



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: Julio, 2023

RESUMEN:

RS Chat es una aplicación web para la interacción entre estudiantes y docentes en tiempo real. Su objetivo es facilitar la comunicación y colaboración entre los usuarios. Ofrece un sistema de almacenamiento del historial de mensajes, notificaciones y compartición de contenido multimedia, filtrando las imágenes inapropiadas debido al entorno educativo al que se ha dirigido.

ABSTRACT:

RS Chat is a web application for real time interaction between students and teachers. Its objective is to facilitate communication and collaboration between users. It offers a message history storage system, notifications and sharing of multimedia content, filtering inappropriate images due to the educational environment to which it has been directed.

Palabras clave: chat, tiempo real, estudiantes, docentes, comunicación, multimedia, inteligencia artificial, filtrado de imágenes.

Firma del alumno:

VºBº Tutor/es:

Índice de contenidos

Índice de figuras	III
Índice de cuadros y tablas	IV
Índice de bloques de código	v
Índice de diagramas UML	VI
Glosario	VII
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	1
1.3. Metodología de trabajo	2
1.4. Sprints de desarrollo	3
1.4.1. Sprint 1	3
1.4.2. Sprint 2	6
1.4.3. Sprint 3	8
1.4.4. Sprint 4	8
1.4.5. Sprint 5	9
1.4.6. Sprint 6	9
1.4.7. Sprint 7	10
1.4.8. Sprint 8	10
1.4.9. Sprint 9	11
1.4.10. Sprint 10	11
1.4.11. Sprint 11	12
1.5. Tecnologías y herramientas	12
1.6. Problemas	14
1.7. Estructura del trabajo	14
2. Estado del arte	16
2.1. Aplicaciones similares	16
2.2. Opinión sobre el estado del arte	18
3. Gestión del proyecto software	20
3.1. Alcance	20
3.1.1. Definición del proyecto	20
3.1.2. Objetivos	21
3.1.3. Limitaciones de uso	21
3.1.4. Entregables	22
3.1.5. Criterios de aceptación	22
3.1.6. Restricciones	23
3.2. Planificación	23
3.3. Gestión de recursos	23

3.3.1. Especificación de recursos	23
3.3.2. Asignación de recursos	24
3.4. Presupuesto	25
3.5. Gestión de riesgos	26
3.5.1. Identificación de riesgos	26
4. Contenido	27
4.1. Patrones de diseño	27
4.1.1. Builder	27
4.1.2. Singleton	28
4.1.3. Strategy	29
4.2. Procesamiento de los mensajes	31
4.3. Ciclo de vida de la conexión de usuarios	33
4.3.1. Frontend	33
4.3.2. Backend	34
4.4. Docker	35
4.4.1. Prometheus (Métricas)	37
4.4.2. Grafana (Panel de observabilidad)	39
4.4.3. Loki (Agregación de logs)	41
4.4.4. Promtail (Agente de logs)	41
4.4.5. NSFWPY	43
4.5. Base de datos	45
4.6. Seguridad	46
4.7. Pruebas	47
4.7.1. Pruebas unitarias	48
4.7.1.1. Pruebas de caja blanca	48
4.7.1.2. Pruebas de caja negra	50
4.8. Problemas en el desarrollo	51
4.8.1. Alternativas a Heroku	52
4.9. Preparación del servidor	53
4.9.1. Instalación del sistema operativo	53
4.9.2. Configuración de la red	54
4.9.3. Instalación y ejecución de la aplicación	54
4.9.4. Configuración de la aplicación	54
4.9.5. Configuración de la base de datos	55
4.9.6. Seguridad del servidor	55
5. Anexos	56
5.1. Anexo A: Pila del producto	56
Bibliografía	70

Índice de figuras

1.1. Pila del producto en Notion	3
2.1. Lista de usuarios conectados en una sala de chateagratís.net.	16
2.2. Mensajes de información del servidor en una sala de chat de chateagratís.net.	17
2.3. Lista de usuarios conectados y mensaje de información en una sala de chat de dalechatea.net.	17
2.4. Chat de la página web de chatsfriends.com.	18
2.5. Chat completo de la página web de dalechatea.me.	18
4.1. Infraestructura docker. Fuente: https://www.docker.com/resources/what-container	36
4.2. Visualización de métricas relacionadas con los recursos del sistema utilizados y tiempo de actividad.	40
4.3. Visualización de métricas relacionadas con los logs y peticiones HTTP.	40
4.4. Flujo de logs entre Promtail y Loki.	42
4.5. Predicción para una imagen de fondo de pantalla.	44
4.6. Predicción para una imagen pornográfica.	44
4.7. Diagrama de las tablas de la base de datos.	46
4.8. Grafo de control de flujo del método <code>probabilityOfClass</code>	49

Índice de cuadros y tablas

3.1. Especificación de recursos y sueldos	23
3.2. Coste de los recursos	24
3.3. Coste del hardware	24
3.4. Coste del software	25
3.5. Presupuesto de RS Chat	25
4.1. Relación entre método HTTP y ruta.	29
4.2. Relación Mensaje - Estrategia	31
4.3. Definición de las clases de equivalencia.	51
4.4. Casos de prueba para la funcionalidad de consultar emojis por nombre.	51
4.5. Valores para los casos de prueba.	51
4.6. Nomenclatura utilizada en las tablas de casos de prueba.	51

Índice de bloques de código

4.1.	Configuración mínima para ejecutar un contenedor con Grafana.	39
4.2.	Servicio de Loki para el registro de logs.	41
4.3.	Servicio de Promtail para la recolección de logs.	42
4.4.	Stage para obtener el nombre del contenedor con una expresión regular a partir de la etiqueta <code>tag</code>	43
4.5.	Servicio de NSFWPY para la detección de imágenes NSFW.	44
4.6.	Método para determinar la clase NSFW asociada a una imagen	49

Índice de diagramas UML

4.1. Patrón Builder empleado en la aplicación.	28
4.2. Patrón Singleton empleado en la aplicación.	29
4.3. Interfaz MessageStrategy.	30
4.4. Clase ChatManagement para la gestión de los chats.	35

Glosario

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

Bot Es un programa que se ejecuta en un servidor y que puede interactuar con los usuarios.

Bundler Es una herramienta que permite la creación de aplicaciones web a partir de ficheros de código fuente.

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.

Framework Es un conjunto de herramientas que facilitan el desarrollo de aplicaciones.

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.

NSFW Es un acrónimo de Not Safe For Work, que significa que el contenido no es apto para el trabajo.

popup Es una ventana emergente que se abre en el navegador.

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, cuando se estaban buscando temáticas para un proyecto personal. El primer paso fue decidir qué tecnologías utilizaría para el desarrollo de la parte backend. Se tenían diferentes opciones para realizarlo: 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 más empresas en un futuro y una forma de aprender una nueva tecnología. Además, se vio que era una tecnología 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 empleado en la misma asignatura y es más fácil de aprender y usar que otras tecnologías como *Angular* o *VueJS*. Este proyecto se comenzó en junio 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 principal de la aplicación era su integración de forma completa con la plataforma de *Moodle* de la Universidad de León (como un complemento a los foros), para que tanto estudiantes y profesores pudieran emplearla e interactuar de manera más rápida. 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).

Actualmente, en cualquier aplicación de chat, se permite compartir archivos, imágenes, vídeos, etc. con los demás usuarios. Se pueden 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 permite a los usuarios 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**.

Los canales **privados** solo pueden ser vistos por los usuarios que hayan sido invitados a ellos de manera explícita o con el código de invitación. Los canales **públicos** son accesibles por cualquier persona que tenga acceso a la aplicación, sin necesidad de invitación. También se pueden enviar mensajes a una sola persona, estableciendo una conversación privada sin notificar al resto de usuarios conectados.

Una de las características que no ofrece ninguna aplicación de chat es la posibilidad de filtrar imágenes inapropiadas mediante un algoritmo de inteligencia artificial, por lo que este es uno de los objetivos principales de la aplicación.

1.3. METODOLOGÍA DE TRABAJO

Para el desarrollo de la aplicación se ha utilizado el marco de trabajo **SCRUM**, que se emplea para la gestión de proyectos ágiles de manera **iterativa** e **incremental**. Iterativa porque se revisa y mejora el producto existente en cada ciclo de desarrollo (**sprint**), e incremental porque las nuevas funcionalidades se integran en la aplicación a medida que se completan.

Todas las funcionalidades a implementar se denominan **historias de usuario** y forman lo que se denomina Pila del Producto (**Product Backlog**). En cada iteración, se seleccionan las historias de usuario que se consideran más prioritarias y se asignan a los miembros del equipo de trabajo, formando la Pila del Sprint (**Sprint Backlog**).

La duración de las iteraciones en SCRUM es variable, pero se recomienda que no sea inferior a 2 semanas. En este proyecto se ha optado por una duración de 4 semanas, ya que el tiempo de desarrollo de cada iteración debe ser suficiente para que el desarrollador pueda completar todas las tareas que se le han asignado durante la misma.

Puesto que el equipo de trabajo está formado por un único miembro, no ha sido necesario realizar reuniones de coordinación, porque se ha podido trabajar de manera autónoma y sin necesidad de supervisión. Además, se ha adaptado el marco de trabajo para que se ajuste a las necesidades del proyecto.

La gestión de la pila del producto se ha realizado con la herramienta **Notion**, que permite crear espacios de trabajo con diferentes vistas para organizar las tareas. En la figura 1.1 se presenta una pequeña muestra de la pila del producto. Para una descripción más detallada y completa de las tareas realizadas durante el desarrollo del

proyecto, se puede consultar el anexo 5.1.

Task	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Spring migration from 2.7.0 to 3.0.2	High	Refactor	Complete	Sprint 08	January 26, 2023 16:45	January 26, 2023 22:00	1.6	
Huge refactor to use JPA Buddy when generating entities and relationships	Critical	Refactor	Complete	Sprint 08	January 22, 2023 17:34	January 24, 2023 21:46	24	https://www.youtube.com/watch?v=DC6FrC40hE
Add teachers to degree by administrator	Very high	Improvement	Complete	Sprint 08	January 22, 2023 16:19	January 22, 2023 17:15	0	Teachers are added implicitly to the degrees when they are added to a subject. Modify admin dashboard to show the degree of each subject
Special rooms for games	Very low	Idea	Not Started					TicTacToe, Bingo...
Refactor of if statements in Policies class to use Yavi validation library	Very high	Refactor	Complete	Sprint 08	January 20, 2023 12:35	January 20, 2023 23:42	3	
Add Quartz lib to schedule all the tasks.	Medium	Improvement	Cancelled	Sprint 08	January 15, 2023 13:50	January 15, 2023 14:00	0	
Improve the security of my home Wi-Fi and network	High	Security	Not Started					
Implement rate limit to avoid spamming a lot of messages in short time	High	Improvement	Complete	Sprint 08	February 1, 2023 0:40	February 1, 2023 15:24	8	Messages that are sent after the notification of TOO_FAST_MESSAGE are deleted from the user chat view
Add relations to entities to reduce queries	Low	Refactor	Reference					See Huge refactor to use JPA Buddy when generating entities and relationships
Fix /dice command. Page crashes	Very high	Bug	Complete	Sprint 07	December 21, 2022 0:20	December 21, 2022 0:50	0	
Add teacher to subject by administrator	High	Task	Complete	Sprint 07	December 18, 2022 14:52	December 21, 2022 17:44	28	2 columns in UI: Left with teacher names and id. Right with subject names. With a search bar for each column
Teacher service to get data	High	Task	Complete	Sprint 07	December 18, 2022 2:08	December 18, 2022 4:00	0	
When sending a non-text message, the Handler crashes because the content could not be read as normal text.	Very high	Bug	Complete	Sprint 07	December 17, 2022 20:35	December 17, 2022 22:57	1	Check the type before the content
Know how to add custom stats to send to Prometheus	Medium	Improvement	Complete	Sprint 07	December 17, 2022 2:49	December 17, 2022 3:20	0	https://habianlee.org/2022/06/29/java-adding-custom-metrics-to-spring-boot-micrometer-prometheus-endpoint/

Figura 1.1: Pila del producto en Notion

1.4. SPRINTS DE DESARROLLO

En esta sección, se describen los sprints de desarrollo que se han realizado durante el proyecto. Para cada uno de ellos, se indica la fecha de inicio y fin, el objetivo y la descripción de las tareas realizadas. Se ha decidido no incluir la pila de producto de cada sprint, ya que se puede consultar en el tablero de Notion del proyecto (ver anexo 5.1).

1.4.1. SPRINT 1

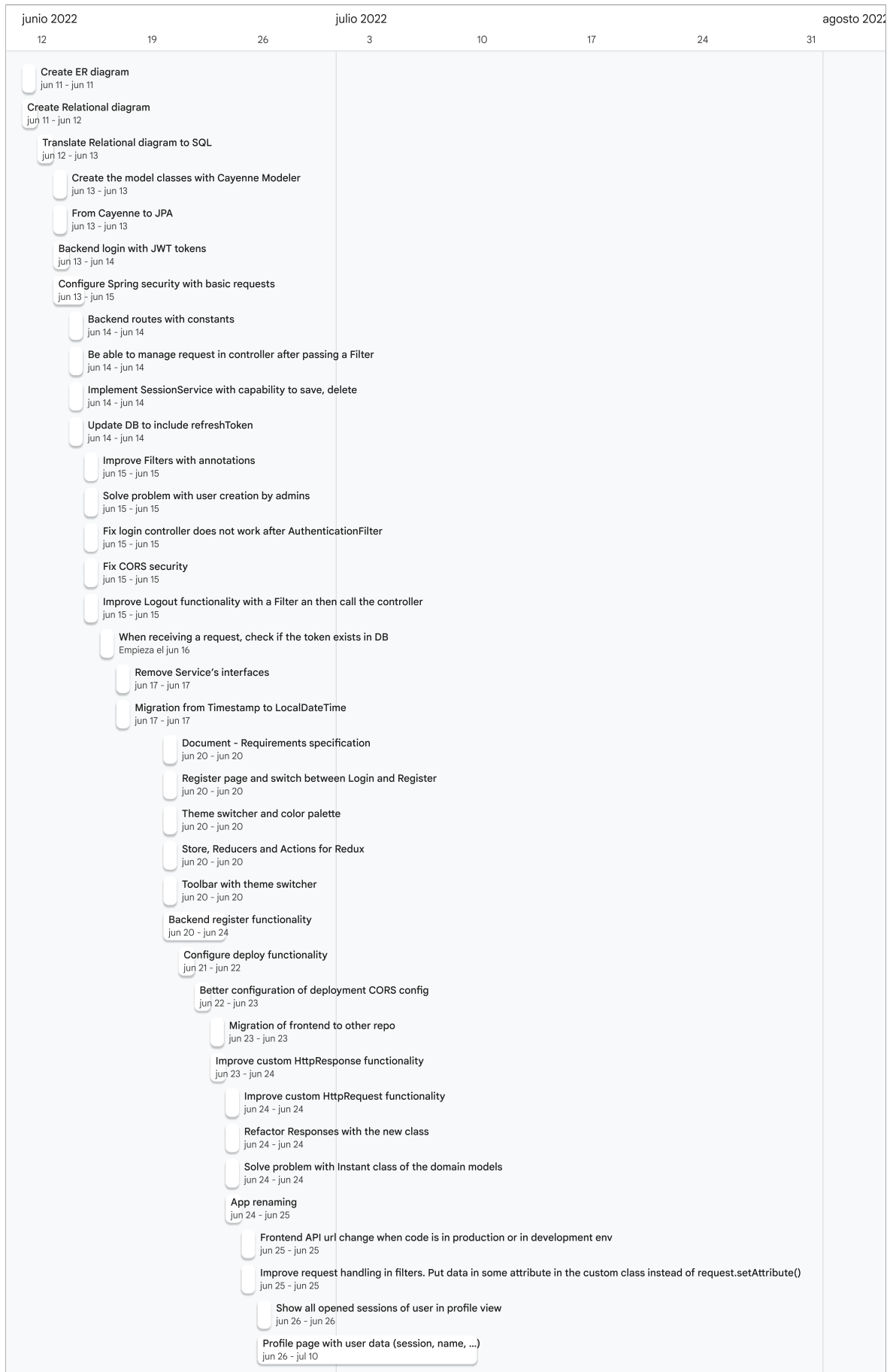
Fecha de inicio: 11/06/2022

Fecha de fin: 11/07/2022

Objetivo: Crear un prototipo de la aplicación.

Descripción: En este sprint se pretende crear un prototipo de la aplicación, con el fin de tener una primera versión de la aplicación que se pueda probar y que sirva como base para el desarrollo de la versión final. Se desarrollará una API con seguridad básica y se creará una aplicación web que consuma dicha API. Las rutas de la última, se dividirán según el tipo de petición que se realice, es decir, habrá rutas para las peticiones GET, POST, PUT y DELETE. Se creará una base de datos con los modelos

necesarios para la aplicación y se crearán los controladores para que la API funcione correctamente. Se creará una aplicación web con React que la consuma y que permita realizar las operaciones básicas de la aplicación. Además, es necesario un endpoint que permita la conexión por WebSocket al servidor, para poder realizar la comunicación en tiempo real entre los usuarios.



1.4.2. SPRINT 2

Fecha de inicio: 11/07/2022

Fecha de fin: 11/08/2022

Objetivos:

- Mantener un historial de mensajes por cada chat.
- Añadir mejoras de seguridad.

