema ya que existe un servicio llamado **DuckDNS** que permite asociar una dirección IP de un ordenador a un dominio. Para configurar este servicio, se ha seguido la guía de instalación que se encuentra en la página web de **DuckDNS** [13]. Este servicio requiere de una tarea CRON para que se actualice la dirección IP cada 5 minutos. Esto se ha realizado con un pequeño script en Bash, que envía al servidor de **DuckDNS** la dirección IP pública del ordenador.

3.9.3. INSTALACIÓN Y EJECUCIÓN DE LA APLICACIÓN

La aplicación se ha instalado en el ordenador descargando el repositorio con el código fuente. Se ha programado un script en Bash que realiza todas las tareas necesarias antes de ejecutar la aplicación en los entornos de producción y desarrollo. Este script se ejecuta de manera manual cada vez que se quiere actualizar la aplicación, y realiza las siguientes tareas:

- Actualiza el código fuente de la aplicación.
- Instala las dependencias y compila el código fuente.
- Exporta las variables de entorno.
- Ejecuta la aplicación: para que se ejecute la nueva versión de la aplicación, se manda la señal `TERM` a todos los procesos que estén en escucha por los puertos que se necesitan para la aplicación (**4040** - Producción), configurados en ficheros `.env`. Esto provoca la terminación los procesos que estén escuchando por esos puertos.

3.9.4. CONFIGURACIÓN DE LA APLICACIÓN

Como se ha mencionado anteriormente, la aplicación cuenta con dos entornos: producción y desarrollo. Para configurar cada uno de ellos, se ha creado un fichero `.env` en la raíz del proyecto. Estos ficheros contienen las variables de entorno que se necesitan para ejecutar la aplicación. De esta manera, se asegura que la aplicación no exponga ninguna información sensible. Las variables más importantes que se han configurado en estos ficheros son las siguientes:

- Usuario y contraseña de la base de datos.
- Una cadena de texto que se utiliza para firmar los tokens de autenticación.
- Claves de acceso a los servicios de AWS.
- Puerto en el que se ejecuta la aplicación.
- Cuenta y contraseña (de aplicación) para enviar correos electrónicos.
- Nombre y contraseña de los ficheros que habilitan SSL en el servidor web.

3.9.5. CONFIGURACIÓN DE LA BASE DE DATOS

La base de datos se ha configurado por defecto con un usuario nuevo para el uso con la aplicación. Este usuario tiene permisos de lectura y escritura sobre la base de datos, por lo que no es necesario crear un usuario para cada aplicación.

3.9.6. SEGURIDAD DEL SERVIDOR

Con el objetivo de prevenir ataques a los dispositivos de la red local, se han configurado diferentes medidas de seguridad en el ordenador y el router. Estas medidas son las siguientes:

- **fail2ban**: es un servicio que permite bloquear las conexiones a un ordenador cuando se detecta un ataque de fuerza bruta a la IP de este servidor.
- Se han desactivado los puertos inactivos del ordenador, para evitar que se puedan utilizar para atacar a otros ordenadores.
- En el router, solo se ha habilitado la redirección de puertos a los que se necesitan para ejecutar la aplicación.

4. Anexos

4.1. ANEXO A: TAREAS DEL DESARROLLO

Durante todo el desarrollo del proyecto se han realizado una serie de tareas, para las cuales se ha utilizado la herramienta **Notion**. Esta herramienta proporciona una interfaz muy sencilla para la creación de listas de tareas, con la posibilidad de añadir etiquetas, asignar responsables, añadir comentarios, entre otras funcionalidades. A continuación se incluye el fichero pdf generado por Notion con las tareas realizadas durante el desarrollo del proyecto.

RS Chat

Aa Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Improve observability with Grafana	Idea	Configuration ⚙️ Improvement 🚀	Not Started			
Improve performance when showing the emojis	Medium	Improvement 🚀	Not Started			
Fix removal of chats with incorrect id. Modify tables to have one column with the id of the table the entries come from.	High	Bug 🐞	Complete	@October 22, 2022 12:15	@October 22, 2022 23:45	3
Add end date to sessions, to not verify on each token every 15 minutes to check if it has expired	Medium	Improvement 🚀 Refactor 🛠️	Not Started			
Send information message when the deployment is going to be done	High	Configuration ⚙️ Improvement 🚀	Not Started			
Add to the name of the subjects and groups a random String at the end to support repeated names, but not show it when retrieving information	Medium	Task 🔨	Not Started			
Improve Administration pages with edit and list views (with edit and remove button on every item of the list)	Medium	Improvement 🚀	Complete	@October 21, 2022 19:30	@October 21, 2022 17:23	10
Block users for 10 minutes when they introduce a wrong password	Medium	Improvement 🚀 Task 🔨	Not Started			
Block users	Medium	Improvement 🚀	Not Started			
Add friends	Medium	Improvement 🚀	Not Started			
Poll system integrated in the chat	Idea	Improvement 🚀	Not Started			
Improve log configuration file to make files with RollingFilePolicy	Medium	Improvement 🚀	Complete	@October 19, 2022 17:20	@October 20, 2022 18:10	0
Create email server at home	High	Configuration ⚙️ Improvement 🚀	Not Started			
Create self-signed SSL cert to enable access from Vercel	Very high	Bug 🐞 Configuration ⚙️	Complete	@October 16, 2022 18:32	@October 17, 2022 21:12	8