Descripción: En este sprint se pretende añadir un historial de mensajes por cada chat, de forma que se pueda acceder a los mensajes anteriores. Además, se añadirán mejoras de seguridad a la aplicación, como por ejemplo, la encriptación de las contraseñas de los usuarios utilizando otro algoritmo más seguro. También se añadirá un sistema de permisos para los usuarios, de forma que se pueda restringir el acceso a ciertas funcionalidades de la aplicación.

julio 2022	agosto 2022	septiembre 2022	octubre 2022
17	24	31	7
14	21	28	4
11	18	25	2
<div>Add support for all types of files (text, images, videos, audios) • ago 1 - ago 3</div> <div>Spinner on login or register, to know that the request has been sent • ago 11 - ago 11</div> <div>New messages when user joins and leaves a chat • ago 3 - ago 4</div> <div>Update docs • ago 5 - ago 7</div> <div>Check token on connect to websocket • ago 5 - ago 15</div> <div>Create updated entities in JPA • ago 5 - ago 6</div> <div>Make administrator view in frontend • ago 6 - sept 13</div> <div>Make separate classes for Route constants (GET, POST, ...) • ago 6 - ago 6</div> <div>Implement Degree controller and associated Service in backend • ago 6 - ago 6</div> <div>Update old methods that used ResponseEntity<?> to use the new HttpResponse class • ago 6 - ago 6</div> <div>Check that all routes have the correct http method and permissions • ago 7 - ago 7</div> <div>Document - Design model • ago 9 - ago 9</div> <div>Domain diagram • ago 9 - ago 9</div> <div>Use case diagram • ago 9 - ago 11</div> <div>Add custom Drop Down, password reset and more info in profile • jul 16 - jul 17</div> <div>Basic chat page • jul 19 - jul 19</div> <div>More style to chat and messages • jul 20 - jul 20</div> <div>Add a new table in DB (Chats) • jul 21 - jul 21</div> <div>Move header of message up one component (to ChatMessage.jsx) • jul 21 - jul 21</div> <div>Send messages from front to back • jul 21 - jul 21</div> <div>Update security of the routes. Make a separate class or something • jul 21 - jul 21</div> <div>Read messages in backend • jul 21 - jul 21</div> <div>Store messages in a file with history • jul 21 - jul 21</div> <div>When sending an empty message, write nothing in the response • jul 23 - jul 23</div> <div>Create WebSocket server • jul 23 - jul 24</div> <div>Send messages to other members of the chat • jul 24 - jul 26</div> <div>Configure frontend to send messages. Create class and Hook • jul 25 - jul 26</div> <div>Receive messages in frontend from other clients and log to console • jul 26 - jul 26</div> <div>Add the sessionId when sending the response to the client • jul 26 - jul 26</div> <div>Improve message handling between front and back • jul 26 - jul 26</div> <div>Clear the sensitive headers to send the received message to other clients in backend • jul 26 - jul 26</div> <div>Improve storage of clients in backend • jul 26 - jul 27</div> <div>Check the date in which the message was sent, to write correctly to the file • jul 27 - jul 27</div> <div>Receive message in component • jul 27 - jul 27</div> <div>Make a record class to identify the WS user in backend • jul 27 - jul 27</div> <div>Show incoming messages in ChatBox • jul 27 - jul 28</div> <div>Provide more style to message components • jul 28 - jul 28</div> <div>Make a deploy test in Heroku, to verify the functionality of the sockets • jul 28 - ago 5</div> <div>Migrate from Java-WebSocket lib to Jetty websockets in backend • jul 29 - jul 30</div> <div>Migrate from native WebSocket to websocket library in frontend • jul 30 - jul 30</div> <div>Migrate backend to spring websocket support with custom Handlers • jul 30 - jul 31</div> <div>Better handling of chats with a new class • jul 31 - jul 31</div> <div>Add a Writer to support message storage in S3. Upload files • jul 31 - jul 31</div> <div>When uploading file (finished chat) delete from disk. If chat is reopened, download file and keep writing • jul 31 - ago 1</div>			

1.4.3. SPRINT 3

Fecha de inicio: 11/08/2022

Fecha de fin: 11/09/2022

Objetivos:

- Envío de mensajes de conexión y desconexión de usuarios.
- Recuperación y guardado del historial de mensajes.
- Soporte de conexión “idle”.

Descripción: En este sprint se pretende añadir la funcionalidad de envío de mensajes de conexión y desconexión de usuarios, de forma que se pueda notificar a los usuarios cuando ocurra cualquiera de esos eventos. Además, cuando el **primer usuario** se conecte, se recuperará el historial de mensajes del chat al que se ha conectado. De manera contraria, cuando el **último usuario** se desconecte, se guardará el historial de mensajes del chat en el servicio de almacenamiento en la nube S3 de Amazon Web Services. Por último, se añadirá soporte de conexión “idle”, de forma que si un usuario se ausenta durante un tiempo y no realiza ninguna acción, se garantiza que no se le desconecte del chat.

Imprevistos: Heroku ha eliminado su plan gratuito de despliegue de aplicaciones, por lo que se ha tenido que buscar una alternativa para desplegar la aplicación y se ha optado por utilizar Vercel.

1.4.4. SPRINT 4

Fecha de inicio: 11/09/2022

Fecha de fin: 11/10/2022

Objetivos:

- Añadir soporte para el envío de archivos.
- Utilizar Docker para la aplicación.

Descripción: En este sprint se añadirá soporte para el envío de archivos multimedia en los mensajes. Se limitará el tamaño de los archivos a 30 MB, para evitar que se envíen archivos demasiado grandes. Además, se utilizará Docker para la aplicación, de

forma que se pueda desplegar fácilmente en cualquier servidor. Esto también facilita la integración entre todos los servicios utilizados en la aplicación, ya que se pueden desplegar todos a la vez con un solo comando. Se ha realizado la migración de la aplicación a un servidor personal, debido a que Heroku ha eliminado su plan gratuito, como se ha comentado en el sprint anterior. Para más información, consultar la sección 1.6.

1.4.5. SPRINT 5

Fecha de inicio: 11/10/2022

Fecha de fin: 11/11/2022

Objetivos:

- Completar la migración al servidor personal.
- Nuevo sistema de menciones.

Descripción: En este sprint se añadirá la funcionalidad de menciones, de forma que se pueda mencionar a un usuario en un mensaje y que este reciba una notificación. Además, se completará la migración de la aplicación al servidor personal. Los correos electrónicos que se reciben a los usuarios se enviarán sin utilizar ningún servicio externo, ya que hay ocasiones en que ciertos servicios de correo electrónico bloquean los correos que se envían desde estas plataformas, como SendGrid. Debido a esto, se ha creado una cuenta de correo electrónico en Gmail, con el nombre de la aplicación, y se utilizará para enviar los correos electrónicos.

1.4.6. SPRINT 6

Fecha de inicio: 11/11/2022

Fecha de fin: 11/12/2022

Objetivo: Mejora de la observabilidad del sistema.

Descripción: En este sprint se pretende mejorar la observabilidad del sistema utilizando paneles para su administración. Estos paneles se utilizarán para monitorizar el estado del sistema y para visualizar los datos de los logs obtenidos. Todo esto conlleva la creación de 3 contenedores Docker adicionales:

- Prometheus: Sistema de monitorización y alertas.
- Grafana: Sistema de visualización de datos.

- Loki: Sistema de logs.

1.4.7. SPRINT 7

Fecha de inicio: 11/12/2022

Fecha de fin: 11/01/2023

Objetivo: Almacenamiento de estadísticas de mensajes enviados.

Descripción: Para este sprint, se pretenden implementar las estadísticas de mensajes enviados por los usuarios. Esto conlleva la creación de una nueva columna en la tabla `users` de la base de datos, que almacenará el número de mensajes de cada tipo en formato JSON. Se añadirán nuevos comandos a la aplicación y se modificará su implementación para soportar múltiples parámetros.

1.4.8. SPRINT 8

Fecha de inicio: 11/01/2023

Fecha de fin: 11/02/2023

Objetivos:

- **Migración de Spring 2.7 a 3.0**
- Optimización de la lectura de historiales de mensajes.
- Limitación de mensajes enviados por segundo.

Descripción: En este sprint se pretende migrar la aplicación a Spring 3.0, ya que añade nuevas mejoras y funcionalidades. Esta migración conlleva la modificación de la configuración de seguridad de la aplicación y la actualización de las dependencias a su última versión.

La lectura de mensajes era muy lenta, porque cada vez que un usuario se conectaba, se descargaba el fichero de mensajes completo. Para solucionar esto, se ha implementado un sistema de caché que almacena los mensajes en memoria.

Para evitar que los usuarios envíen mensajes de forma masiva, se ha implementado un limitador de mensajes por segundo. Es de implementación propia y se basa en un contador para cada usuario, que se reinicia cada segundo. Si el contador supera el límite, no se permite el envío de mensajes hasta que se reinicie.

1.4.9. SPRINT 9

Fecha de inicio: 11/02/2023

Fecha de fin: 11/03/2023

Objetivos:

- Mejor implementación de las respuestas HTTP.
- **Uso de un modelo de inteligencia artificial para la detección de imágenes inapropiadas.**

Descripción: Para este sprint se realizará un completo rediseño de la implementación de las respuestas HTTP, ya que se debe facilitar su uso y la creación de nuevas respuestas en cada controlador.

Se utilizará un modelo de inteligencia artificial para la detección de imágenes inapropiadas. Se realizará un estudio de los modelos de inteligencia artificial existentes para la detección de imágenes inapropiadas y se elegirá el más adecuado para el proyecto. Para la aplicación se ha usado un modelo pre-entrenado de código abierto que se ha obtenido del repositorio de GitHub *GantMan/nsfw_model* [1]. La implementación de este modelo se realizará en un nuevo servicio utilizando NodeJS y utilizando la librería TensorFlowJS.

1.4.10. SPRINT 10

Fecha de inicio: 11/03/2023

Fecha de fin: 11/04/2023

Objetivo: Migración del filtro de imágenes NSFW a Python y bloqueo de usuarios.

Descripción: La integración del modelo de inteligencia artificial con JavaScript no ha funcionado como se esperaba, por lo que se ha decidido migrar el código a Python, que es el lenguaje en el que se ha desarrollado el modelo. Esto ofrece una mejora de rendimiento a la hora de procesar las imágenes que no se puede conseguir con JavaScript. Además, se ha implementado un sistema de bloqueo de usuarios para evitar que envíen más de 5 imágenes inapropiadas. En caso de que lo hagan, se les bloqueará la cuenta durante 24 horas.

1.4.11. SPRINT 11

Fecha de inicio: 11/04/2023

Fecha de fin: 11/05/2023

Objetivo: Mantenimiento y corrección de errores.

Descripción: A partir de este sprint, el equipo de desarrollo se centrará en la corrección de errores y el mantenimiento del producto.

1.5. 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 de la parte backend de la aplicación.
- **Spring Boot**: es un framework de Java que permite la creación de aplicaciones web. Además del paquete básico, se han utilizado los paquetes **Security**, **Data JPA**, **Web**, **Websocket**, **Mail**, y **Test**.
- **MySQL**: es un sistema de gestión de bases de datos relacional que ha sido utilizado para el almacenamiento de los datos de la aplicación.
- **JUnit**: es un framework de Java que permite la realización de pruebas unitarias.
- **Mockito**: es otro framework de Java que permite la realización de pruebas unitarias mediante el uso de mocks, que son objetos simulados que facilitan la realización de pruebas independientes.
- **Lombok**: es una biblioteca de Java que ayuda a reducir la cantidad de código repetitivo, generando automáticamente los métodos **getters** y **setters** de las clases, así como los constructores, **equals**, **hashCode**, etc.
- **Logback**: es una biblioteca que permite la creación de logs de forma sencilla.
- **Maven**: es un gestor de dependencias de Java que facilita la descarga e integración de las librerías utilizadas en la aplicación. El fichero **pom.xml** tiene una estructura XML donde se añaden todas las dependencias a utilizar empleando etiquetas concretas, proporcionando su nombre, versión e identificador.

- **JavaScript:** es el lenguaje de programación utilizado para la implementación de la parte frontend de la aplicación.
- **React:** es una biblioteca de JavaScript que permite la creación de interfaces de usuario mediante componentes.
- **Redux:** es una biblioteca de JavaScript que permite la gestión del estado de la aplicación.
- **React Router:** es una biblioteca que permite la creación de rutas en aplicaciones React.
- **Axios:** es una biblioteca que permite realizar peticiones HTTP desde aplicaciones JavaScript de manera sencilla.
- **Material-UI:** es una biblioteca que permite la creación de interfaces utilizando el diseño de Material Design de Google.
- **Vite:** es un bundler de JavaScript que permite la creación de aplicaciones web de forma rápida.
- **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.
- **Vercel:** es un servicio de hosting que permite el despliegue de la parte frontend de la aplicación web.
- **AWS S3:** es un servicio de almacenamiento de Amazon que ha permitido el guardado de los ficheros asociados a la aplicación, como los archivos multimedia, historiales de chat, etc.
- **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 L^AT_EX.

- **Notion:** es una aplicación que ha permitido la organización de las tareas a realizar durante el proyecto.

1.6. 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, se desplegó en dos clusters con el plan gratuito de Heroku (uno para el frontend y otro para el backend), pero se ha tenido que retirar la aplicación de esta plataforma 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 han considerado varias opciones:

- Despliegue en *Platform.sh*: plataforma de despliegue de aplicaciones en la nube.
- Despliegue en un servidor propio.

La primera opción no ha sido posible, ya que, tras una reunión con el encargado de *Cloud Services* de la plataforma, no se ha podido conseguir una cuenta gratuita para el despliegue de la aplicación durante su desarrollo. Por lo tanto, se ha optado por la segunda opción, comprando un pequeño ordenador personal y desplegando el backend en él. De esta manera, también se abarca el campo de la administración de sistemas.

1.7. ESTRUCTURA DEL TRABAJO

Este documento se divide en varios capítulos, cada uno de los cuales se centra en un aspecto diferente del proyecto. A continuación se incluye una breve descripción de cada uno de ellos, sin entrar en detalles sobre sus secciones:

1. **Estado del arte:** se realiza un análisis y comparación de las páginas web existentes para la interacción en tiempo real entre usuarios, de forma que permita la elección del diseño y funcionalidad más adecuada para el proyecto.
2. **Gestión del proyecto software:** en este capítulo se detallarán todos los aspectos relacionados con el alcance, presupuesto, recursos necesarios para el desarrollo, entre otros conceptos asociados a la gestión de un proyecto software.
3. **Contenido:** se describe todo lo relativo a la aplicación web y cómo se ha desarrollado cada una de sus partes. Además, se incluyen diagramas para facilitar la

comprensión de la estructura de la aplicación, servicios y la distribución de los componentes.

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, para determinar las funcionalidades generales que ofrecen y poder tomar una decisión sobre las nuevas funcionalidades que se implementarán en la aplicación. La búsqueda se ha realizado en diferentes idiomas (español e inglés) y se han encontrado diferentes páginas web y aplicaciones que ofrecen este tipo de servicio, que son las siguientes:

- **Chateagratís.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 bots 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 @).

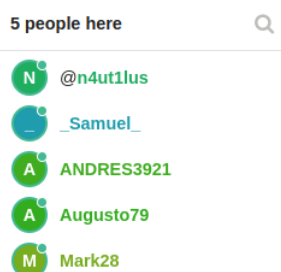


Figura 2.1: Lista de usuarios conectados en una sala de chateagratís.net.



Figura 2.2: Mensajes de información del servidor en una sala de chat de chateagratias.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. Esta funcionalidad solo está disponible en los chats privados, no en los públicos. Además, se pueden agregar amigos, bloquear usuarios y añadir reacciones a mensajes de otros usuarios (como mandar un saludo o añadir marcadores), entre otras funciones.

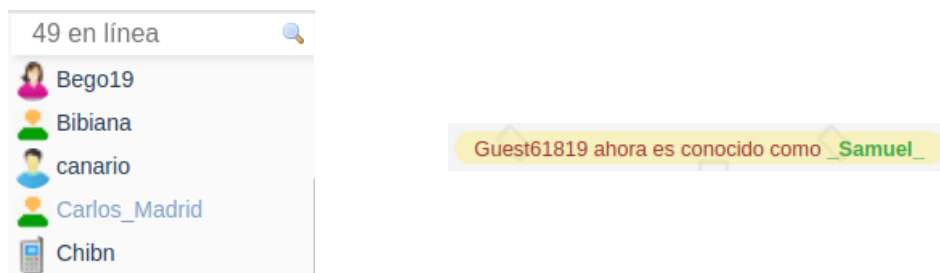


Figura 2.3: Lista de usuarios conectados y mensaje de información en una sala de chat de dalechatea.net.

- **Chatsfriends.com:** Esta página web ofrece las mismas funcionalidades que las anteriores, pero con un diseño renovado y diferente. Además, en la parte inferior (donde se escribe el mensaje) se muestra el nombre del usuario que será mostrado cuando se envíe el mensaje. A continuación se muestra una imagen de la página web, en la que se puede observar el chat en la parte central de la pantalla y un listado de usuarios en la parte derecha.

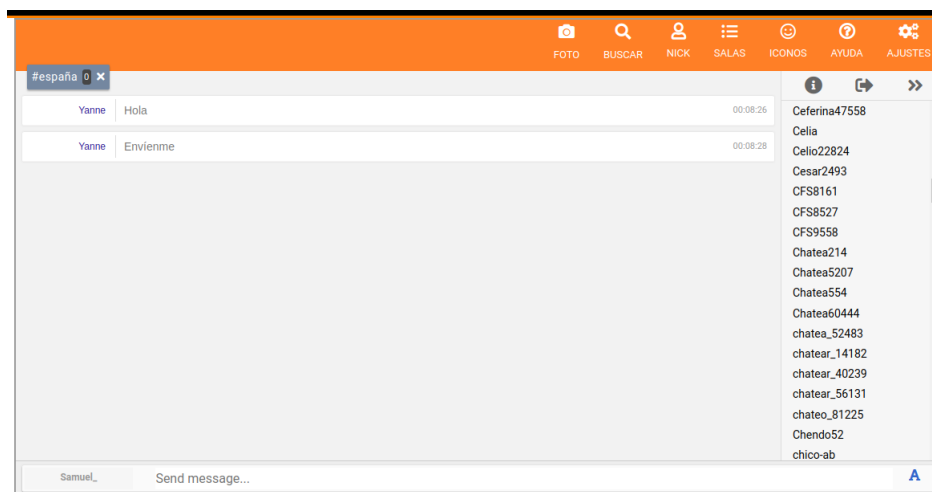


Figura 2.4: Chat de la página web de chatsfriends.com.

Todas las 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 2.5). 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 mensajes de información, que se envían por parte del servidor del chat, para notificar a todos los usuarios conectados al mismo de cualquier evento. Cuando se hace clic en un usuario de la lista, se muestra un pequeño popup con la información del usuario seleccionado, que muestra su nombre de usuario y ciertos botones para realizar las acciones previamente mencionadas.

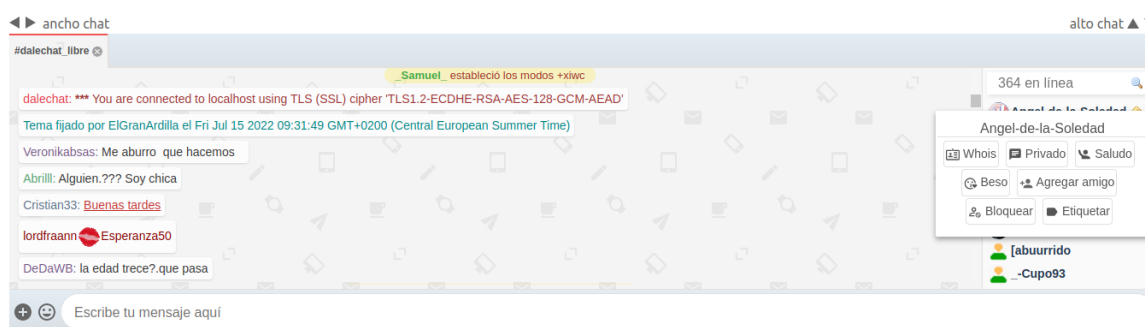


Figura 2.5: Chat completo de la página web de dalechatea.me.

2.2. OPINIÓN SOBRE EL ESTADO DEL ARTE

Haciendo una comparación entre las diferentes páginas web presentadas en la sección anterior, se observa que ninguna de ellas dispone de las siguientes funcionalidades: un

sistema de notificación auditiva de mensajes recibidos, creación de grupos de chat personalizados y la posibilidad de enviar archivos multimedia en los chats públicos.

- **Notificación de mensajes:** En estas páginas web, cuando un usuario recibe un mensaje, no se le notifica de forma inmediata, sino que debe estar atento al chat para ver si aparece un mensaje nuevo. Esto puede ser molesto para los usuarios que no pueden prestar atención a la aplicación durante todo el tiempo que están conectados. Por ello, se ha decidido implementar un sistema de notificación de mensajes, que alerte al usuario cuando recibe un mensaje, de forma que pueda continuar con sus tareas sin tener que estar pendiente de la pantalla.
- **Creación de grupos de chat:** En la mayoría de las páginas web y aplicaciones de chat, los usuarios pueden unirse a salas de chat públicas, pero no pueden crear salas privadas. Esto puede no ser de agrado para algunos usuarios, que no pueden crear grupos de chat personalizados para comunicarse con sus amigos o familiares. Por ello, se ha decidido implementar un sistema de creación de grupos de chat, que permita a todos los usuarios crear grupos de chat personalizados para comunicarse con las personas que deseen.
- **Envío de archivos multimedia en los chats públicos:** En estas páginas web de chat, los usuarios pueden enviar distintos tipos de archivos multimedia en los chats privados, pero no en los públicos. Esto puede no satisfacer a todos los usuarios, ya que limita la comunicación entre los usuarios en los chats públicos. Por ello, se ha decidido implementar en todos los chats (ya sean públicos o privados) un sistema de envío de archivos multimedia.
- **Filtrado de imágenes inapropiadas:** En estas páginas web, los usuarios pueden enviar imágenes de cualquier tipo, incluyendo pornografía y otros contenidos inapropiados. Esto puede ser una molestia para la mayoría de los usuarios, que no quieren ver este tipo de imágenes. Por ello, se ha decidido implementar un sistema de filtrado de este tipo de contenido y que impida su envío en los chats. Además, cuando un usuario envíe un cierto número de imágenes pornográficas, se le bloqueará el acceso a la aplicación durante un tiempo determinado.

3. Gestión del proyecto software

En este capítulo se describen los aspectos relativos a la gestión del proyecto software. En primer lugar, se describe el alcance del proyecto, que incluye los objetivos y los productos que se van a desarrollar, además de la definición del proyecto, limitaciones de uso y entregables, entre otros. Seguido a esto, se describe la planificación del proyecto llevada a cabo, es decir, las actividades a realizar y los recursos necesarios para completarlas de manera efectiva. Después, se enumeran los recursos humanos y materiales que se van a necesitar para continuar con el proyecto, además de describir el presupuesto del mismo, incluyendo el coste para llevar a cabo el proyecto. Por último, se describen los riesgos a afrontar durante el desarrollo del proyecto.

3.1. ALCANCE

3.1.1. DEFINICIÓN DEL PROYECTO

El proyecto consiste en el desarrollo de una aplicación web para la comunicación mediante mensajes e intercambio de archivos entre los usuarios. Gracias a los diferentes roles en los que se distinguen los usuarios dentro de la aplicación, se puede realizar una comunicación más fluida entre los mismos, ya que los profesores pueden crear grupos (además del ya existente para cada asignatura) para, por ejemplo, dividir a los alumnos en grupos de trabajo y así no interferir con el resto del alumnado. También cabe la posibilidad de que cada usuario pueda abrir un chat privado con cualquier otro que esté presente en los chats a los que pertenece, abriendo una nueva conversación. Los administradores son los encargados de crear las asignaturas y los profesores, así como de gestionar los usuarios de la aplicación.

Con respecto al diseño, se ha optado por una interfaz sencilla y fácil de usar, minimalista y funcional. Se dispone de adaptabilidad a diferentes dispositivos, por lo que la aplicación es accesible desde cualquier sistema con conexión a Internet. Además, se ha optado por un diseño *responsive*, lo que permite que la aplicación se adapte a la pantalla del dispositivo en el que se esté visualizando, ya sea un ordenador de sobremesa, un portátil, una tablet o un teléfono móvil.

La implementación del proyecto se divide en frontend y backend. El frontend es el encargado de mostrar las vistas y de gestionar las interacciones con el usuario, mientras

que el backend es el encargado de controlar la lógica de la aplicación, la base de datos y la comunicación con el frontend.

Debido a que la aplicación necesita que los usuarios se registren para poder utilizarla, se implementa una política de privacidad para que los usuarios conozcan cómo se van a tratar sus datos. Éstos no se compartirán con ninguna otra entidad, salvo que el usuario lo autorice. Además, se ha introducido un sistema de recuperación de contraseña para que los usuarios puedan restablecer su contraseña en caso de que la hayan olvidado.

3.1.2. OBJETIVOS

Los objetivos del proyecto son los siguientes:

- Desarrollar una aplicación web para la comunicación entre los usuarios.
- Establecer un sistema de roles, para permitir la interacción entre usuarios de manera efectiva y fluida. Esto ayuda a los usuarios a determinar a qué personas comunicar incidencias u otra información.
- Se debe ofrecer un sistema de grupos y chats individuales para los usuarios.
- Los usuarios deben poder intercambiar archivos entre ellos.
- Prevenir la compartición de imágenes inapropiadas.
- Los administradores deben tener un panel de control para gestionar todo lo relacionado con la aplicación y el uso de recursos por parte del servidor, para evitar sobrecargas o caídas.
- Garantizar el debido cumplimiento de la política de privacidad de los usuarios.
- Ofrecer un sistema de recompensa a los usuarios que más utilicen la aplicación, otorgando insignias que se mostrarán en su perfil.

3.1.3. LIMITACIONES DE USO

Para garantizar el correcto funcionamiento de la aplicación, se establecen las siguientes restricciones de uso:

- Se garantiza una buena visualización de la aplicación en dispositivos de cualquier resolución, tanto dispositivos móviles como de escritorio, siempre que dispongan de conexión a Internet y un navegador web actualizado.
- El funcionamiento del sistema se ha probado para el navegador Firefox y los basados en **Chromium**, como Google Chrome y Brave. No se garantiza el correcto funcionamiento en otros navegadores web.
- No se requiere instalación de ningún programa externo. De esta manera, se simplifica el acceso a la aplicación y se evitan problemas de compatibilidad con el sistema operativo.
- Para el almacenamiento de archivos, se utilizará un sistema de almacenamiento en la nube, siguiendo una estructura de directorios dependiendo del tipo de archivo que los usuarios suban.
- Durante el desarrollo del proyecto, no se utilizarán recursos que incumplan la ley de protección de datos ni la ley de propiedad intelectual.

3.1.4. ENTREGABLES

3.1.5. CRITERIOS DE ACEPTACIÓN

Para que el proyecto se considere aceptado, se deben cumplir una serie de requisitos, que son los siguientes:

- Los puntos mencionados en el apartado 3.1.2 deben estar implementados.
- Con el fin de garantizar la ausencia de errores, se debe realizar una serie de pruebas por parte del equipo de desarrollo y también por parte de los usuarios. En caso de existir algún error, se debe corregir antes de que el proyecto sea aceptado.
- Los casos de uso deben estar implementados y funcionando correctamente.
- Los test unitarios, de aceptación e integración deben pasar sin errores.

3.1.6. RESTRICCIONES

3.2. PLANIFICACIÓN

3.3. GESTIÓN DE RECURSOS

En este capítulo se describen los recursos humanos y materiales que se han necesitado para el desarrollo del proyecto.

3.3.1. ESPECIFICACIÓN DE RECURSOS

Cuando se habla de recursos humanos, en términos de un proyecto, se hace referencia a las personas que van a participar en el mismo. Los recursos materiales son los equipos y herramientas que se van a utilizar para llevar a cabo el proyecto. En el caso de este, que ha sido implementado enteramente por una persona, se simplifican los cálculos, ya que se conocen de manera más precisa. A continuación, se presenta una tabla con la especificación de los recursos para cada rol que se ha desempeñado en el desarrollo del proyecto:

Recursos	Sal. Anual	Sal. Mensual	Sal. Diario	Sal. Hora
Desarrollador backend	31,412 €	2,243.71 €	112.18 €	14.02 €
Desarrollador frontend	37,123 €	2,651.64 €	132.58 €	16.57 €
Diseñador	22,417 €	1,601.21 €	80.06 €	10.00 €
Tester	28,036 €	2,002.57 €	100.12 €	12.51 €
Técnico de sistemas	24,422 €	1,744.42 €	87.22 €	10.90 €

Cuadro 3.1: Especificación de recursos y sueldos

Se ha considerado que la jornada laboral es de 8 horas diarias, 5 días a la semana, y 14 pagas al año. Los datos de la columna *Sal. anual* se han obtenido del buscador de sueldos de *Indeed* [2] a fecha 8 de junio de 2023. Esta web se encarga de calcular el sueldo promedio de los trabajos ofrecidos en España para un puesto en concreto. El resto de datos se han calculado con las siguientes fórmulas:

$$\text{Sal. Mensual} = \frac{\text{Sal. Anual}}{14}$$

$$\text{Sal. Diario} = \frac{\text{Sal. Mensual}}{20}$$

$$\text{Sal. Hora} = \frac{\text{Sal. Diario}}{8}$$

3.3.2. ASIGNACIÓN DE RECURSOS

Una estimación del trabajo que se ha realizado con el coste asociado se puede ver en la tabla 3.2. Se ha calculado el tiempo que se ha dedicado a cada rol y el coste asociado a cada uno de ellos utilizando los datos de la pila del producto (ver anexo 5.1).

Recursos	Trabajo (horas)	Coste (€)
Desarrollador backend	2000	28,046.42 €
Desarrollador frontend	1500	24,859.15 €
Diseñador	300	3,002.27 €
Tester	500	6,258.03 €
Técnico de sistemas	700	7,631.87 €
Totales	5000	69,797.74 €

Cuadro 3.2: Coste de los recursos

La razón por la que el coste total del proyecto y horas dedicadas están más inflados de lo que debería, es porque se ha desarrollado por una sola persona y se ha dispuesto de **tiempo extra** para dedicarle al mismo.

Los costes del material necesario para la realización del proyecto se han dividido en 2 categorías: hardware y software. En las siguientes dos tablas se puede observar el desglose de cada uno de ellos.

Hardware	Coste (€)	Unidades	Total (€)
Ordenador	1,442 €	1	1,442 €
Ratón	23.88 €	1	23.88 €
Monitor Secundario	40 €	1	40 €
Monitor Terciario	130 €	1	130 €
Silla ergonómica	325 €	1	325 €
Servidor local	200 €	1	200 €
Total hardware	-	-	2,160.88 €

Cuadro 3.3: Coste del hardware

Software	Coste (€)	Unidades	Total (€)
IDEs JetBrains	289 €	1	0 €*
Plugin JPA Buddy	25.99 €	1	0 €*
Plugin BashSupport	14 €	1	0 €*
Plugin CodeMR	124.24 €	1	0 €*
Office 365	100 €	1	0 €*
Total software	-	-	0 €

Cuadro 3.4: Coste del software

* El coste de estos programas se ha reducido a 0 porque se ha utilizado la licencia de estudiante de la Universidad de León para obtenerlos.

3.4. PRESUPUESTO

Teniendo en cuenta todos los recursos mencionados en la sección anterior, se ha realizado un presupuesto del proyecto que se puede ver en la siguiente tabla:

Tipo de coste	Coste
Recursos humanos	69,797.74 €
Recursos hardware	2,160.88 €
Recursos software	0 €
Presupuesto parcial	71,958.62 €
Tarifa internet total	418.8 €
Gastos de gestión	3,000 €
Viajes	1,360 €
Beneficio	7,195.86 €
Parcial	83,932.28 €
IVA (21 %)	17,626.18 €
Total	101,558.46 €

Cuadro 3.5: Presupuesto de RS Chat

3.5. GESTIÓN DE RIESGOS

La gestión de riesgos es un proceso iterativo que comienza en la fase de planificación del proyecto y continúa a lo largo de todo el ciclo de vida del mismo. El objetivo de este proceso es identificar, analizar y responder a los riesgos del proyecto. Para ello, se han seguido los siguientes pasos:

1. Identificación de riesgos: se identifican los riesgos que pueden afectar al proyecto.
2. Análisis de riesgos: se analizan los riesgos identificados para determinar su probabilidad de ocurrencia y su impacto en el proyecto.
3. Planificación de respuestas a los riesgos: se planifican las respuestas a los riesgos identificados.
4. Seguimiento y control de riesgos: se monitorizan los riesgos identificados y se identifican nuevos riesgos.

3.5.1. IDENTIFICACIÓN DE RIESGOS

4. Contenido

En este capítulo se describe el diseño, implementación y arquitectura del sistema y se incluyen diagramas para una mejor comprensión de la misma.

4.1. PATRONES DE DISEÑO

Un patrón de diseño es una solución que se aplica para resolver problemas comunes en el diseño de software [3]. Actúan como plantillas que se pueden personalizar para resolver un problema concreto. Hay varias categorías de patrones de diseño, cada una con una finalidad diferente [4]:

- **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 información que se incluye de los patrones utilizados:

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

4.1.1. BUILDER

Es un patrón de **creación** que permite instanciar objetos complejos de una forma sencilla. Se ha creado una interfaz genérica (**Builder**) 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:

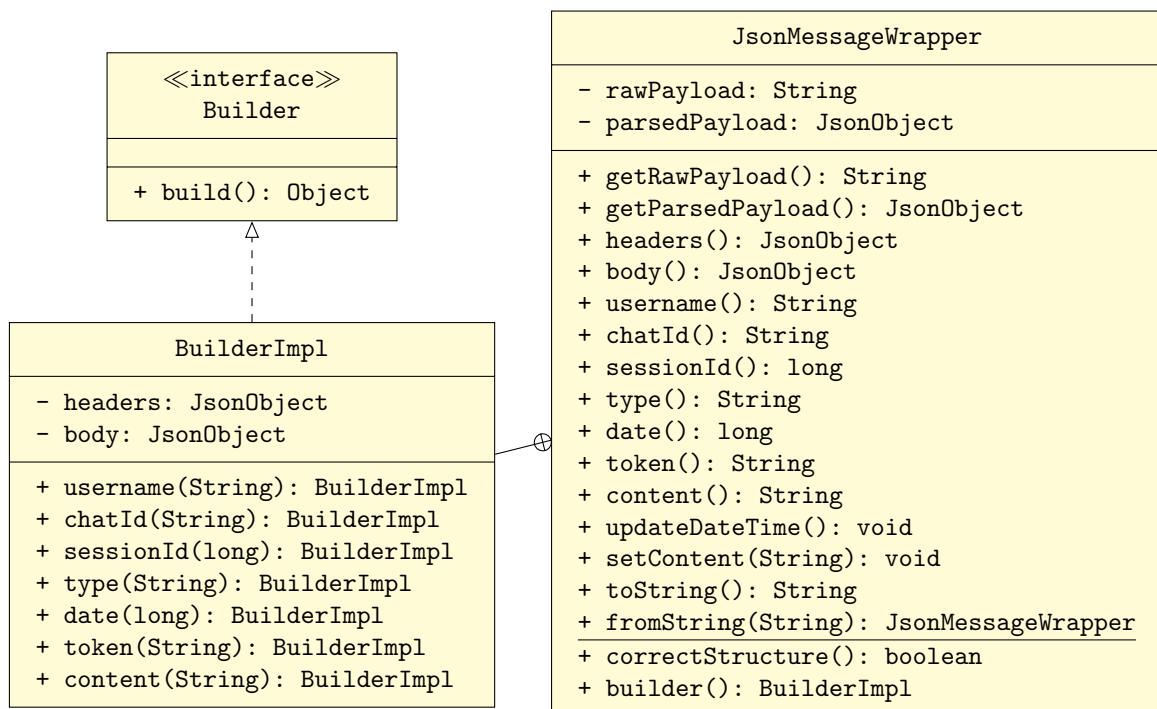


Diagrama UML 4.1: Patrón Builder empleado en la aplicación.

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.

4.1.2. SINGLETON

También es un patrón de **creación**. Se emplea para garantizar que una clase concreta tenga una única instancia y proporciona un punto de acceso global a ella [5]. 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.

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

Cuadro 4.1: Relación entre método HTTP y ruta.