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Configure home server	Critical 🔥	Configuration ⚙️ Improvement 🚀	Complete	@October 14, 2022 16:00	@October 16, 2022 22:00	16
Make WebSockets work in the home server	Critical 🔥	Configuration ⚙️	Complete	@October 15, 2022 23:28	@October 15, 2022 23:32	0
Make dockerfiles to have a dev and prod containers of the backend	Very high	Configuration ⚙️ Improvement 🚀	Cancelled			
Backend works correctly in home server	Critical 🔥	Configuration ⚙️	Complete	@October 15, 2022 20:00	@October 15, 2022 23:15	0.7
Improve Active users tab with list switching (active / all)	Medium	Improvement 🚀	Complete	@October 13, 2022 15:50	@October 13, 2022 17:34	0.5
Improve chat grid at home page	Very high	Bug 🐞	Complete	@October 13, 2022 0:02	@October 13, 2022 0:22	0
Infinite requests when going to add new group	High	Bug 🐞	Complete	@October 12, 2022 23:24	@October 12, 2022 23:26	0
Register email address in Sendgrid to prevent automatic emails to be received in spam folder	Very high	Bug 🐞	In Progress	@October 12, 2022 2:35		
Published Instagram story to get feedback	Medium	Research ⛰️	Complete	@October 12, 2022 0:36	@October 13, 2022 0:36	0
Fix general crashes of the app	Critical 🔥	Bug 🐞	TempCancel			
Active users table have repeated usernames (users from mobile)	High	Bug 🐞	TempCancel			
Sound must be played on every message (fix hook to execute completely and not pause the sound)	Low	Bug 🐞	TempCancel			
When mobiles quit browser when they are inside the chat, the WS crashes	Very high	Bug 🐞	Not Started			
When a user wants to add another to a group, show all the available groups to that person to select one, to copy the invitation code with instructions or send an email	Idea	Improvement 🚀	Not Started			
Fit content to all devices (Phone, tablet, pc)	Critical 🔥	Improvement 🚀 Task 🔨	Complete	@October 11, 2022 23:07	@October 12, 2022 0:29	0
Implementación de algo de funcionalidad más especial. Papers	Idea		Not Started			

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Rank system, get a chat badge (or different colors) when user has written a lot of messages	Idea		Not Started			
Meeting with David Nkoto from Platform.sh	Critical 🔥	Task 🔨	Complete	@October 4, 2022 11:00	@October 4, 2022 11:20	0
Refactor message strategy to have only one implementation of handle method (Audio/Image/Text/Video classes)	Medium	Refactor 🛠️	Complete	@September 30, 2022 19:48	@September 30, 2022 20:00	0
FIX: max width in image dialog is not correct. See the image sent by amiguelediez	Medium	Bug 🐞	Not Started			
Improve services to throw exceptions if the entity is not found or an error occurs (and not make it in the controller) and other functionality that could be moved from controllers	Medium	Refactor 🛠️ Test 🧪	May change	@September 28, 2022 16:20	@October 7, 2022 0:19	-1
Notification when a message is received	Low	Improvement 🚀	Complete	@October 8, 2022 1:36	@October 8, 2022 2:00	0
Add chat tabs to have several opened at the same time	Low	Improvement 🚀	Not Started			
Add response messages by pressing a button or making double click	Medium	Improvement 🚀	Not Started			
FIX: when a user receives a message and the image is opened, the dialog is closed for some reason	Medium	Bug 🐞	Not Started			
Add some unit and integration tests	High	Task 🔨 Test 🧪	In Progress	@September 27, 2022 11:30		
Fix problems with tests	Very high	Bug 🐞 Test 🧪	Complete	@September 26, 2022 21:45	@September 26, 2022 22:36	0.2
Check the indexes in the LaTeX document to correctly generate the numbers	High	Documentation 📘	Cancelled			
Apply observer pattern when users join a chat	Medium	Improvement 🚀	Not Started			
Document the pattern used by the ChatMap class	High	Documentation 📘	Complete	@October 1, 2022 13:45	@October 4, 2022 12:16	-1
Instead of ':' for showing emojis, add a button at the beginning of the TextBox	High	Refactor 🛠️	Complete	@September 22, 2022 15:00	@October 6, 2022 20:17	-1

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Add support for commands with '/' character	Low	Improvement 🚀	Not Started			
Add the clock (bean) to Instant.now() calls	Medium	Task 🔨	Complete	@September 16, 2022 13:26	@September 16, 2022 13:36	0
Get all the users that belong to a chat and put a small green icon to those connected (show first), those unconnected with a red icon and last	Idea	Improvement 🚀 Task 🔨	Not Started			
Reduce emojis file to only needed fields (name, emoji, unicode, category, sub-category)	Medium	Improvement 🚀	Complete	@September 15, 2022 12:36	@September 15, 2022 13:46	0
Refactor the messages sent over WS to remove encoding field (we only use UTF-8), and simplify to UTF8	High	Refactor 🛠️	Complete	@September 15, 2022 12:21	@September 15, 2022 12:33	0
Add support for adding emojis without pasting them (some keyword like ':')	Low	Improvement 🚀	Complete	@September 16, 2022 14:03	@September 19, 2022 21:04	-1
Fix problems with audio files that move one message down	Critical 🔥	Bug 🐞	Complete	@September 14, 2022 16:47	@September 14, 2022 17:24	0
Save the files uploaded to the database	High	Task 🔨	Complete	@September 14, 2022 13:50	@September 14, 2022 14:52	0
Limit the size of the files uploaded	High	Improvement 🚀	Complete	@September 14, 2022 14:52	@September 14, 2022 14:57	0
Improve administration view to have several big buttons to select the next window to visualize (not having all data in the same page)	High	Improvement 🚀 Refactor 🛠️	Complete	@September 13, 2022 12:15	@September 13, 2022 13:42	0
Solve problem of not sending USER_LEFT (and USER_JOINED) messages to the server	Critical 🔥	Bug 🐞	Complete	@September 12, 2022 17:15	@September 12, 2022 20:00	0.1
Fix exception when a user exits a chat. User is not created and added to chat	Critical 🔥	Bug 🐞	Complete	@September 12, 2022 8:35	@September 12, 2022 8:54	0
Remove checks on register. Only show error when server sends the response	High	Refactor 🛠️ Task 🔨	Cancelled			

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Change all TextField components to be size='small'	Low	Refactor 🛠️	Complete	@September 10, 2022 20:49	@September 11, 2022 20:55	0
Fix 404 error when accessing app in Vercel from external page	High	Bug 🐞 Documentation 📘	Complete	@September 9, 2022 14:00	@September 9, 2022 14:30	0
GET request to a URL with an invitation code as a parameter. Then, the user can join the chat if user is not joined and redirected to it in both cases (joined or not)	Idea	Task 🔨	TempCancel			
Add email templates for register and password change	High	Improvement 🚀	Complete	@September 9, 2022 1:20	@September 11, 2022 17:25	32
Remove null clients from list in backend when a message is going to be sent	Medium	Bug 🐞	Complete	@September 14, 2022 20:04	@September 14, 2022 20:16	0
Extract to a JSON the router structure in App component	High	Refactor 🛠️	Complete	@September 8, 2022 16:11	@September 8, 2022 16:54	0
Add style to have chats structured in a grid in Home view	High	Improvement 🚀 Task 🔨	Complete	@September 7, 2022 21:12	@September 8, 2022 2:10	1.5
Administrators and teachers must have access to invitation code of a chat	High	Improvement 🚀	Complete	@September 12, 2022 23:00	@September 13, 2022 1:00	0
Remove express and file to serve frontend (since Vercel does not need it)	Very high	Refactor 🛠️	Complete	@September 8, 2022 17:06	@September 8, 2022 17:10	0
Configure email sender to be able to send emails when users are registered or password are reset	Very high	Improvement 🚀 Task 🔨	Complete	@September 9, 2022 0:05	@September 9, 2022 1:11	0.5
Make a Feature readme file	Very high	Documentation 📘	Complete	@September 6, 2022 20:00	@September 7, 2022 0:13	2
Implement Forgot password functionality and improve reset password. When the user wants to change the password in the profile, send a forgotPassword request with the email, then redirect to create password, since it will be correct	High	Improvement 🚀 Task 🔨	Complete	@September 9, 2022 20:35	@September 10, 2022 3:06	2.2

Aa Projects	Priority	Type	Status	Started	Finished	# Inac (h)
If user checks the “Remember me” checkbox, extend the expiration time of the access token to 2 weeks or more	High	Improvement 🚀	Complete	@September 9, 2022 20:05	@September 9, 2022 20:27	0
Be able to handle several files at once, to keep the order in which they were sent	Medium	Improvement 🚀	Complete	@September 6, 2022 1:45	@September 6, 2022 3:00	0
Disable send button if no text or image is going to be sent	Low	Improvement 🚀 Task 🔨	Complete	@September 6, 2022 1:00	@September 6, 2022 1:50	0
Show how many images are attached to the message	Low	Improvement 🚀 Task 🔨	Complete	@September 6, 2022 1:20	@September 6, 2022 1:40	0
Migrate applications from Heroku to Vercel	High	Task 🔨	Complete	@September 5, 2022 23:00	@September 5, 2022 23:15	0
Add support for having database in localhost	High	Task 🔨	Complete	@September 2, 2022 18:45	@September 2, 2022 18:56	0
Send a ping message from frontend to keep the connection alive	Critical 🔥	Task 🔨	Complete	@September 4, 2022 23:00	@September 4, 2022 23:27	0
Idle connection of WebSockets	Critical 🔥	Bug 🐞	Complete	@September 7, 2022 14:22	@September 7, 2022 17:00	1
Fix url when receiving response from server	Critical 🔥	Bug 🐞	Complete	@August 28, 2022 23:19	@August 29, 2022 0:40	0
Add functionality for adding more media files	Very high	Task 🔨	Complete	@August 28, 2022 23:15	@September 2, 2022 17:34	-1
Migrate maxWidth determination to backend	Low	Refactor 🛠️	Complete	@August 28, 2022 20:43	@August 28, 2022 20:55	0
Group multiple images/videos in one message	Low	Improvement 🚀 Task 🔨	Not Started			
Zoom on scroll (in images)	Medium	Improvement 🚀 Research ⛰️	Not Started			
Send the media files with http requests to the server instead of encoding and sending through the WebSocket	Very high	Improvement 🚀 Refactor 🛠️	Complete	@August 27, 2022 14:30	@August 27, 2022 23:45	1.5
Add a small dropdown to select the type of file the user is going to send	High	Improvement 🚀	Not Started			