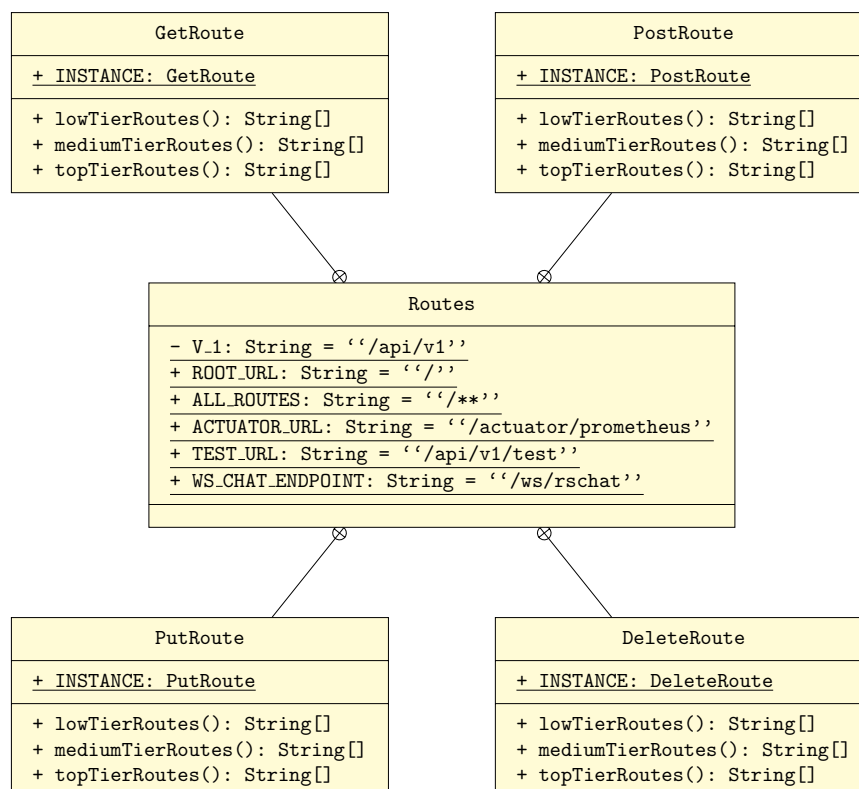


Diagrama UML 4.2: Patrón Singleton empleado en la aplicación.

Se han utilizado diferentes formas de acceso a las instancias de las clases. En el caso de las 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.

4.1.3. 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(MessageHandlingDTO)`, encargado de realizar el procesamiento del mensaje. Este método recibe un parámetro de tipo `MessageHandlingDTO`, que contiene la información necesaria para realizar el procesamiento del mensaje:

- El mensaje que se debe manejar.
- Otros datos que puedan 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 se pueden ver en la tabla 4.2.

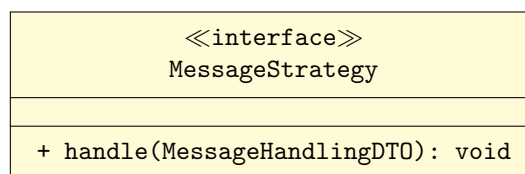


Diagrama UML 4.3: Interfaz `MessageStrategy`.

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 encuentra en la clase `MessageStrategyMappings`. Esta clase es la encargada de determinar qué estrategia se debe utilizar para manejar el mensaje. Para esto, se utiliza el método `decideStrategy(receivedMessageType: Message)` que recibe como parámetro el tipo de mensaje y devuelve la estrategia que se debe utilizar para manejarlo.

Mensaje	Clase de la estrategia
USER_CONNECTED	UserConnectedStrategy
USER_DISCONNECTED	UserDisconnectedStrategy
USER_TYPING	UserTypingStrategy
USER_STOPPED_TYPING	UserStoppedTypingStrategy
USER_JOINED	UserJoinedStrategy
USER_LEFT	UserLeftStrategy
TEXT_MESSAGE	TextMessageStrategy
IMAGE_MESSAGE	ImageMessageStrategy
AUDIO_MESSAGE	AudioMessageStrategy
VIDEO_MESSAGE	VideoMessageStrategy
PDF_MESSAGE	PdfMessageStrategy
TEXT_DOC_MESSAGE	TextDocMessageStrategy
PARSEABLE_MESSAGE	ParseableMessageStrategy
ACTIVE_USERS_MESSAGE	ActiveUsersStrategy
GET_HISTORY_MESSAGE	GetHistoryStrategy
INFO_MESSAGE	InfoMessageStrategy
PING_MESSAGE	PingStrategy
ERROR_MESSAGE	ErrorMessageStrategy
RESTART_MESSAGE	RestartMessageStrategy
MAINTENANCE_MESSAGE	MaintenanceMessageStrategy

Cuadro 4.2: Relación Mensaje - Estrategia

4.2. 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:

- **UserConnected:** es un mensaje que se envía al servidor cuando un usuario se conecta a la aplicación. Sirve para establecer la conexión WebSocket.

- **UserDisconnected**: es un mensaje que se envía al servidor cuando un usuario se desconecta de la aplicación. Sirve para cerrar la conexión WebSocket.
- **UserTyping y UserStoppedTyping**: no se han implementado todavía, ya que no se han considerado indispensables para la interacción.
- **UserJoined y UserLeft**: se notifica a las personas conectadas el nombre del usuario que se ha unido o ha salido del chat, y se añade o elimina de la lista de usuarios conectados.
- **Text, Image, Audio, Video, PDF, TextDoc**: 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.
- **Parseable**: es un tipo especial, que se utiliza para enviar mensajes que contienen información adicional. Se utiliza para leer parámetros en los comandos.
- **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 consulta del historial de mensajes (almacenado en caché) y un filtrado de los mensajes de actividad del solicitante, ya que no son relevantes. Además, si el usuario lo desea, se pueden leer los mensajes anteriores a esos 65 iniciales, utilizando la paginación, hasta llegar al primer mensaje enviado al chat.
- **Info, Restart y Maintenance**: son un tipo de mensajes que los administradores pueden enviar a todos los chats a la vez. Se utilizan para notificar a los usuarios de algún evento que va a ocurrir en el servidor, un reinicio o un mantenimiento.
- **Ping**: se utiliza exclusivamente para mantener la conexión WebSocket abierta. Se envía un texto corto para no sobrecargar la red, y se recibe el mismo texto como respuesta.

- **Error:** se envía cuando se produce algún error en el servidor, y contiene un mensaje con la descripción del error.

4.3. CICLO DE VIDA DE LA CONEXIÓN DE USUARIOS

Cuando un usuario accede a la aplicación, se inicia una conexión entre el cliente y el servidor a través del protocolo de comunicación bidireccional **WebSocket** [6]. Esta conexión se mantiene abierta mientras el usuario tenga la sesión iniciada en la aplicación, y se cierra cuando el usuario termina la sesión. La secuencia de eventos que ocurren durante la conexión de un usuario al chat es la siguiente:

4.3.1. FRONTEND

1. Se realiza una solicitud de conexión WebSocket al servidor al iniciar sesión.
2. Se recibe la respuesta del servidor, que se acepta si el código de estado HTTP es 101 (código de estado **SWITCHING_PROTOCOLS**) y se crea un temporizador para enviar mensajes de tipo **PING_MESSAGE** cada 30 segundos, manteniendo la conexión activa hasta que el usuario cierre sesión. En caso contrario, no se establece la conexión.
3. Cuando el usuario desea entrar a un chat, se realiza una petición HTTP para comprobar que puede acceder. Esto se realiza para que, en caso de que el usuario introduzca de forma manual la URL de un chat al que no tiene acceso, se le redirija a la página de inicio de la aplicación.
4. Si se confirma que el usuario **puede acceder** al chat:
 - 4.1. Se envía un mensaje de tipo **USER_JOINED** al servidor.
 - 4.2. Se consultan los últimos mensajes del historial del chat con el mensaje de tipo **GET_HISTORY_MESSAGE**.
 - 4.3. Se realiza una petición de la lista de usuarios activos con un mensaje de tipo **ACTIVE_USERS_MESSAGE**.
5. Si el usuario **no puede acceder** al chat, se le redirige a la página principal.

4.3.2. 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’`. En esta clase se redefine el método `handleTextMessage` de la clase `AbstractWebSocketHandler` de Spring Boot, que es el encargado de gestionar los mensajes de texto recibidos por el servidor. El proceso de **conexión** al servidor sigue este flujo de eventos:

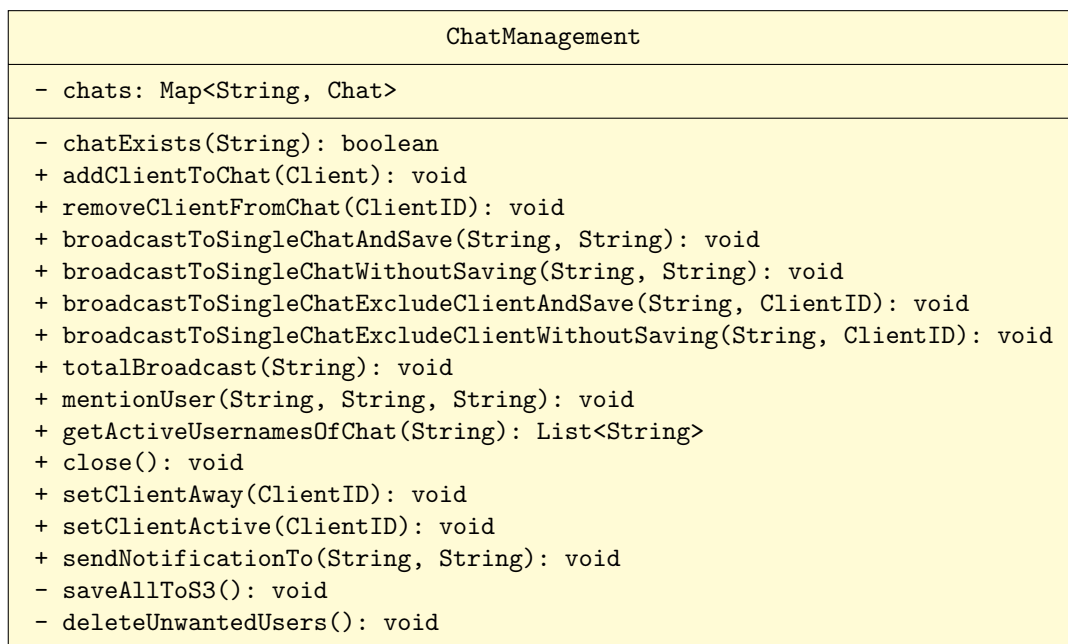
1. Se establece la conexión WebSocket con el usuario. Esto ocurre de forma transparente al programador debido a que la implementación se realiza en el framework de Spring Boot.
2. Se recibe el mensaje `USER_JOINED` del cliente y se crea un objeto de tipo `Client` (formado por la instancia de `WebSocketSession` y el `ClientID` 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`. Antes de que esto ocurra, 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).



Los dos últimos métodos se ejecutan de manera periódica cada 10 y 3 minutos respectivamente.

Diagrama UML 4.4: Clase **ChatManagement** para la gestión de los chats.

4.4. DOCKER

Docker es una plataforma software que permite el desarrollo, despliegue y ejecución de aplicaciones de forma rápida y cómoda, separando la aplicación de la infraestructura, permitiendo ejecutar múltiples aplicaciones en un mismo servidor (ver figura 4.1). 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 [7].

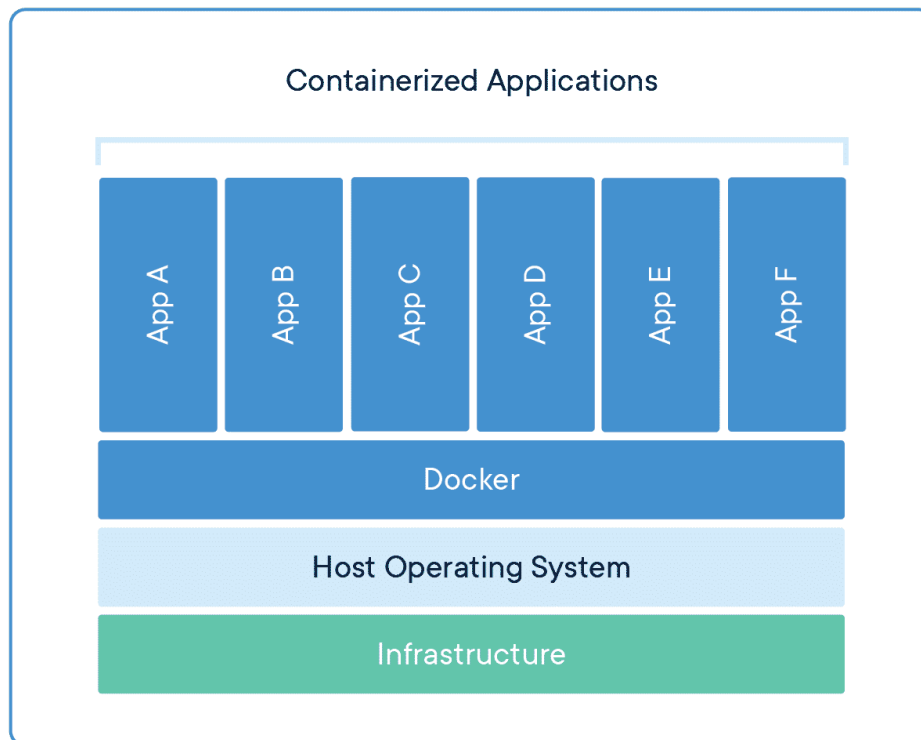


Figura 4.1: Infraestructura docker. Fuente: <https://www.docker.com/resources/what-container>

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.

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 inician [8]. De esta forma, los datos no se pierden al eliminar el contenedor.

Los volúmenes se definen en el mismo fichero `docker-compose.yml` 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 nombres 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.
- `nsfwpy*`: Contenedor que ejecuta el servicio de detección de contenidos inapropiados.

4.4.1. 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 [9]. 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 los siguientes cambios en los ficheros de configuración:

- Añadir 2 librerías en el fichero `pom.xml` [10]:
 - `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 a cada ruta.
- `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.

4.4.2. 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 [11]. 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:

```
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 4.1: 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 [12] 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 [13].



Figura 4.2: Visualización de métricas relacionadas con los recursos del sistema utilizados y tiempo de actividad.

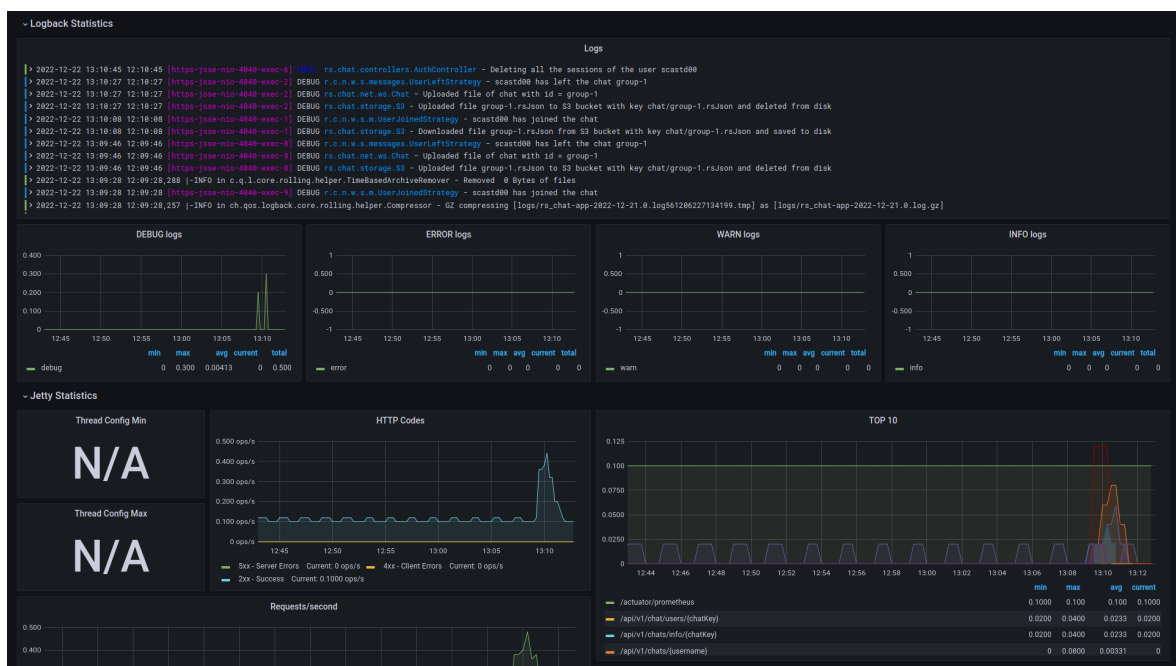


Figura 4.3: Visualización de métricas relacionadas con los logs y peticiones HTTP.

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`

4.4.3. 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 [14]. 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 subsección. 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 [15] y el servicio en el fichero `docker-compose.yaml`:

```
loki:
  image: grafana/loki:2.7.0
  container_name: loki
  restart: unless-stopped
  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 4.2: Servicio de Loki para el registro de logs.

4.4.4. 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 4.4) a través de una API HTTP [14]. 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`:

```

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 4.3: Servicio de Promtail para la recolección de logs.

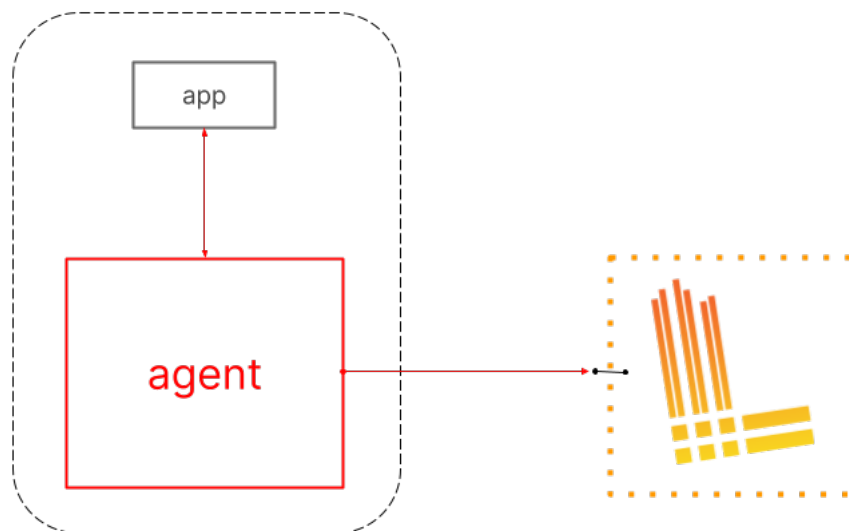


Figura 4.4: Flujo de logs entre Promtail y Loki.

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`, usando 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>(?:[^\s]*[^\s]))
    source: tag
  # ...
```

Código 4.4: Stage para obtener el nombre del contenedor con una expresión regular a partir de la etiqueta `tag`.

4.4.5. NSFWPY

NSFWPY es un detector de contenido no seguro para el trabajo (NSFW, por sus siglas en inglés), que se ha utilizado para detectar imágenes inapropiadas en los mensajes de los usuarios. Es un programa escrito en Python, que consiste en un pequeño servidor HTTP (utilizando Flask) que recibe una imagen en formato `base64` y devuelve un JSON con la predicción de la imagen. Para realizar la predicción, se utiliza el modelo de inteligencia artificial *GantMan/nsfw_model* [1], que está pre-entrenado con imágenes de contenido NSFW y SFW (seguro para el trabajo). El modelo devuelve una predicción de la clase a la que pertenece la imagen, con una probabilidad entre 0 y 1, siendo 0 la probabilidad de que la imagen sea SFW y 1 la probabilidad de que la imagen sea NSFW. El modelo está entrenado para detectar 5 clases de imágenes NSFW:

- Drawings (Dibujos): Imágenes de dibujos animados o hechos a mano.
- Hentai (Manga): Imágenes de manga o anime con contenido sexual.
- Neutral: Imágenes neutrales.
- Porn (Pornografía): Imágenes que tiene contenido pornográfico.
- Sexy (Sensual): Imágenes que tienen un tono sensual.

Para que la aplicación pueda utilizar el detector NSFWPY, se ha creado un nuevo servicio en el fichero `docker-compose.yaml`, que se encarga de ejecutar el programa de Python con el detector, y se ha añadido una nueva ruta en la API de RSChat, que se encarga de llamar al detector NSFWPY.

```
nsfwpy:
  image: nsfwpy:latest
  container_name: nsfwpy
  ports:
    - "4042:4042"
  networks:
    - rschat-net
  healthcheck:
    test: curl -f http://localhost:4042/api/v1/health -s > /dev/null || exit 1
    interval: 2m
    timeout: 10s
    retries: 3
```

Código 4.5: Servicio de NSFWPY para la detección de imágenes NSFW.

Este modelo se ha conocido a través a la página web NSFWJS [16], que lo utiliza para el análisis de las imágenes desde el propio navegador del usuario, sin necesidad de enviarlas a un servidor. La web permite probar el modelo con imágenes que se pueden arrastrar y soltar en la página, y muestra la predicción del modelo para cada una de ellas. Las figuras 4.5 y 4.6 muestran dos ejemplos de predicción del modelo para dos imágenes diferentes.

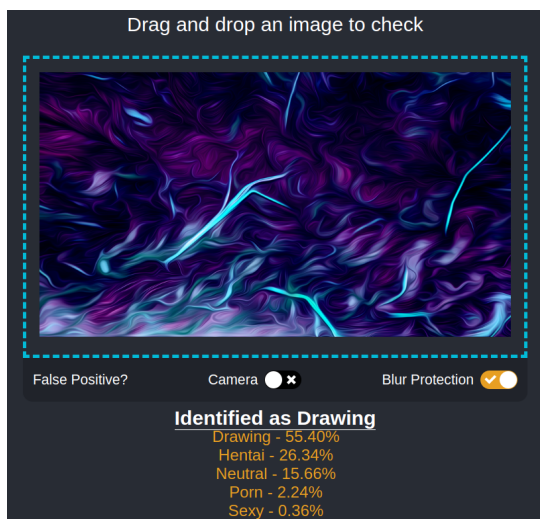


Figura 4.5: Predicción para una imagen de fondo de pantalla.

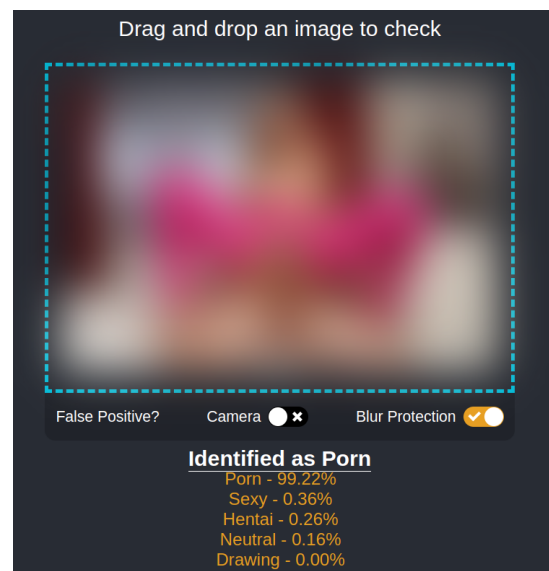


Figura 4.6: Predicción para una imagen pornográfica.

Como se puede observar en las imágenes, la web muestra la predicción del modelo para cada una de las clases como un porcentaje. En el caso de la imagen de la izquierda, se puede ver que el modelo ha predicho que la imagen es un dibujo con un 55.40 % de

probabilidad, mientras que en la imagen de la derecha, el modelo ha predicho que la imagen es pornográfica con un 99.22 % de probabilidad.

4.5. 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:

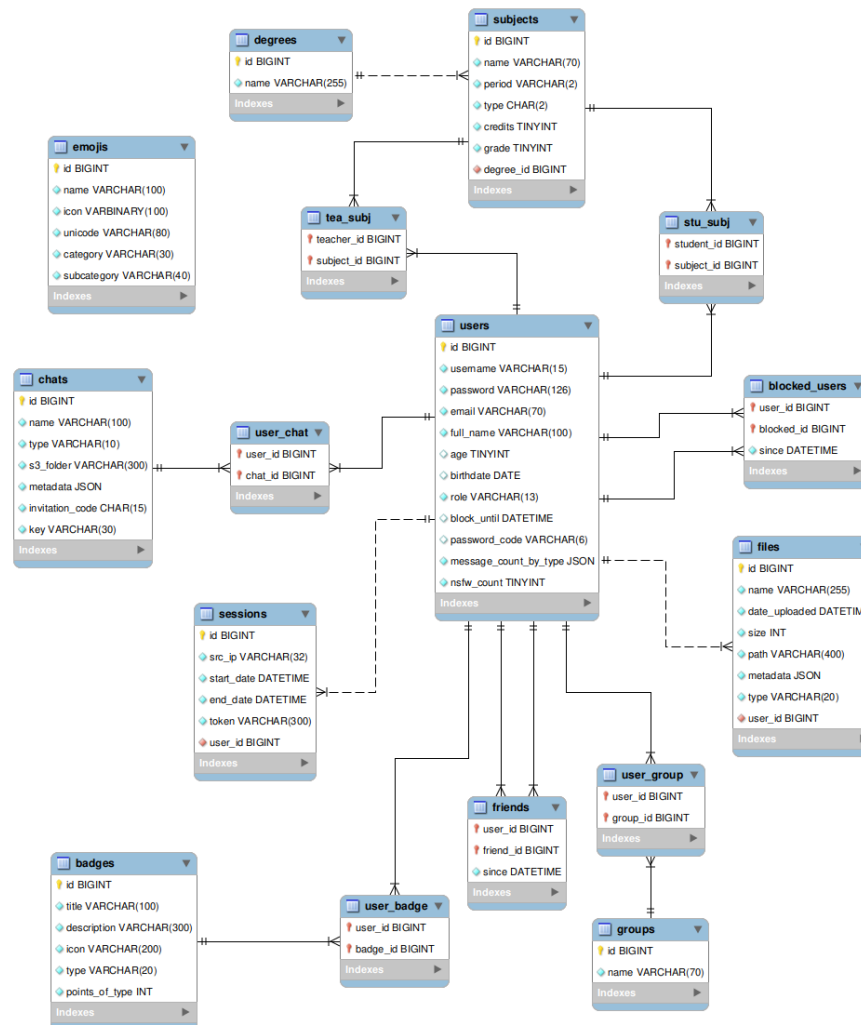


Figura 4.7: Diagrama de las tablas de la base de datos.

4.6. 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 a los siguientes puertos:
 - **22:** puerto de ssh.
 - **80:** puerto de http.
 - **4040:** puerto de la aplicación en entorno de producción.
 - **4041:** puerto de la base de datos de la aplicación.
 - **4042:** puerto del servicio NSFW.
 - **4046:** puerto del panel de administración de Grafana.

Además, se ha configurado el servidor web para que únicamente se pueda acceder mediante el protocolo https, y no mediante 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. Para poder conectarse a la base de datos desde el exterior, se debe utilizar la clave ssh del servidor web, de forma que se accede este y, desde ahí, se puede acceder a la base de datos.

4.7. PRUEBAS

En un proyecto software, las pruebas son una parte fundamental del desarrollo, ya que garantiza que el software funciona correctamente y se comporta como se espera. Además, se asegura que el software no se rompe al añadir nuevas funcionalidades o

al modificar las existentes. Existen varios tipos de pruebas, dependiendo del nivel de abstracción al que se realicen. En este proyecto se han implementado pruebas unitarias y de integración.

4.7.1. PRUEBAS UNITARIAS

Las pruebas unitarias son aquellas que se realizan sobre la unidad más pequeña de código, que en el caso de este proyecto son los métodos. Se dividen en dos tipos:

4.7.1.1. Pruebas de caja blanca

Las pruebas unitarias de caja blanca son un método de diseño de casos de prueba que se basa en la estructura interna del código a probar. Se centra en la cobertura de las rutas de ejecución del código, es decir, se comprueba que se ejecuta cada línea de código al menos una vez. Para ello, se realizan diagramas de control flujo de los métodos y se analiza la **complejidad ciclomática**, que es una medida de la complejidad de un programa de ordenador. Este tipo de grafo es dirigido, en el que los nodos representan grupos de instrucciones y las aristas las posibles rutas de ejecución. La complejidad ciclomática se puede calcular mediante 3 fórmulas diferentes, que son equivalentes:

- $C = R$
- $C = E - N + 2$
- $C = D + 1$
- Donde:
- C es la complejidad ciclomática.
- R es el número de regiones.
- E es el número de aristas.
- N es el número de nodos.
- D es el número de sentencias de decisión.

Cuanto mayor sea la complejidad, más difícil será probar el código y es más probable que contenga errores. Por ello, se recomienda que la complejidad ciclomática sea menor

o igual que 10, para que sea más fácil de probar [17]. A continuación, se muestran las pruebas unitarias de caja blanca realizadas para un método que tiene sentencias iterativas y condicionales:

```
public double probabilityOfClass(NSFWClass nsfwClass) {
    double probability = 0.0;

    for (NSFWClass value : NSFWClass.values()) {
        String predictionName = value.name().toLowerCase();
        probability = this.predictions.get(predictionName).getAsDouble();

        if (value.equals(nsfwClass)) {
            break;
        }
    }

    return probability;
}
```

Código 4.6: Método para determinar la clase NSFW asociada a una imagen

El análisis de la complejidad ciclomática quedaría de la siguiente forma:

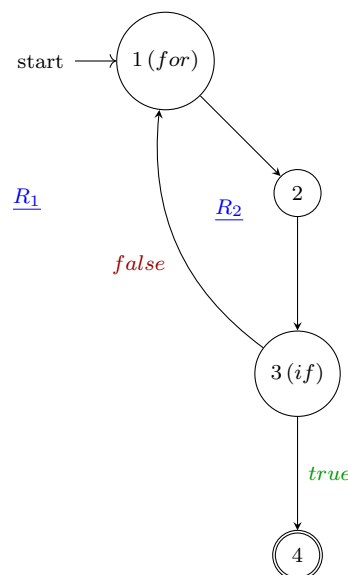


Figura 4.8: Grafo de control de flujo del método `probabilityOfClass`

Los resultados para cada fórmula son los siguientes:

- $C = R = 2$
- $C = E - N + 2 = 4 - 4 + 2 = 2$

- $C = D + 1 = 1 + 1 = 2$

Como se puede observar, el método tiene una complejidad ciclomática de 2, por lo que es fácil de probar. Ahora se procede a realizar los casos de prueba para que cada línea de código se ejecute al menos una vez:

1. **Camino 1:** $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$.

Aristas cubiertas: $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$.

Condición: `nsfwClass.name = value.name`.

Caso de prueba: `nsfwClass = NSFWClass.NEUTRAL`.

Descripción: De esta manera, el valor del parámetro del método es igual al primer valor del enumerado, por lo que se ejecuta la rama `true` del condicional y se sale del bucle en la primera iteración.

2. **Camino 2:** $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$.

Nueva arista cubierta: $3 \rightarrow 1$.

Condición: `nsfwClass.name \neq value.name`.

Caso de prueba: `nsfwClass = NSFWClass.DRAWINGS`.

Descripción: De esta manera, el valor del parámetro del método es distinto al primer valor del enumerado, por lo que se ejecuta la rama `false` del condicional y se vuelve a ejecutar el bucle.

4.7.1.2. Pruebas de caja negra

Las pruebas de caja negra son aquellas que se enfocan en la funcionalidad del programa, sin tener en cuenta su implementación interna. Para ello, se diseñan casos de prueba que cubran todas las posibles **entradas y salidas** del programa. La realización de estas pruebas conlleva la creación de un conjunto de datos de prueba, que se organizan en **clases de equivalencia** (o particiones de equivalencia) y **condiciones límite**. En este caso, se han diseñado los casos de prueba usando la técnica de particiones de equivalencia, que consiste en dividir el conjunto de datos de entrada en subconjuntos de datos de entrada válidos e inválidos, de tal forma que se garantice que todos los casos de prueba se validan una vez. A continuación, se muestran las clases de equivalencia definidas para la funcionalidad de consultar los emojis en función de su nombre:

Sec.	Condición de entrada	Tipo	Clases válidas		Clases No Válidas	
			Entrada	Código	Entrada	Código
1	Nombre de emoji	Lógico	Cadena vacía	CEV<1>		
		Si está, es Valor	Cadena con caracteres válidos	CEV<2>	Cadena con caracteres válidos pero no existe en base de datos	CENV<1>
					Cadena con caracteres no soportados (números, /, ?, !, etc.)	CENV<2>

Cuadro 4.3: Definición de las clases de equivalencia.

ID CP	Escenario	Condiciones de entrada	Resultado esperado
		Nombre de emoji	
CP1	Escenario 1	Válida	Lista vacía
CP2	Escenario 1	Válida	Lista con emojis cuyo nombre contiene la cadena
CP3	Escenario 2	No Válida	Lista vacía
CP4	Escenario 3	No Válida	Respuesta HTTP 404 - No emojis found

Cuadro 4.4: Casos de prueba para la funcionalidad de consultar emojis por nombre.

ID CP	Clases de Equivalencia	Condiciones de entrada	Resultado esperado
		Nombre de emoji	
CP1	CEV<1>	""	[]
CP2	CEV<2>	"grinning"	[😄 , 😄 , 😄 , 😄 , 😄 , 😄 , 😄]
CP3	CENV<1>	"emoji_grin"	404 - No emojis found
CP4	CENV<2>	"grin2?"	404 - No emojis found

Cuadro 4.5: Valores para los casos de prueba.

Código	Descripción
CP	Caso de prueba
CEV<N>	Clase de equivalencia válida n ^o N
CENV<N>	Clase de equivalencia no válida n ^o N

Cuadro 4.6: Nomenclatura utilizada en las tablas de casos de prueba.

4.8. 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.

4.8.1. 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 de despliegue de 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 de datos o desplegar otras aplicaciones más potentes que con la Raspberry Pi. 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.

4.9. 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 de una aplicación web.

4.9.1. 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 ha realizado la instalación de 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 al servidor 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.
- **zsh**: es un intérprete de comandos que permite personalizar la terminal [18].
- **oh-my-zsh**: es un framework para personalizar la terminal utilizando colores, temas, etc [19].

4.9.2. 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 router, pero no se ha podido conseguir, ya que utilizan la tecnología CG-NAT. Esto no supone 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 su página web [20]. 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.

4.9.3. 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.
- Crea la imagen de Docker, instalando las dependencias necesarias y compilando el código fuente.
- Crea los contenedores Docker especificados en el fichero `docker-compose.yaml` y los ejecuta en segundo plano.

4.9.4. 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.

4.9.5. 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.

4.9.6. 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.

5. Anexos

5.1. ANEXO A: PILA DEL PRODUCTO

Durante 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 tareas, con la posibilidad de añadir etiquetas, asignar responsables, añadir comentarios a cada una de ellas, entre otras funcionalidades. A continuación, se incluye el fichero pdf generado por Notion con la pila del producto completa. Además, se puede consultar la pila del producto de una manera más interactiva en el siguiente enlace: <https://scastd00.notion.site/9c59c2812d8b440faa453924fef2f320?v=20f7c2f47dd64574ac57713e8efe2718>



RS Chat

- 🏔 Epics are large overarching initiatives.
- 📅 Sprints are time-bounded pushes to complete a set of tasks.
- 🔧 Tasks are the actions that make up epics.
- 🐛 Bugs are tasks to fix things.