Aa Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Add support to encode (in base64) files to be able to send to the server as normal WebSocket messages	High	Task 🔨	Complete	@August 26, 2022 22:50	@August 26, 2022 23:20	0
Add drag ‘n’ drop to select files to send to the chat	High	Task 🔨	Complete	@August 26, 2022 20:50	@August 27, 2022 15:00	12
Add the README.md	Low	Documentation 📘	Complete	@August 26, 2022 19:05	@August 26, 2022 21:00	0.2
Chat statistics - Per user, number of messages, words written…	Idea	Improvement 🚀	Not Started			
Replace disk reads for memory reads when reading chat history	Idea	Improvement 🚀	Not Started			
Schedule the upload of the history file to S3 after a period of time	High	Improvement 🚀	Reference	@August 24, 2022 20:30	@August 24, 2022 20:50	0
Read history from file stored in disk if it is downloaded. To have updated history to new users	Critical 🔥	Bug 🐞 Refactor 🛠️	Complete	@August 23, 2022 18:00	@August 24, 2022 1:00	2
Implement pagination of the chat history	Medium	Improvement 🚀 Task 🔨	Not Started			
Do the documentation of the Java code	Very high	Documentation 📘	Complete	@August 24, 2022 22:47	@August 26, 2022 0:40	12
Solve problems of websocket connections in remote	Critical 🔥	Bug 🐞	Cancelled			
Discover the limit to send data through WS to frontend	High	Refactor 🛠️	Cancelled		@September 5, 2022 0:29	
Check this lib to support message compression in frontend. Must be implemented in backend since it is written in Java	Idea	Improvement 🚀 Refactor 🛠️	TempCancel			
See this to send partial data (maybe)	Idea	Improvement 🚀 Refactor 🛠️	TempCancel			
Add a reducer for chat and store the current chat	Medium	Improvement 🚀	TempCancel			
Order active users by name	Low	Improvement 🚀	Complete	@August 23, 2022 20:25	@August 23, 2022 20:34	0
Use strategy pattern to make MessageHandler more readable	Very high	Refactor 🛠️	Complete	@August 22, 2022 0:16	@August 22, 2022 20:47	15

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Upon connection of the user, receive all the messages of the chat up to a limit of 3 days before	High	Improvement 🚀	Complete	@August 22, 2022 20:49	@August 22, 2022 23:48	0.75
Show popup when hovering hour of message to display full date	High	Improvement 🚀	Complete	@August 21, 2022 19:26	@August 21, 2022 20:00	0
When clicking a username in a chat, display a small popup with a button to start a user-user chat	High	Improvement 🚀 Task 🔨	Cancelled	@August 20, 2022 0:20	@August 21, 2022 23:23	1.2
Edit diagrams and refactor entity classes (relation classes) and SQL definition to make columns match the table name	Very high	Bug 🐞 Refactor 🛠️	Complete	@August 19, 2022 16:41	@August 19, 2022 17:18	0
Create invitation links to put in profile and join a chat	Very high	Improvement 🚀 Task 🔨	Complete	@September 7, 2022 17:31	@September 7, 2022 20:56	1.5
Listen to new messages and check if they are USER_JOINED or USER_LEFT and send a new type of message to get the connected users of the chat in the background	Medium	Improvement 🚀	Complete	@August 19, 2022 22:00	@August 20, 2022 0:17	0
Add a list of connected users to the chat view	Idea	Improvement 🚀	Complete	@August 19, 2022 22:37	@August 20, 2022 0:13	0
Improve chat with name and more information about it	High	Task 🔨	May change	@August 18, 2022 19:15	@August 18, 2022 20:19	0.25
Fix problems adding users to global chat on register	Very high	Bug 🐞	Complete	@August 17, 2022 20:32	@August 17, 2022 23:20	0.75
Add a global variable in redux state and a custom hook to get the websocket client only when the variable is true (userLoggedIn), if false, create client and then return. This will work like the color theme.	Medium	Improvement 🚀 Refactor 🛠️	Not Started			
Change type of chat in DB to longer name	High	Refactor 🛠️	Complete	@August 16, 2022 20:56	@August 17, 2022 22:30	1
Make chatReducer to shorten loading of home screen when requesting all available chats for a user	High	Improvement 🚀 Task 🔨	Complete	@August 16, 2022 16:55	@August 16, 2022 17:36	0