A3 Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Add Report page for admins and display all the report notifications	Low	Improvement 🚀	Not Started					
Remove the invite command	Medium	Refactor 🔧	Complete	Sprint 11	@ May 1, 2023 13:20	@ May 2, 2023 13:40	0	Added the invite button in chat view
Fix bug not sending email on invitation	High	Bug 🐛	Complete	Sprint 11	@ May 1, 2023 1:40	@ May 1, 2023 1:57	0	Instead of using the email, it was used the username of the invetee. Used Tuple to return the chat and the user from the lambda.
Refactor Utils class to not be named like that	Low	Documentation 📖 Refactor 🔧	Complete	Sprint 11	@ April 30, 2023 3:25	@ April 30, 2023 3:58	0	
Video handling with lib	UNSET	Idea 💡	Not Started					https://github.com/kokorin/Jaffree https://github.com/Manevolent/fmmpegd
Fix Redux localstorage modification by checking the username of the token in the backend.	Very high	Bug 🐛	Not Started					Also the storage should be changed to other with more security
Fix error in profile view	Medium	Bug 🐛	Revision	Sprint 11				
When inviting someone to a chat, send an email to the user to notify the invitation. The user is added to the chat immediately	Medium	Improvement 🚀	Complete	Sprint 11	@ April 11, 2023 2:05	@ April 12, 2023 3:30	12	Instead of a command, do a button near chat name and send a <code>POST</code> request. This allows to ease the process of DI and send the email to the user.
New design when sending nsw media	Medium	Improvement 🚀	Not Started					The user can see the image, but other users cannot
Send the session in in the login and register response to identify the session in the profile view	Low	Improvement 🚀	Not Started					
Improve session display in frontend (show dates, and the index is the id of the session)	Medium	Improvement 🚀	Complete	Sprint 10	@ April 10, 2023 14:52	@ April 10, 2023 17:03	0	
Do not store empty commands	Medium	Bug 🐛	Not Started					
When showing command stats in profile, the app crashes (see image)	Critical 🔥	Bug 🐛	Complete	Sprint 10	@ April 10, 2023 12:35	@ April 10, 2023 14:52	0.7	An object (command stats) is stored inside, and the app does not support that.
Fix app crash when sending a command that is unavailable	Critical 🔥	Bug 🐛	Complete	Sprint 10	@ April 10, 2023 3:08	@ April 10, 2023 4:34	0	Added NoOp command
Improve command parsing	Medium	Improvement 🚀 Refactor 🔧	Not Started					
Improve blocked users and friends functionality and relationships in db	Low	Refactor 🔧	Not Started					
Chats with limited number of members	To define	Idea 💡 Improvement 🚀	Not Started					
Use Optional<T> in all return types from the repository	High	Refactor 🔧	Complete	Sprint 11	@ May 3, 2023 12:39	@ May 3, 2023 12:44	0	
Refactor exception handling in controllers with @ControllerAdvice	Medium	Refactor 🔧	In Progress	Sprint 11	@ May 7, 2023 2:31			https://reflectoring.io/spring-boot-exception-handling/
Create group by every user (private or public)	Low	Improvement 🚀	Not Started					
Unread messages balloon notification	Medium	Improvement 🚀	Not Started					
Multiple chat tabs	Medium	Improvement 🚀	Not Started					
Webcam integration	To define	Idea 💡 Improvement 🚀	Not Started					https://primefaces.github.io/primefaces/srdocs/modules/src_PrimeFaces.Webcam.html
Configure the CORS policy when checking connectivity to the bucket	High	Improvement 🚀	Complete	Sprint 10	@ April 7, 2023 16:42	@ April 7, 2023 22:03	1.5	
Fix pdf not showing	Very high	Bug 🐛	Complete	Sprint 10	@ April 7, 2023 15:25	@ April 7, 2023 16:42	0	CORS policy has incorrect <code>AllowedOrigins</code> prop
Make a form to get feedback from users	Very high	Research 🏔	Complete	Sprint 10	@ April 6, 2023 16:00	@ April 8, 2023 16:00	18	
Check run conditions when uploading several images at once when performing the request to the Python API	Very high	Bug 🐛	Complete	Sprint 10	@ April 5, 2023 3:43	@ April 5, 2023 4:06	0	When uploading 5 nsw images, in the DB is stored only 4 (in the user nsw_count column). https://stackoverflow.com/questions/30346882/do-synchronized-java-methods-queue-calls
If user is blocked, do not allow to enter the app	High	Improvement 🚀	Complete	Sprint 10	@ April 5, 2023 4:10	@ April 5, 2023 4:20	0	Very easy. Added a single check in the <code>login</code> endpoint and a pretty format
When sending a server message, the type must be a generic one.	High	Refactor 🔧	Complete	Sprint 10	@ April 5, 2023 13:00	@ April 5, 2023 13:12		Create a constant and update all usages of the <code>createMessage</code> function
Add more CI steps for testing	Medium	CI/CD 🛠 Improvement 🚀 Test 🧪	Complete	Sprint 10	@ March 30, 2023 13:56	@ March 30, 2023 15:15	0.5	





































Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Prepare environment to perform frontend tests	High	Configuration Test	Complete	Sprint 10	@ March 30, 2023 1:00	@ March 30, 2023 1:10	0	Simple jest execution
Age verification	Low	Improvement	Not Started					
Kick the users out of the session when they upload the last NSFW content	High	Improvement	Complete	Sprint 10	@ April 5, 2023 2:03	@ April 5, 2023 3:29	0	
Set to users the block date when several NSFW content is uploaded to chats	High	Improvement	Complete	Sprint 10	@ March 21, 2023 23:00	@ March 21, 2023 23:56	0	
Fix not showing the images when uploading	Very high	Bug	Complete	Sprint 10	@ April 6, 2023 15:57	@ April 6, 2023 16:20	0	Resources are uploaded to the wrong folder in S3 (uppercase) and should be stored inside the lowercase ones
When users upload several NSFW images, block them for a period of time	Medium	Improvement Task	Reference	Sprint 10			0	Set to users the block date when several NSFW content is uploaded to chats
Migrate nstsv service to Python	Critical	Bug Refactor	Complete	Sprint 10	@ March 12, 2023 13:57	@ March 14, 2023 2:49	24	https://pythonspeed.com/articles/activate-virtualenv-dockerfile/ https://stackoverflow.com/questions/30215830/dockerfile-copy-keep-subdirectory-structure Needed to set the Internet Address to 0.0.0.0 to be able to receive external requests
Improve friends functionality to not add symmetric rows to DB	Medium	Improvement Refactor	Not Started					To represent friends, with only one relation suffices (or look for other techniques).
Report user form and send a notification to administrators of a new report	Medium	Improvement	Not Started					
Dockerize rs-chat-utils services	Very high	Configuration Improvement	Complete	Sprint 10	@ March 11, 2023 17:45	@ March 14, 2023 3:14	36	https://github.com/tensorflow/tfjs/issues/1425#issuecomment-596197404 I have to build tensorflow from source or downgrade to 1.5. After compiling from source (6.7 h) found version https://github.com/yaroslavvb/tensorflow-community-wheels/issues/217
Send requests from the FileController to the utility NSFW service	High	Improvement Security	Complete	Sprint 09	@ March 11, 2023 2:35	@ March 11, 2023 5:34	0	Store the file before the request and send the path to the NSFW service, remove <code>multer</code> lib from the service, since it is not used. For making the requests <code>testTemplate</code> is useful.
When the chat passes 1 month, delete the history	Medium	Idea Improvement	Not Started					
Integrate the Keras NSFW model to Java (new service or integrated in the app)	Low	Idea Improvement	Cancelled			@ March 11, 2023 2:30		https://towardsdatascience.com/deploying-keras-deep-learning-models-with-java-62d80464f34a
Single IntelliJ project with all modules inside	High	Configuration	Complete	Sprint 09	@ February 25, 2023 0:29	@ February 25, 2023 0:42	0	Import as modules and that's it
For image detection use nstsv's backend lib to use the IA model	High	Improvement	Complete	Sprint 09	@ February 25, 2023 1:40	@ February 25, 2023 23:26	12	https://www.ngmjs.com/package/nstsvs#node-js-app https://github.com/tensorflow/tfjs/issues/7273#issuecomment-1384413639
Use magic-8-ball api for that chat command	Medium	Improvement Task	Complete	Sprint 09	@ February 25, 2023 1:40	@ February 25, 2023 19:48	10	https://www.eightballapi.com/docs
Add a 'Copy code' in reset password email template	Medium	Task	Complete	Sprint 09	@ February 23, 2023 2:54	@ February 23, 2023 3:20	0	
Fix email sender	Critical	Bug	Complete	Sprint 09	@ February 23, 2023 2:00	@ February 23, 2023 2:54	0	Use the package <code>spring-boot-starter-mail</code> instead of <code>javax.mail</code> https://mkyong.com/spring-boot/spring-boot-how-to-send-email-via-smtp/
Update policies messages	High	Bug	Complete	Sprint 09	@ February 22, 2023 0:23	@ February 22, 2023 1:38	0	<code>message()</code> call overrides the default message assigned to the constraint
Make the MailSender a component	Medium	Configuration Refactor	Cancelled	Sprint 09	@ February 21, 2023 23:30	@ February 21, 2023 23:50	0	
Integration tests	Very high	Improvement Test	In Progress	Sprint 09	@ February 20, 2023 17:05			https://reflectoring.io/spring-boot-test/ Disable some beans to avoid conflicts in tests
Refactor of HttpResponse, back to previous implementation	Critical	Bug Improvement Refactor Test	Complete	Sprint 09	@ February 18, 2023 17:16	@ February 18, 2023 18:51	0.5	
New implementation for HttpSenderResponse to be able to test the controllers	Medium	Refactor Test	Complete	Sprint 09	@ February 16, 2023 1:38	@ February 17, 2023 20:13	18	
Controller tests with MockMvc	Medium	Improvement Test	In Progress	Sprint 09	@ February 15, 2023 0:55		13	See description for more details
Use WebTau for E2E tests (for test the API)	High	Improvement Test	Cancelled			@ February 21, 2023 23:30		https://github.com/testingisdocumenting/webtau
Remove from shouldNotSomething tests the given save to the repository	High	Refactor Test	Complete	Sprint 09	@ February 14, 2023 1:31	@ February 14, 2023 2:08	0	Also improved existing tests and added more
Script to update secrets in GitHub for tests	High	Improvement Security Test	Complete	Sprint 09	@ February 14, 2023 14:08	@ February 14, 2023 20:04	4	
Fix problems when sending a media message after uploading it	Very high	Bug	Complete	Sprint 08	@ February 10, 2023 2:00	@ February 10, 2023 2:30	0	
DTO for upload strategies	Low	Refactor	Complete	Sprint 08	@ February 10, 2023 2:48	@ February 10, 2023 3:01	0	
Count the commands used and save it in the user's column in db	Low	Improvement	Complete	Sprint 08	@ February 9, 2023 21:54	@ February 9, 2023 22:14	0	
Move command instantiation to its class as static members	Medium	Improvement Refactor	Complete	Sprint 08	@ February 9, 2023 21:18	@ February 9, 2023 21:53	0	

Aa Projects	■ Priority	≡ Type	⚙ Status	🕒 Sprint	📅 Started	📅 Finished	# Inactivity (h)	≡ Notes
Other users should not receive the command message if it is for own use. Mentions are sent to all users	Medium	Improvement 🚀	Complete	Sprint 08	@February 9, 2023 11:55	@February 9, 2023 14:31	0.5	
New DTO for handling commands	Medium	Refactor 🔧	Complete	Sprint 08	@February 9, 2023 2:00	@February 9, 2023 2:15	0	
Revise the JSON parse method calls in frontend to check if they must be removed since the JSON is sent correctly now	High	Task 🏹	Complete	Sprint 08	@February 8, 2023 12:32	@February 8, 2023 17:21	1	
Fix serialization problems of JsonObject when sending responses	Very high	Improvement 🚀 Refactor 🔧	Complete	Sprint 08	@February 8, 2023 1:05	@February 8, 2023 2:50	0.5	New Serializers for ObjectMapper
Badge view in profile or new page	Medium	Improvement 🚀	Complete	Sprint 08	@February 7, 2023 18:30	@February 8, 2023 20:53	0	Also added stats
Create the design of the badges	Medium	Task 🏹	Complete	Sprint 08	@February 7, 2023 11:50	@February 7, 2023 16:48	2	https://badge.design/ https://pictogrammers.com/library/mdl/ https://vectorink.io/app/
When reconnecting to a chat, the history is not retrieved	Very high	Bug 🐛	Complete	Sprint 08	@February 8, 2023 19:30	@February 8, 2023 19:48	0	Detected in websocket readyState
Big refactor to inject some objects to message strategies	High	Improvement 🚀 Refactor 🔧	Complete	Sprint 08	@February 1, 2023 19:38	@February 1, 2023 21:06	0	
Make all strategies a @Component, to be able to inject attributes	Low	Improvement 🚀 Refactor 🔧	Cancelled					See Big refactor to inject some objects to message strategies . Strategies do not have the annotation but all the needed arguments are passed to the constructor of the strategy from the Mappings class (this is a @Component class)
Bufix: uri is not a property in messages, it was changed to path	Critical 🔥	Bug 🐛	Complete	Sprint 08	@January 31, 2023 0:20	@January 31, 2023 0:25	0	
Mappings for Upload strategies	High	Improvement 🚀	Complete	Sprint 08	@January 30, 2023 20:34	@January 30, 2023 21:05	0	Instead of using Map, an ArrayList of new value object is used
Bufix: history file is deleted from disk after 10 minutes (scheduled task) but should not be deleted unless there is no users in the chat	Very high	Bug 🐛	Complete	Sprint 08	@January 30, 2023 23:45	@January 30, 2023 23:53	0	
Replace single MIME type for files with the full MIME type name	Medium	Refactor 🔧	Cancelled			@January 31, 2023 23:30	-1	This change would affect the storage of the files in S3 bucket
Image recognition with deeplearning4j library	Medium	Improvement 🚀	Cancelled			@February 25, 2023 0:13		https://medium.com/datactw/image-classification-neural-network-tutorial-getting-started-with-deep-learning-for-java-d14-b36372aad656 https://www.baeldung.com/java-cnn-deeplearning4j
Delegate functionality to Chat class instead of ChatManagement	High	Improvement 🚀 Refactor 🔧	Complete	Sprint 08	@January 30, 2023 2:10	@January 30, 2023 4:24	0	Also improved code readability of both classes
Possibility to add a proxy between the client and the server	UNSET	Idea 💡 Improvement 🚀 Security 🛡	Not Started					
Add JWTService class and change library to jst	Low	Refactor 🔧	Complete	Sprint 08	@January 26, 2023 22:06	@January 27, 2023 1:58	2.4	
Go down button with small number icon to specify the received messages	Medium	Improvement 🚀	Not Started					
🚀 10k lines of code 🚀	Critical 🔥		Complete			@January 26, 2023 3:35		
Close all sessions does not work well (after doing it, I continued to the chat page in other tab)	High	Bug 🐛	Complete	Sprint 08	@January 31, 2023 13:20	@January 31, 2023 14:10	0	
Administration button to close/open chats and send to home the connected users	Low	Improvement 🚀	Cancelled			@January 31, 2023 23:06		
File caching to not read files from disk all times	Very high	Improvement 🚀	Complete	Sprint 08	@January 25, 2023 15:54	@January 26, 2023 0:32	3.5	
Spring migration from 2.7.0 to 3.0.2	High	Improvement 🚀 Refactor 🔧 Security 🛡 Task 🏹	Complete	Sprint 08	@January 26, 2023 16:45	@January 26, 2023 22:00	1.6	
Huge refactor to use JPA Buddy when generating entities and relationships	Critical 🔥	Refactor 🔧	Complete	Sprint 08	@January 22, 2023 17:34	@January 24, 2023 21:46	24	https://www.youtube.com/watch?v=DC6FC4olhE
Add teachers to degree by administrator	Very high	Improvement 🚀 Task 🏹	Complete	Sprint 08	@January 22, 2023 16:19	@January 22, 2023 17:15	0	Teachers are added implicitly to the degrees when they are added to a subject. Modify admin dashboard to show the degree of each subject
Special rooms for games	Very low	Idea 💡 Improvement 🚀	Not Started					TicTacToe, Bingo...
Refactor of if statements in Policies class to use Yavi validation library	Very high	Improvement 🚀 Refactor 🔧	Complete	Sprint 08	@January 20, 2023 12:35	@January 20, 2023 23:42	3	
Add Quartz lib to schedule all the tasks	Medium	Improvement 🚀	Cancelled	Sprint 08	@January 15, 2023 13:50	@January 15, 2023 14:00	0	
Improve the security of my home Wi-Fi and network	High	Research 🔬 Security 🛡	Not Started					
Implement rate limit to avoid spamming a lot of messages in short time	High	Improvement 🚀	Complete	Sprint 08	@February 1, 2023 0:40	@February 1, 2023 15:24	8	Messages that are sent after the notification of TOO_FAST_MESSAGE are deleted from the user chat view
Add relations to entities to reduce queries	Low	Refactor 🔧	Reference					See Huge refactor to use JPA Buddy when generating entities and relationships

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Fix /dice command. Page crashes	Very high	Bug 🐛	Complete	Sprint 07	@ December 21, 2022 0:20	@ December 21, 2022 0:50	0	
Add teacher to subject by administrator	High	Improvement 🚀 Task 🏃	Complete	Sprint 07	@ December 18, 2022 14:52	@ December 21, 2022 17:44	28	2 columns in UI: Left with teacher names and id, Right with subject names. With a search bar for each column
Teacher service to get data	High	Task 🏃	Complete	Sprint 07	@ December 18, 2022 2:08	@ December 18, 2022 4:00	0	
When sending a non-text message, the Handler crashes because the content could not be read as normal text.	Very high	Bug 🐛	Complete	Sprint 07	@ December 17, 2022 20:35	@ December 17, 2022 22:57	1	Check the type before the content
Know how to add custom stats to send to Prometheus	Medium	Configuration ⚙️ Improvement 🚀	Complete	Sprint 07	@ December 17, 2022 2:49	@ December 17, 2022 3:20	0	https://fabianlee.org/2022/06/29/java-adding-custom-metrics-to-spring-boot-micrometer-prometheus-endpoint/
Mark html containing the mentioned user	Low	Improvement 🚀	Complete	Sprint 07	@ December 16, 2022 20:46	@ December 17, 2022 0:13	1	
Send mention message to the mentioned user to notify in frontend	Medium	Improvement 🚀	Complete	Sprint 07	@ December 16, 2022 1:40	@ December 16, 2022 20:07	12	
Add support for multi-parameter commands and multiple commands in one message	High	Bug 🐛 Improvement 🚀	Complete	Sprint 07	@ December 11, 2022 0:00	@ December 13, 2022 0:36	24	Implement a parser in backend to correctly send messages. It must support multiple parameters for a command and several commands in one message
Add new commands for fun (roll a dice, @ ball, ...)	Low	Improvement 🚀	Complete	Sprint 07	@ December 10, 2022 3:40	@ December 16, 2022 3:00	-1	
Make a table to display command help in frontend	Low	Improvement 🚀	Not Started					
Refactor commands in backend	Medium	Refactor 🔄	Complete	Sprint 06	@ December 10, 2022 2:00	@ December 10, 2022 3:00	0	New record with several fields and removed from the CommandStrategy interface
Send logs to Grafana with Loki	High	Improvement 🚀	Complete	Sprint 06	@ December 5, 2022 20:00	@ December 7, 2022 1:00	16	Improve by making a custom archive to send the logs to and read from that
Teacher routes in frontend	Medium	Improvement 🚀	In Progress	Sprint 06	@ November 20, 2022 12:37			
Fix Force Browsing the images in S3 bucket	Very high	Bug 🐛 Improvement 🚀 Security 🛡️	Not Started					See Change IDOR (Insecure Direct Object Reference) when showing the chat URL
When user has friends and they are connected to the same chat, the first active users shown must be that friends	Low	Improvement 🚀	Not Started					
View profile functionality. Username is used in URL	Medium	Improvement 🚀	In Progress	Sprint 06	@ November 20, 2022 2:06			
Improve going back and forward with mouse buttons	Medium	Improvement 🚀	Not Started					Store the location where the user joined the page and if there are no more entries to go back, go to that location.
Do not refresh page when changing chat. Clear history, info and users	High	Improvement 🚀 Task 🏃	Cancelled			@ November 20, 2022 0:20		
Change IDOR (Insecure Direct Object Reference) when showing the chat URL	High	Bug 🐛 Improvement 🚀 Security 🛡️	Complete	Sprint 06	@ November 18, 2022 20:54	@ November 20, 2022 0:22	8	Check this video for explanation: https://www.youtube.com/watch?v=rlopMGcPMkI IDOR - fix with new reducer in frontend (history of all pages visited)
Implement individual chats	Very high	Improvement 🚀	Complete	Sprint 06	@ November 17, 2022 16:35	@ November 18, 2022 15:41	3	Now do this → Do not refresh page when changing chat. Clear history, info and users
Backend logging division by level	Very high	Configuration ⚙️ Improvement 🚀	Complete	Sprint 06	@ November 16, 2022 23:30	@ November 17, 2022 0:05	0	
Check response bodies in frontend	Very high	Refactor 🔄 Task 🏃	Complete	Sprint 06	@ November 16, 2022 20:00	@ November 16, 2022 20:06	0	
Refactor all the controllers. When an exception is thrown in other method, catch it to send a response message to show to the user	Very high	Refactor 🔄 Task 🏃	Complete	Sprint 06	@ November 15, 2022 23:28	@ November 16, 2022 19:24	14	Change the error message retrieval in frontend. Can be rolled back to not using this, since the exceptions are sent if thrown in other method, but the message retrieval changes a bit.
Web Socket does not connect to backend in production	Critical 🔥	Bug 🐛 Task 🏃	Complete	Sprint 06	@ November 11, 2022 18:30	@ November 16, 2022 1:28	-1	Brave Browser was blocking the connection 🤔
Application statistics. Maybe other application using SOS locally?	Medium	Improvement 🚀	Reference					Ref: Chat statistics - Per user, number of messages, words written ...
Implement commands and mentions	High	Improvement 🚀	In Progress	Sprint 05	@ November 6, 2022 2:34			
Support mentions with '@' character	Medium	Improvement 🚀	Complete	Sprint 05	@ November 5, 2022 3:36	@ November 5, 2022 15:33	9	Use the same tokenizer for commands but for @ symbol
Revise all todo tasks	Very high	Research 🔍 Task 🏃	Complete	Sprint 05	@ November 4, 2022 20:27	@ November 5, 2022 0:58	1.5	First frontend, then backend
Extract connection display to new component	Medium	Refactor 🔄	Complete	Sprint 05	@ November 4, 2022 13:39	@ November 4, 2022 14:00	0	
New component colors for the information messages	High	Improvement 🚀	Complete	Sprint 05	@ November 4, 2022 14:21	@ November 4, 2022 14:23	0	
Reconnect button in ToolBar	Medium	Improvement 🚀	Not Started					Might not be needed
An 'ActiveUsers' message is sent after the client closes a chat	High	Bug 🐛	Ignore					It is a message to get all the active users when one leaves the chat. No problem