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Remove sessionId of the user both in frontend and backend	Medium	Refactor 🛠️	Revision			
Add a callback to client to execute it when a message is sent correctly	Low	Refactor 🛠️	Complete	@August 16, 2022 17:41	@August 16, 2022 17:58	0
Show available chats to user in home view	High	Task 🔨	Complete	@August 15, 2022 19:33	@August 15, 2022 23:05	1.4
Try the reconnection of the WebSocket in frontend	Medium	Improvement 🚀	Complete			
How to escape SQL reserved keywords with Spring boot, JPA and Hibernate - Java JEE tutorials by Tarik Chrouki	High	Bug 🐞	Complete	@August 14, 2022 17:40	@August 14, 2022 17:45	0
Divide each DropDown into new component and include them in Administration view	Medium	Refactor 🛠️	Complete	@August 18, 2022 20:22	@August 19, 2022 0:15	1.5
Add groups to all diagrams	Critical 🔥	Task 🔨	Complete	@August 13, 2022 0:45	@August 13, 2022 14:05	12
Decide structure of the chat URLs	Very high	Task 🔨	May change	@August 13, 2022 0:30	@August 13, 2022 1:00	0
When entering a chat, send a http request to check if the user has access to that chat	Very high	Task 🔨	Complete	@September 8, 2022 17:23	@September 8, 2022 19:59	0.5
Decide chat type names	High	Refactor 🛠️	Revision	@August 13, 2022 0:11	@August 13, 2022 0:30	0
If several messages are received from the same sender, do not render the username	Idea	Task 🔨	Not Started			
Add support to compute the color hash that a user has.	Idea	Improvement 🚀	Not Started			
Domain diagram	High	Documentation 📘	Revision	@August 9, 2022 22:10	@August 9, 2022 22:30	0
Use case diagram	High	Documentation 📘	Revision	@August 9, 2022 23:15	@August 11, 2022 22:58	18
Classes diagram	High	Documentation 📘	Not Started			
Interaction diagram	High	Documentation 📘	Not Started			
Packages diagram	High	Documentation 📘	Not Started			
Send a confirmation email to new registered users	Medium	Security 🔒 Task 🔨	Not Started			
Spinner on login or register, to know that the request has been sent	Medium	Improvement 🚀	Complete	@August 11, 2022 23:00	@August 11, 2022 23:40	0

Aa Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Check that all routes have the correct http method and permissions	High	Task 🔨	Complete	@August 7, 2022 16:53	@August 7, 2022 17:32	0
Update old methods that used ResponseEntity<?> to use the new HttpResponse class	Low	Improvement 🚀 Refactor 🛠️	Complete	@August 6, 2022 23:29	@August 6, 2022 23:42	0
Implement Degree controller and associated Service in backend	High	Task 🔨	Revision	@August 6, 2022 16:45	@August 6, 2022 20:48	0.3
Add the service files to the frontend	Medium	Task 🔨	Ignore			
Make separate classes for Route constants (GET, POST, ...)	Low	Refactor 🛠️	Complete	@August 6, 2022 14:26	@August 6, 2022 16:14	1.5
Make administrator view in frontend	Medium	Improvement 🚀 Task 🔨	Complete	@August 6, 2022 12:30	@September 13, 2022 13:42	-1
Create updated entities in JPA	High	Task 🔨	Complete	@August 5, 2022 22:15	@August 6, 2022 23:26	12
Check migration guide java-jwt to version 4	Low	Security 🔒	Complete	@August 12, 2022 20:43	@August 12, 2022 23:56	2.5
Move ChatMap functionality to Chat class	Low	Refactor 🛠️ Task 🔨	Not Started			
Restrict the connection to chats. Users can only connect to available ones	Critical 🔥	Refactor 🛠️ Task 🔨	Complete	@September 8, 2022 17:23	@September 8, 2022 19:59	0.5
New messages when user joins and leaves a chat	High	Refactor 🛠️ Task 🔨	Cancelled	@August 3, 2022 20:15	@August 4, 2022 22:30	
Use a context in private routes to encapsulate a single WebSocket for all the session	High	Improvement 🚀 Task 🔨	Cancelled		@August 4, 2022 22:30	
Upload images from frontend with drag 'n' drop and encode into base64	High	Task 🔨	Reference			
Add random generated string at the end of media files	Medium	Improvement 🚀 Task 🔨	Not Started			
Add support for all types of files (text, images, videos, audios)	Medium	Task 🔨	May change	@August 1, 2022 23:18	@August 3, 2022 0:05	18
Get all files that a user has sent over the time	Idea		Not Started			
Achievements' system	Idea		Not Started			

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
When uploading file (finished chat) delete from disk. If chat is reopened, download file and keep writting	Medium	Improvement 🚀 Task 🔨	Complete	@July 31, 2022 20:06	@August 1, 2022 23:05	20
Update docs	Very high	Documentation 📘	May change	@August 5, 2022 19:50	@August 7, 2022 17:52	
Add a Writer to support message storage in S3. Upload files	High	Improvement 🚀 Task 🔨	Complete	@July 31, 2022 19:55	@July 31, 2022 20:05	0
Check token on connect to websocket	Medium	Improvement 🚀 Security 🔒 Task 🔨	Revision	@August 5, 2022 20:30	@August 15, 2022 17:04	-1
Better handling of chats with a new class	Low	Refactor 🛠️	Complete	@July 31, 2022 15:45	@July 31, 2022 17:22	0
Migrate backend to spring websocket support with custom Handlers	High	Refactor 🛠️ Task 🔨	Complete	@July 30, 2022 18:42	@July 31, 2022 15:00	10
Migrate from Java-WebSocket lib to Jetty websockets in backend	High	Refactor 🛠️ Task 🔨	Complete	@July 29, 2022 21:03	@July 30, 2022 15:59	14
Migrate from native WebSocket to websocket library in frontend	High	Refactor 🛠️ Task 🔨	Cancelled	@July 30, 2022 16:05	@July 30, 2022 17:31	
Migrate from native WebSocket to SockJS and	Medium	Refactor 🛠️	Cancelled		@July 30, 2022 23:00	
Provide more style to message components	High	Improvement 🚀 Task 🔨	Complete	@July 28, 2022 17:05	@July 28, 2022 20:33	0.8
If up arrow key is pressed, visit the message history	Idea	Improvement 🚀	Not Started			
Show incoming messages in ChatBox	High	Task 🔨	Complete	@July 27, 2022 20:30	@July 28, 2022 16:55	14
Make a record class to identify the WS user in backend	High	Refactor 🛠️ Task 🔨	Complete	@July 27, 2022 17:57	@July 27, 2022 18:42	0
Receive message in component	High	Refactor 🛠️ Task 🔨	Complete	@July 27, 2022 0:48	@July 27, 2022 2:07	0
Improve storage of clients in backend	High	Task 🔨	Complete	@July 26, 2022 22:52	@July 27, 2022 0:32	0.6
Improve message handling between front and back	High	Refactor 🛠️ Task 🔨	Complete	@July 26, 2022 18:20	@July 26, 2022 21:28	0.6
Clear the sensitive headers to send the received message to other clients in backend	Critical 🔥	Refactor 🛠️ Task 🔨	Complete	@July 26, 2022 18:22	@July 26, 2022 18:38	0

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Limit rate at which users can send messages to 6 per second	Idea	Improvement 🚀 Task 🔨	Not Started			
Return an existing session to the user that logs in with other host	Idea	Improvement 🚀 Task 🔨	Not Started			
Add the sessionId when sending the response to the client	High	Refactor 🛠️ Task 🔨	Complete	@July 26, 2022 16:37	@July 26, 2022 17:30	0.25
Make a deploy test in Heroku, to verify the functionality of the sockets	High	Research ⛰️	Complete	@July 28, 2022 17:42	@August 5, 2022 0:45	-1
Receive messages in frontend from other clients and log to console	High	Task 🔨	Complete	@July 26, 2022 0:30	@July 26, 2022 1:28	0
Configure frontend to send messages. Create class and Hook	High	Task 🔨	Complete	@July 25, 2022 19:34	@July 26, 2022 0:34	1.7
Logout check	Low		Not Started			
Messages could be fragmented. Add an attribute in the JSON	Idea		Cancelled			
File writing from several threads (maybe use syncronized)	Low		Cancelled			
Create WebSocket server	High	Task 🔨	Complete	@July 23, 2022 20:15	@July 24, 2022 1:12	1.5
When sending an empty message, write nothing in the response	Critical 🔥	Refactor 🛠️ Task 🔨	Complete	@July 23, 2022 17:40	@July 23, 2022 17:47	0
Send messages to other members of the chat	Critical 🔥	Task 🔨	Complete	@July 24, 2022 14:48	@July 26, 2022 1:28	14
Update security of the routes. Make a separate class or something	Low	Refactor 🛠️ Task 🔨	Complete	@July 21, 2022 20:00	@July 21, 2022 20:57	0
Move header of message up one component (to ChatMessage.jsx)	Medium	Task 🔨	Complete	@July 21, 2022 19:27	@July 21, 2022 19:47	0
Check the date in which the message was sent, to write correctly to the file	Critical 🔥	Bug 🐞 Task 🔨	Complete	@July 27, 2022 0:26	@July 27, 2022 0:29	0
Add a new table in DB (Chats)	Critical 🔥	Task 🔨	Complete	@July 21, 2022 17:20	@July 21, 2022 19:06	0.5
When hovering the date of the message, show full date (with millisecond precission)	Low	Task 🔨	Reference			
Show a division of messages per day	Low	Task 🔨	Not Started			
Font size selector	Low	Improvement 🚀 Task 🔨	Not Started			