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
State information about the connection of the socket to the server (Connected or disconnected) in Toolbar	Medium	Improvement 🚀	Complete	Sprint 05	@ November 2, 2022 17:03	@ November 2, 2022 17:11	0	
Global variable to store the WebSocket client when the user is logged in	High	Improvement 🚀	Complete	Sprint 05	@ November 2, 2022 16:41	@ November 2, 2022 17:00	0	Kept global state with a React context: https://stackoverflow.com/questions/69675357/react-what-is-the-proper-way-to-do-global-state
Refactor connection to web socket server by adding one previous step to joining the chat	High	Improvement 🚀 Refactor 🔧 Task 📌	Complete	Sprint 05	@ October 30, 2022 0:39	@ October 30, 2022 1:18	0	Now messages can be sent without being in a chat
Add a new page to send information message to all users	High	Improvement 🚀	Complete	Sprint 05	@ October 30, 2022 22:30	@ October 31, 2022 2:34	0	In frontend we make a new page with a chat (only for administrators) and every message that is sent, is globally notified (new type and task). Also, I can define buttons to have predefined messages to send.
Remove refresh token from sessions, because it is not used	High	Refactor 🔧	Complete	Sprint 05	@ October 28, 2022 1:57	@ October 28, 2022 2:40	0	Also from frontend in Redux
Administration button to schedule the restart of the server	Very high	Improvement 🚀	Complete	Sprint 05	@ October 29, 2022 0:40	@ October 29, 2022 16:06	8	
Migrate useEffect with API calls with useQuery	Medium	Improvement 🚀	Not Started					
Leave chat (user action)	Medium	Improvement 🚀	Complete	Sprint 05	@ October 26, 2022 18:53	@ October 26, 2022 19:10	0	
Add support for text and pdf files messages	High	Improvement 🚀	Complete	Sprint 05	@ October 26, 2022 0:38	@ October 26, 2022 1:30	0	Remains giving style to the message boxes
PDF visualizer	Medium	Improvement 🚀	Complete	Sprint 05	@ October 26, 2022 15:30	@ October 26, 2022 17:38	0.3	
Check why the sessions are removed every time the scheduled task is executed	Medium	Bug 🐛	Complete	Sprint 05	@ October 25, 2022 20:25	@ October 25, 2022 20:32	0	Token did not have the prefix when verifying it. Changed implementation of the method to support having or not the "Bearer " prefix
Improve observability with Grafana	Medium	Configuration ⚙️ Improvement 🚀	Complete	Sprint 06	@ November 27, 2022 0:14	@ December 7, 2022 17:15	-1	https://grafana.com/blog/2022/04/26/set-up-and-observe-a-spring-boot-application-with-grafana-cloud-prometheus-and-opentelemetry/ Also add Loki for managing the logs
Improve performance when showing the emojis	Very high	Improvement 🚀	In Progress	Sprint 05	@ October 26, 2022 19:38			
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	Sprint 05	@ 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 🔧	Complete	Sprint 05	@ October 28, 2022 1:35	@ October 28, 2022 1:56	0	
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	Sprint 05	@ October 21, 2022 19:30	@ October 22, 2022 17:23	10	Deleting chats, remove history
Block users for 10 minutes when they introduce a wrong password	Medium	Improvement 🚀 Task 📌	Not Started					See description for more details
Block users	Medium	Improvement 🚀	Complete	Sprint 10	@ March 11, 2023 16:00	@ April 5, 2023 13:00	-1	
Add friends	Medium	Improvement 🚀	Complete	Sprint 10	@ March 11, 2023 14:00	@ March 12, 2023 3:18	8	Disable users from sending messages themselves, if users see their own profile, go to the profile page (both new functionality)
Poll system integrated in the chat	Very low	Idea 💡 Improvement 🚀	Not Started					
Improve log configuration file to make files with RollingFilePolicy.	Medium	Improvement 🚀	Complete	Sprint 05	@ 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	Sprint 05	@ October 16, 2022 18:32	@ October 17, 2022 21:12	8	No self-signed SSL used. We use <code>certbot</code> to get the certificates automatically with reliable CA.
Configure home server	Critical 🔥	Configuration ⚙️ Improvement 🚀	Complete	Sprint 05	@ October 14, 2022 16:00	@ October 16, 2022 22:00	16	
Make WebSockets work in the home server	Critical 🔥	Configuration ⚙️	Complete	Sprint 05	@ October 15, 2022 23:28	@ October 15, 2022 23:32	0	Only needed to change WebSocket connection URL
Make dockerfiles to have a dev and prod containers of the backend	Very high	Configuration ⚙️ Improvement 🚀	Cancelled					Unnecessary. Background processes can be used
Backend works correctly in home server	Critical 🔥	Configuration ⚙️	Complete	Sprint 05	@ October 15, 2022 20:00	@ October 15, 2022 23:15	0.7	
Improve Active users tab with list switching (active / all)	Medium	Improvement 🚀	Complete	Sprint 05	@ October 13, 2022 15:50	@ October 13, 2022 17:34	0.5	
Improve chat grid at home page	Very high	Bug 🐛	Complete	Sprint 05	@ October 13, 2022 0:02	@ October 13, 2022 0:22	0	Removed margin from DropDown and must be set whenever it is used to the inner components











































Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Infinite requests when going to add new group	High	Bug 🐛	Complete	Sprint 05	@ October 12, 2022 23:24	@October 12, 2022 23:26	0	A dependency list in useEffect was not present
Register email address in Sandgrid to prevent automatic emails to be received in spam folder	Very high	Bug 🐛	TempCancel	Sprint 05	@ October 12, 2022 2:35			Emails are received in spam folder 📧. Moved to Gmail (using tricks 🧙 by creating account from mobile without adding it) and new implementation. The links are modified and email is sent to spam.
Published Instagram story to get feedback	Medium	Research 🔍	Complete	Sprint 05	@ 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					Server must be restarted to enable users to use the chat again
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	Very low	Idea 💡 Improvement 🚀	Not Started					
Fit content to all devices (Phone, tablet, pc)	Critical	Improvement 🚀 Task 🛠️	Complete	Sprint 05	@ October 11, 2022 23:07	@October 12, 2022 0:29	0	
Implementación de algo de funcionalidad más especial. Papers	Very low	Idea 💡	Not Started					
Rank system, get a chat badge (or different colors) when user has written a lot of messages	Very low	Idea 💡 Improvement 🚀	Complete	Sprint 08	@ February 6, 2023 0:47	@February 7, 2023 3:52	10	This implies changes to the database, temporal changes made with red (in the drawings)
Meeting with David Nkoto from Platform.sh	Critical	Task 🛠️	Complete	Sprint 04	@ October 4, 2022 11:00	@October 4, 2022 11:20	0	No contract agreed
Refactor message strategy to have only one implementation of handle method (Audio/Image/Text/Video classes)	Medium	Refactor 🔄	Complete	Sprint 04	@ 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 amiqueldiez	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	Sprint 04	@ September 28, 2022 16:20	@October 7, 2022 0:19	-1	Needed refactor of Service and Controller classes
Notification when a message is received	Low	Improvement 🚀	Complete	Sprint 04	@ October 8, 2022 1:36	@October 8, 2022 2:00	0	Source of the notification sound: https://mobcup.net/ringtone/mobile-ah-song-jcxfody9
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	Sprint 04	@ September 27, 2022 11:30			
Fix problems with tests	Very high	Bug 🐛 Test 🧪	Complete	Sprint 04	@ September 26, 2022 21:45	@September 26, 2022 22:36	0.2	This solves problems. This solves problems running tests with Maven
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					Watch video
Document the pattern used by the ChatMap class	High	Documentation 📖	Complete	Sprint 04	@ 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	Sprint 04	@ September 22, 2022 15:00	@October 6, 2022 20:17	-1	Paused the implementation due to creation of the document for the final delivery
Add support for commands with "/" character	Low	Improvement 🚀	Complete	Sprint 05	@ November 5, 2022 2:01	@November 5, 2022 15:33	9	Use a Tokenizer (idea)
Add the clock (bean) to Instant.now() calls	Medium	Task 🛠️	Complete	Sprint 04	@ September 16, 2022 13:26	@September 16, 2022 13:36	0	

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
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	Very low	Idea  Improvement  Task 	Not Started					
Reduce emojis file to only needed fields (name, emoji, unicode, category, sub-category)	Medium	Improvement 	Complete	Sprint 04	@ September 15, 2022 12:36	@ September 15, 2022 13:46	0	Make a table for emojis
Refactor the messages sent over WS to remove encoding field (we only use UTF-8) and simplify to UTF8	High	Refactor 	Complete	Sprint 04	@ 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	Sprint 04	@ September 16, 2022 14:03	@ September 19, 2022 21:04	-1	Api: https://github.com/aboutnik/emojis-world Or have the list in memory or read from the file or have a table in db. Decision: DB. When the users enters character ':' some emojis are suggested. The emoji is between ':' characters
Fix problems with audio files that move one message down	Critical 	Bug 	Complete	Sprint 04	@ September 14, 2022 16:47	@ September 14, 2022 17:24	0	Only fails when the app is executed locally
Save the files uploaded to the database	High	Task 	Complete	Sprint 04	@ September 14, 2022 13:50	@ September 14, 2022 14:52	0	
Limit the size of the files uploaded	High	Improvement 	Complete	Sprint 04	@ 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	Sprint 04	@ September 13, 2022 12:15	@ September 13, 2022 13:42	0	Also added Statistics page
Solve problem of not sending USER_LEFT (and USER_JOINED) messages to the server	Critical 	Bug 	Complete	Sprint 04	@ September 12, 2022 17:15	@ September 12, 2022 20:00	0.1	The client was disabled before sending the USER_LEFT message. USER_JOINED are sent after checking that the user can connect to the chat, in the chat page
Fix exception when a user exits a chat. User is not created and added to chat	Critical 	Bug 	Complete	Sprint 04	@ September 12, 2022 8:35	@ September 12, 2022 8:54	0	The connection with the method does not work as expected. The connection is made in the onopen of the WebSocket
Remove checks on register. Only show error when server sends the response	High	Refactor  Task 	Cancelled					
Change all TextField components to be size="small"	Low	Refactor 	Complete	Sprint 03	@ 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	Sprint 03	@ September 9, 2022 14:00	@ September 9, 2022 14:30	0	https://t3.og/blog/post/vite-vercel
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)	Very low	Idea  Task 	TempCancel					
Add email templates for register and password change	High	Improvement 	Complete	Sprint 03	@ 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	Sprint 04	@ September 14, 2022 20:04	@ September 14, 2022 20:16	0	Made to prevent errors. Every 3 minutes, check if there are null clients in the list.
Extract to a JSON the router structure in App component	High	Refactor 	Complete	Sprint 03	@ September 8, 2022 16:11	@ September 8, 2022 16:54	0	Make an array of objects that have: path, type (public, private or admin), component. If public, add restricted or not
Add style to have chats structured in a grid in Home view	High	Improvement  Task 	Complete	Sprint 03	@ 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	Sprint 04	@ September 12, 2022 23:00	@ September 13, 2022 1:00	0	When clicking the chat name or show next to it
Remove express and file to serve frontend (since Vercel does not need it)	Very high	Refactor 	Complete	Sprint 03	@ September 8, 2022 17:06	@ September 8, 2022 17:10	0	Vercel uses package.json file
Configure email sender to be able to send emails when users are registered or password are reset	Very high	Improvement  Task 	Complete	Sprint 03	@ September 9, 2022 0:05	@ September 9, 2022 1:11	0.5	
Make a Feature readme file	Very high	Documentation 	Complete	Sprint 03	@ 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	Sprint 03	@ September 9, 2022 20:35	@ September 10, 2022 3:06	2.2	Create a new column in users' table (Or a composite value (so a new table is created) with password, reset password and a code to generate when it resets or forgets). The last option implies heavier refactoring (finally first option is implemented). See Forgot password functionality .
If user checks the "Remember me" checkbox, extend the expiration time of the access token to 2 weeks or more	High	Improvement 	Complete	Sprint 03	@ September 9, 2022 20:05	@ September 9, 2022 20:27	0	














As Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Be able to handle several files at once, to keep the order in which they were sent	Medium	Improvement	Complete	Sprint 03	@ 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	Sprint 03	@ September 6, 2022 1:00	@ September 6, 2022 1:50	0	
Show how many images are attached to the message	Low	Improvement Task	Complete	Sprint 03	@ September 6, 2022 1:20	@ September 6, 2022 1:40	0	They must be confirmed in the dialog. If the dialog is closed without clicking on the "Attach" button, the files will remain selected but the user cannot send the message. After sending a message, if images were selected but not attached, they are removed from the selection.
Migrate applications from Heroku to Vercel	High	Task	Complete	Sprint 03	@ September 5, 2022 23:00	@ September 5, 2022 23:15	0	Only frontend was moved. Heroku will not have free plans after 28th November 2022. Contact Vercel if they have student plans
Add support for having database in localhost	High	Task	Complete	Sprint 03	@ 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	Sprint 03	@ September 4, 2022 23:00	@ September 4, 2022 23:27	0	
Idle connection of WebSockets	Critical	Bug	Complete	Sprint 03	@ September 7, 2022 14:22	@ September 7, 2022 17:00	1	Needs to be tested. Solved by adding an interval
Fix url when receiving response from server	Critical	Bug	Complete	Sprint 03	@ August 28, 2022 23:19	@ August 29, 2022 0:40	0	
Add functionality for adding more media files	Very high	Task	Complete	Sprint 03	@ August 28, 2022 23:15	@ September 2, 2022 17:34	-1	Another Strategy pattern in FileController
Migrate max width determination to backend	Low	Refactor	Complete	Sprint 03	@ August 28, 2022 20:43	@ August 28, 2022 20:55	0	
Group multiple images/videos in one message	Low	Improvement Task	Not Started					New message type constant and component
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	Sprint 03	@ August 27, 2022 14:30	@ August 27, 2022 23:45	1.5	Improvement for this task, since through WebSockets the data transmission is limited to few kB.
Add a small dropdown to select the type of file the user is going to send	High	Improvement	Not Started					
Add support to encode (in base64) files to be able to send to the server as normal WebSocket messages	High	Task	Complete	Sprint 03	@ August 26, 2022 22:50	@ August 26, 2022 23:20	0	They are sent as HTTP requests, not through WebSocket, since there is a data size limit. See Upload images from frontend with drag'n'drop and encode into base64
Add drag 'n' drop to select files to send to the chat	High	Task	Complete	Sprint 03	@ August 26, 2022 20:50	@ August 27, 2022 15:00	12	
Add the README.md	Low	Documentation	Complete	Sprint 03	@ August 26, 2022 19:05	@ August 26, 2022 21:00	0.2	Added file to frontend and backend
Chat statistics - Per user, number of messages, words written...	Very low	Idea Improvement	Not Started					
Replace disk reads for memory reads when reading chat history	Very low	Idea Improvement	Not Started					
Schedule the upload of the history file to S3 after a period of time	High	Improvement	Reference	Sprint 03	@ August 24, 2022 20:30	@ August 24, 2022 20:50	0	See Upload the file to S3 every 10 minutes
Read history from file stored in disk if it is downloaded. To have updated history to new users	Critical	Bug Refactor	Complete	Sprint 03	@ August 23, 2022 18:00	@ August 24, 2022 1:00	2	Since it is always downloaded when the first user is connected, if more users connect to the chat, it is only needed to read from the file
Implement pagination of the chat history	Medium	Improvement Task	Complete	Sprint 08	@ January 25, 2023 15:54	@ January 26, 2023 16:38		Add a "Show more" button in the top of the chat
Do the documentation of the Java code	Very high	Documentation	Complete	Sprint 03	@ August 24, 2022 22:47	@ August 26, 2022 0:40	12	
Solve problems of websocket connections in remote	Critical	Bug	Cancelled					There are no problems now (next day). Because the chat was closed and the history file was uploaded, so no errors occur
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	Very low	Idea Improvement Refactor	TempCancel					
See this to send partial data (maybe)	Very low	Idea Improvement Refactor	TempCancel					Maybe change by this
Add a reducer for chat and store the current chat	Medium	Improvement	TempCancel					Update it before going to the page and take it from the state to show its name in the chat view
Order active users by name	Low	Improvement	Complete	Sprint 03	@ August 23, 2022 20:25	@ August 23, 2022 20:34	0	
Use strategy pattern to make MessageHandler more readable	Very high	Refactor	Complete	Sprint 03	@ August 22, 2022 0:16	@ August 22, 2022 20:47	15	
Upon connection of the user, receive all the messages of the chat up to a limit of 3 days before	High	Improvement	Complete	Sprint 03	@ August 22, 2022 20:49	@ August 22, 2022 23:48	0.75	New message type. Messages are sent after the listeners are set
Show popup when hovering hour of message to display full date	High	Improvement	Complete	Sprint 03	@ August 21, 2022 19:26	@ August 21, 2022 20:00	0	Also added dateformat lib. See Show popup when hovering hour of message to display full date