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Upload the file to S3 every 10 minutes	Medium	Improvement 🚀 Task 🔨	Revision	@August 24, 2022 20:30	@August 24, 2022 20:50	0
Store messages in a file with history	High	Task 🔨	Complete	@July 21, 2022 22:00	@July 21, 2022 23:30	0.4
Send messages from front to back	High	Task 🔨	Complete	@July 21, 2022 19:55	@July 21, 2022 21:18	1
Read messages in backend	High	Task 🔨	Complete	@July 21, 2022 21:20	@July 21, 2022 21:30	0
More style to chat and messages	Medium	Task 🔨	Complete	@July 20, 2022 17:55	@July 20, 2022 23:55	2
Basic chat page	High	Task 🔨	Complete	@July 19, 2022 17:00	@July 19, 2022 23:50	1.5
Add custom Drop Down, password reset and more info in profile	High	Task 🔨	Complete	@July 16, 2022 22:04	@July 17, 2022 0:05	0
Enter the app as anonymous user and after the session is closed, all the data, messages, etc. is deleted	Idea		Cancelled			
Make a Router class to configure HTTP methods and permissions	Low	Task 🔨	Cancelled	@June 27, 2022 11:26	@June 27, 2022 11:29	0
Change color themes to fit logo	Low	Improvement 🚀	Not Started			
Show all opened sessions of user in profile view	Medium	Task 🔨	Complete	@June 26, 2022 0:50	@June 26, 2022 1:15	0
Profile page with user data (session, name, ...)	High	Task 🔨	Complete	@June 26, 2022 23:30	@July 17, 2022 0:04	-1
Home page with chat selection	Medium	Task 🔨	Reference			
Improve request handling in filters. Put data in some attribute in the custom class instead of request.setAttribute()	High	Refactor 🛠️ Task 🔨	Complete	@June 25, 2022 15:00	@June 25, 2022 15:20	0
Frontend API url change when code is in production or in development env	High	Bug 🐞 Refactor 🛠️	Complete	@June 25, 2022 0:30	@June 25, 2022 0:50	0
App renaming	High	Refactor 🛠️	Complete	@June 24, 2022 23:54	@June 25, 2022 0:15	0
Solve problem with Instant class of the domain models	High	Bug 🐞 Refactor 🛠️	Complete	@June 24, 2022 22:45	@June 24, 2022 23:15	0.25
Refactor Responses with the new class	Medium	Refactor 🛠️ Task 🔨	Complete	@June 24, 2022 22:00	@June 24, 2022 22:33	0
Improve custom HttpRequest functionality	High	Task 🔨	Complete	@June 24, 2022 0:05	@June 24, 2022 1:04	0

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
Improve custom HttpResponse functionality	High	Task 🔨	Complete	@June 23, 2022 22:40	@June 24, 2022 3:08	0
Migration of frontend to other repo	Critical 🔥	Bug 🐞 Refactor 🛠️ Task 🔨	Complete	@June 23, 2022 0:37	@June 23, 2022 1:35	0
Better configuration of deployment CORS config	Critical 🔥	Task 🔨	Complete	@June 22, 2022 20:30	@June 23, 2022 1:35	1.5
Configure deploy functionality	Critical 🔥	Initialization 🎉 Task 🔨	Complete	@June 21, 2022 13:00	@June 22, 2022 1:46	3
Backend register functionality	Critical 🔥	Task 🔨	Complete	@June 20, 2022 21:15	@June 24, 2022 22:05	26
Toolbar with theme switcher	Critical 🔥	Task 🔨	Complete	@June 20, 2022 20:55	@June 20, 2022 21:03	0
Store, Reducers and Actions for Redux	High	Task 🔨	Complete	@June 20, 2022 20:41	@June 20, 2022 20:50	0
Theme switcher and color palette	Low	Task 🔨	Complete	@June 20, 2022 20:36	@June 20, 2022 21:03	0
Register page and switch between Login and Register	High	Task 🔨	Complete	@June 20, 2022 20:26	@June 20, 2022 20:45	0
Forgot password functionality	Medium	Task 🔨	Reference			
Documentation - All diagrams	High	Documentation 📘 Initialization 🎉	Not Started			
Migration from Timestamp to LocalDateTime	Medium	Refactor 🛠️	Cancelled	@June 17, 2022 15:45	@June 17, 2022 15:50	0
Remove Service's interfaces	High	Refactor 🛠️	Complete	@June 17, 2022 14:00	@June 17, 2022 14:17	0
Fix CORS security	Critical 🔥	Bug 🐞 Task 🔨	Complete	@June 15, 2022 17:50	@June 15, 2022 18:56	0
When receiving a request, check if the token exists in DB	High	Task 🔨	Not Started	@June 16, 2022 18:00		
Fix login controller does not work after AuthenticationFilter	Critical 🔥	Bug 🐞 Configuration ⚙️	Complete	@June 15, 2022 13:00	@June 15, 2022 14:10	0
Improve Logout functionality with a Filter an then call the controller	High	Task 🔨	Cancelled	@June 15, 2022 2:00	@June 15, 2022 18:40	7
Solve problem with user creation by admins	Critical 🔥	Bug 🐞 Task 🔨	Complete	@June 15, 2022 0:29	@June 15, 2022 1:08	0
Ability to close all sessions of the user	Medium	Task 🔨	Not Started			
Update DB to include refreshToken	High	Configuration ⚙️ Task 🔨	Complete	@June 14, 2022 21:43	@June 14, 2022 21:48	0

Projects	Priority	Type	Status	Started	Finished	# Inac (h)
SessionService update when refreshToken is used	Medium	Task 🔨	Not Started			
Implement SessionService with capability to save, delete	High	Task 🔨	Complete	@June 14, 2022 21:37	@June 14, 2022 22:00	0
Improve Filters with annotations	Medium	Refactor 🛠️	Cancelled	@June 15, 2022 0:18	@June 15, 2022 0:27	0
Backend routes with constants	Medium	Configuration ⚙️ Task 🔨	Complete	@June 14, 2022 13:16	@June 14, 2022 14:12	0
Be able to manage request in controller after passing a Filter	Critical 🔥	Configuration ⚙️	Complete	@June 14, 2022 17:30	@June 14, 2022 20:40	1
Document - Design model	High	Documentation 📘 Initialization 🎉	May change	@August 9, 2022 19:50	@August 9, 2022 20:15	0
Document - Requirements specification	High	Documentation 📘 Initialization 🎉	Complete	@June 20, 2022 14:00	@June 20, 2022 20:23	2
Document - Use cases	High	Documentation 📘 Initialization 🎉	Not Started			
Backend login with JWT tokens	High	Task 🔨	Complete	@June 13, 2022 19:00	@June 14, 2022 2:30	2
From Cayenne to JPA	High	Configuration ⚙️ Task 🔨	Complete	@June 13, 2022 13:53	@June 13, 2022 18:00	3
Configure Spring security with basic requests	Critical 🔥	Configuration ⚙️ Task 🔨	Complete	@June 13, 2022 23:15	@June 15, 2022 1:55	9
Translate Relational diagram to SQL	High	Initialization 🎉	Complete	@June 12, 2022 23:40	@June 13, 2022 0:20	0
Probar el módulo de traducción en Frontend	Idea		Not Started			
Create Relational diagram	High	Documentation 📘 Initialization 🎉	Complete	@June 11, 2022 22:18	@June 12, 2022 2:30	3
Create the model classes with Cayenne Modeler	Medium	Initialization 🎉	Cancelled	@June 13, 2022 0:20	@June 13, 2022 0:45	0
Create ER diagram	High	Documentation 📘 Initialization 🎉	Complete	@June 11, 2022 13:15	@June 11, 2022 22:15	3
Sistema de menciones	Idea	Research ⛰️	Not Started			
How To Build React with Java Backend For Production | by Bhargav Bachina | Bachina Labs | Medium	Idea	Research ⛰️	Complete			
How To Develop and Build React App With Java Backend | by Bhargav Bachina | Bachina Labs | Medium	Idea	Research ⛰️	Complete			

Bibliografía

[1] Refactoring Guru. *Classification of patterns*. URL: https://refactoring.guru/design-patterns/classification.

[2] V. Sarcar. *Java Design Patterns: A Hands-On Experience with Real-World Examples*. Apress, 2018. Cap. 1. ISBN: 9781484240786. URL: https://books.google.es/books?id=vPt9DwAAQBAJ.

[3] I. Fette y A. Melnikov. «The WebSocket Protocol». En: (dic. de 2011). ISSN: 2070-1721. DOI: 10.17487/RFC6455. URL: https://www.rfc-editor.org/info/rfc6455.

[4] *Docker overview | Docker Documentation*. (Accedido el 21/12/2022). URL: https://docs.docker.com/get-started/overview/.

[5] *Volumes | Docker Documentation*. (Accedido el 21/12/2022). URL: https://docs.docker.com/storage/volumes/.

[6] *Overview | Prometheus*. (Accedido el 21/12/2022). URL: https://prometheus.io/docs/introduction/overview/.

[7] *Spring Boot app metrics - with Prometheus and Micrometer - Tutorial Works*. (Accedido el 22/12/2022). URL: https://www.tutorialworks.com/spring-boot-prometheus-micrometer/.

[8] *What Is Grafana? Why Use It? Everything You Should Know About It - Scaleyourapp*. (Accedido el 22/12/2022). URL: https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/.

[9] *Spring Boot 2.1 System Monitor | Grafana Labs*. URL: https://grafana.com/grafana/dashboards/11378-justai-system-monitor/.

[10] *Visualizations | Grafana documentation*. URL: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/.

[11] *Promtail-Loki-Grafana-using-Docker-Compose/loki-config.yaml*. URL: https://github.com/shazforiot/Promtail-Loki-Grafana-using-Docker-Compose/blob/main/loki-config.yaml.

[12] *Overview | Grafana Loki documentation*. URL: https://grafana.com/docs/loki/latest/fundamentals/overview/.

[13] *Duck DNS - install*. (Accedido el 17/10/2022). URL: https://www.duckdns.org/install.jsp.