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
When clicking a username in a chat, display a small popup with a button to start a user-user chat or other actions	High	Improvement Task	Complete	Sprint 05	@ October 26, 2022 23:10	@October 26, 2022 23:48	0	Causes errors and the page turns completely white because the following exception occurs <code>Rendered fewer hooks than expected. This may be caused by an accidental early return statement.</code> Update: Initial date: 20 August 0:20 -- 21 August 23:23 Now no errors
Edit diagrams and refactor entity classes (relation classes) and SQL definition to make columns match the table name	Very high	Bug Refactor	Complete	Sprint 03	@ 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	Sprint 03	@ September 7, 2022 17:31	@September 7, 2022 20:56	1.5	Add new column in chat table. They are codes, not links (for the moment)
Listen to new messages and check if they are <code>USER_JOINED</code> or <code>USER_LEFT</code> and send a new type of message to get the connected users of the chat in the background	Medium	Improvement	Complete	Sprint 03	@ August 19, 2022 22:00	@August 20, 2022 0:17	0	Use <code>ACTIVE_USERS_MESSAGE</code>
Add a list of connected users to the chat view	Very low	Idea Improvement	Complete	Sprint 03	@ August 19, 2022 22:37	@August 20, 2022 0:13	0	
Improve chat with name and more information about it	High	Task	May change	Sprint 03	@ 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	Sprint 03	@ August 17, 2022 20:32	@August 17, 2022 23:20	0.75	IMPORTANT: PK classes are now <code>@embeddable</code> and they are used in related entity class as a private field with the annotation <code>@EmbeddedId</code>
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	Cancelled					It is implemented with a React Context
Change type of chat in DB to longer name	High	Refactor	Complete	Sprint 03	@ August 16, 2022 20:56	@August 17, 2022 22:30	1	Also changed metadata column to json type
Make chatReducer to shorten loading of home screen when requesting all available chats for a user	High	Improvement Task	Complete	Sprint 03	@ August 16, 2022 16:55	@August 16, 2022 17:36	0	Added the request of all the chats on login and return to client.
Remove sessionId of the user both in frontend and backend	Medium	Refactor	Revision					Could be kept to support multi-session connections
Add a callback to client to execute it when a message is sent correctly.	Low	Refactor	Complete	Sprint 03	@ August 16, 2022 17:41	@August 16, 2022 17:58	0	
Show available chats to user in home view	High	Task	Complete	Sprint 03	@ August 15, 2022 19:33	@August 15, 2022 23:05	1.4	See Home page with chat selection
Try the reconnection of the WebSocket in frontend	Medium	Improvement	Complete					Reference Send a ping message from frontend to keep the connection alive
How to escape SQL reserved keywords with Spring boot, JPA and Hibernate - Java JEE tutorials by Tarik Chrouki	High	Bug	Complete	Sprint 03	@ 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	Sprint 03	@ August 18, 2022 20:22	@August 19, 2022 0:15	1.5	Made in 2 phases (Extraction, Refactoring). Also extracted Dialogs to new components
Add groups to all diagrams	Critical	Task	Complete	Sprint 03	@ August 13, 2022 0:45	@August 13, 2022 14:05	12	
Decide structure of the chat URLs	Very high	Task	May change	Sprint 03	@ 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	Sprint 03	@ September 8, 2022 17:23	@September 8, 2022 19:59	0.5	This is done to prevent users to join a chat that is not available to them (modify the URL by hand). See also Restrict the connection to chats. Users can only connect to available ones
Decide chat type names	High	Refactor	Revision	Sprint 03	@ 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	Very low	Idea Task	Not Started					
Add support to compute the color hash that a user has.	Very low	Idea Improvement	Not Started					
Domain diagram	High	Documentation	Revision	Sprint 02	@ August 9, 2022 22:10	@August 9, 2022 22:30	0	13-08-22 > It has changes (add groups)
Use case diagram	High	Documentation	Revision	Sprint 02	@ 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	Sprint 02	@ August 11, 2022 23:00	@August 11, 2022 23:40	0	
Check that all routes have the correct http method and permissions	High	Task	Complete	Sprint 02	@ August 7, 2022 16:53	@August 7, 2022 17:32	0	

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Update old methods that used ResponseEntity? > to use the new HttpResponseMessage class	Low	Improvement Refactor	Complete	Sprint 02	@ August 6, 2022 23:29	@ August 6, 2022 23:42	0	
Implement Degree controller and associated Service in backend	High	Task	Revision	Sprint 02	@ 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	Sprint 02	@ August 6, 2022 14:26	@ August 6, 2022 16:14	1.5	
Make administrator view in frontend	Medium	Improvement Task	Complete	Sprint 02	@ August 6, 2022 12:30	@ September 13, 2022 13:42	-1	Implementation of services
Create updated entities in JPA	High	Task	Complete	Sprint 02	@ August 5, 2022 22:15	@ August 6, 2022 23:26	12	Check if the entities can contain only the reference object, not the id and the object (see Session last 2 attributes). Update: entities only have the id (if an object of other entity is required)
Check migration guide java-jwt to version 4	Low	Security	Complete	Sprint 03	@ 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	Sprint 03	@ September 8, 2022 17:23	@ September 8, 2022 19:59	0.5	See also When entering a chat, send a http request to check if the user has access to that chat
New messages when user joins and leaves a chat	High	Refactor Task	Cancelled	Sprint 02	@ August 3, 2022 20:15	@ August 4, 2022 22:30		Decided to rollback to the working implementation (create a socket per chat connection)
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	Complete	Sprint 02	@ August 1, 2022 23:18	@ August 3, 2022 0:05	18	Added a builder for comfort, generalized filetype query to S3
Get all files that a user has sent over the time	Very low	Idea	Not Started					
Achievements' system	Very low	Idea Improvement	Not Started					
When uploading file (finished chat) delete from disk. If chat is reopened, download file and keep writing	Medium	Improvement Task	Complete	Sprint 02	@ July 31, 2022 20:06	@ August 1, 2022 23:05	20	Key to check for existence was invalid (.txt was not added to the search which returned false)
Update docs	Very high	Documentation	Complete	Sprint 02	@ 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	Sprint 02	@ July 31, 2022 19:55	@ July 31, 2022 20:05	0	
Check token on connect to websocket	Medium	Improvement Security Task	Revision	Sprint 02	@ August 5, 2022 20:30	@ August 15, 2022 17:04	-1	
Better handling of chats with a new class	Low	Refactor	Complete	Sprint 02	@ July 31, 2022 15:45	@ July 31, 2022 17:22	0	
Migrate backend to spring websocket support with custom Handlers	High	Refactor Task	Complete	Sprint 02	@ July 30, 2022 18:42	@ July 31, 2022 15:00	10	Client is receiving 403 code from the server
Migrate from Java-WebSocket lib to Jetty websockets in backend	High	Refactor Task	Complete	Sprint 02	@ July 29, 2022 21:03	@ July 30, 2022 15:59	14	Tested with an auxiliary file in frontend
Migrate from native WebSocket to websocket library in frontend	High	Refactor Task	Cancelled	Sprint 02	@ July 30, 2022 16:05	@ July 30, 2022 17:31		I can use the native WebSocket. 1.5 hours lost 😞
Migrate from native WebSocket to SockJS and	Medium	Refactor	Cancelled			@ July 30, 2022 23:00		See update . Tested but it did not work.
Provide more style to message components	High	Improvement Task	Complete	Sprint 02	@ July 28, 2022 17:05	@ July 28, 2022 20:33	0.8	Can be updated anytime
If up arrow key is pressed, visit the message history	Very low	Idea Improvement	Not Started					
Show incoming messages in ChatBox	High	Task	Complete	Sprint 02	@ July 27, 2022 20:30	@ July 28, 2022 16:55	14	1 - Only shows the last message received 2 - I lasted 6 minutes the following day after watching how to use prevState in setter
Make a record class to identify the WS user in backend	High	Refactor Task	Complete	Sprint 02	@ July 27, 2022 17:57	@ July 27, 2022 18:42	0	
Receive message in component	High	Refactor Task	Complete	Sprint 02	@ July 27, 2022 0:48	@ July 27, 2022 2:07	0	
Improve storage of clients in backend	High	Task	Complete	Sprint 02	@ July 26, 2022 22:52	@ July 27, 2022 0:32	0.6	WSClietID record was created
Improve message handling between front and back	High	Refactor Task	Complete	Sprint 02	@ 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	Sprint 02	@ July 26, 2022 18:22	@ July 26, 2022 18:38	0	

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Limit rate at which users can send messages to 6 per second	Very low	Idea  Improvement  Task 	Not Started					
Return an existing session to the user that logs in with other host	Very low	Idea  Improvement  Task 	Not Started					
Add the sessionId when sending the response to the client	High	Refactor  Task 	Complete	Sprint 02	@ July 26, 2022 16:37	@ July 26, 2022 17:30	0.25	Previously it was sent the date of sign in
Make a deploy test in Heroku, to verify the functionality of the sockets	High	Research 	Complete	Sprint 02	@ July 28, 2022 17:42	@ August 5, 2022 0:45	-1	Working with S3 correctly, remains connecting to available chats, not the one you want
Receive messages in frontend from other clients and log to console	High	Task 	Complete	Sprint 02	@ July 26, 2022 0:30	@ July 26, 2022 1:28	0	Received and printed in class, not component
Configure frontend to send messages. Create class and Hook	High	Task 	Complete	Sprint 02	@ 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	Very low	Idea 	Cancelled					
File writing from several threads (maybe use synchronized)	Low		Cancelled					
Create WebSocket server	High	Task 	Complete	Sprint 02	@ 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	Sprint 02	@ July 23, 2022 17:40	@ July 23, 2022 17:47	0	
Send messages to other members of the chat	Critical	Task 	Complete	Sprint 02	@ 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	Sprint 02	@ July 21, 2022 20:00	@ July 21, 2022 20:57	0	
Move header of message up one component (to ChatMessage.jsx)	Medium	Task 	Complete	Sprint 02	@ 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	Sprint 02	@ July 27, 2022 0:26	@ July 27, 2022 0:29	0	
Add a new table in DB (Chats)	Critical	Task 	Complete	Sprint 02	@ July 21, 2022 17:20	@ July 21, 2022 19:06	0.5	
When hovering the date of the message, show full date (with millisecond precision)	Low	Task 	Reference					
Show a division of messages per day	Low	Task 	Not Started					
Font size selector	Low	Improvement  Task 	Not Started					
Upload the file to S3 every 10 minutes	Medium	Improvement  Task 	Revision	Sprint 03	@ August 24, 2022 20:30	@ August 24, 2022 20:50	0	Do this if a chat has clients for this amount of time
Store messages in a file with history	High	Task 	Complete	Sprint 02	@ July 21, 2022 22:00	@ July 21, 2022 23:30	0.4	
Send messages from front to back	High	Task 	Complete	Sprint 02	@ July 21, 2022 19:55	@ July 21, 2022 21:18	1	1 hour interrupted, I was updating the security routes in a separate class.
Read messages in backend	High	Task 	Complete	Sprint 02	@ July 21, 2022 21:20	@ July 21, 2022 21:30	0	
More style to chat and messages	Medium	Task 	Complete	Sprint 02	@ July 20, 2022 17:55	@ July 20, 2022 23:55	2	Messages are broken in several lines if long
Basic chat page	High	Task 	Complete	Sprint 02	@ 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	Sprint 02	@ 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	Very low	Idea 	Cancelled					
Make a Router class to configure HTTP methods and permissions	Low	Task 	Cancelled	Sprint 01	@ 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	Sprint 01	@ June 26, 2022 0:50	@ June 26, 2022 1:15	0	
Profile page with user data (session, name, ...)	High	Task 	Complete	Sprint 01	@ 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	Sprint 01	@ June 25, 2022 15:00	@ June 25, 2022 15:20	0	

Aa Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Frontend API url change when code is in production or in development env	High	Bug Refactor	Complete	Sprint 01	@ June 25, 2022 0:30	@ June 25, 2022 0:50	0	
App renaming	High	Refactor	Complete	Sprint 01	@ June 24, 2022 23:54	@ June 25, 2022 0:15	0	
Solve problem with Instant class of the domain models	High	Bug Refactor	Complete	Sprint 01	@ June 24, 2022 22:45	@ June 24, 2022 23:15	0.25	ObjectMapper must have JavaTimeModule registered
Refactor Responses with the new class	Medium	Refactor Task	Complete	Sprint 01	@ June 24, 2022 22:00	@ June 24, 2022 22:33	0	
Improve custom HttpRequest functionality	High	Task	Complete	Sprint 01	@ June 24, 2022 0:05	@ June 24, 2022 1:04	0	Little modification was done
Improve custom HttpResponse functionality	High	Task	Complete	Sprint 01	@ June 23, 2022 22:40	@ June 24, 2022 3:08	0	
Migration of frontend to other repo	Critical	Bug Refactor Task	Complete	Sprint 01	@ June 23, 2022 0:37	@ June 23, 2022 1:35	0	
Better configuration of deployment CORS config	Critical	Task	Complete	Sprint 01	@ June 22, 2022 20:30	@ June 23, 2022 1:35	1.5	
Configure deploy functionality	Critical	Initialization Task	Complete	Sprint 01	@ June 21, 2022 13:00	@ June 22, 2022 1:46	3	
Backend register functionality	Critical	Task	Complete	Sprint 01	@ June 20, 2022 21:15	@ June 24, 2022 22:05	26	
Toolbar with theme switcher	Critical	Task	Complete	Sprint 01	@ June 20, 2022 20:55	@ June 20, 2022 21:03	0	
Store Reducers and Actions for Redux	High	Task	Complete	Sprint 01	@ June 20, 2022 20:41	@ June 20, 2022 20:50	0	
Theme switcher and color palette	Low	Task	Complete	Sprint 01	@ June 20, 2022 20:36	@ June 20, 2022 21:03	0	
Register page and switch between Login and Register	High	Task	Complete	Sprint 01	@ 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	Sprint 01	@ June 17, 2022 15:45	@ June 17, 2022 15:50	0	Made with IntelliJ UI JPA Inspector. Returned to Timestamp because Jackson cannot make correct bindings for that type
Remove Service's interfaces	High	Refactor	Complete	Sprint 01	@ June 17, 2022 14:00	@ June 17, 2022 14:17	0	
Fix CORS security	Critical	Bug Task	Complete	Sprint 01	@ June 15, 2022 17:50	@ June 15, 2022 18:56	0	Improved using WebMvcConfigurer in main class with a Bean. This configures CORS globally
When receiving a request, check if the token exists in DB	High	Task	Not Started	Sprint 01	@ June 16, 2022 18:00			
Fix login controller does not work after AuthenticationFilter	Critical	Bug Configuration	Complete	Sprint 01	@ June 15, 2022 13:00	@ June 15, 2022 14:10	0	Problem with the LogoutFilter. Removed
Improve Logout functionality with a Filter and then call the controller	High	Task	Cancelled	Sprint 01	@ June 15, 2022 2:00	@ June 15, 2022 18:40	7	
Solve problem with user creation by admins	Critical	Bug Task	Complete	Sprint 01	@ June 15, 2022 0:29	@ June 15, 2022 1:08	0	I'm a bit idiot
Ability to close all sessions of the user	Medium	Task	Not Started					
Update DB to include refreshToken	High	Configuration Task	Complete	Sprint 01	@ June 14, 2022 21:43	@ June 14, 2022 21:48	0	
SessionService update when refreshToken is used	Medium	Task	Not Started					
Implement SessionService with capability to save, delete	High	Task	Complete	Sprint 01	@ June 14, 2022 21:37	@ June 14, 2022 22:00	0	
Improve Filters with annotations	Medium	Refactor	Cancelled	Sprint 01	@ June 15, 2022 0:18	@ June 15, 2022 0:27	0	
Backend routes with constants	Medium	Configuration Task	Complete	Sprint 01	@ 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	Sprint 01	@ June 14, 2022 17:30	@ June 14, 2022 20:40	1	
Document - Design model	High	Documentation Initialization	Complete	Sprint 02	@ August 9, 2022 19:50	@ August 9, 2022 20:15	0	
Document - Requirements specification	High	Documentation Initialization	Complete	Sprint 01	@ 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	Sprint 01	@ June 13, 2022 19:00	@ June 14, 2022 2:30	2	
From Cayenne to JPA	High	Configuration Task	Complete	Sprint 01	@ June 13, 2022 13:53	@ June 13, 2022 18:00	3	
Configure Spring security with basic requests	Critical	Configuration Task	Complete	Sprint 01	@ June 13, 2022 23:15	@ June 15, 2022 1:55	9	

As Projects	Priority	Type	Status	Sprint	Started	Finished	# Inactivity (h)	Notes
Translate Relational diagram to SQL	High	Initialization 	Complete	Sprint 01	@ June 12, 2022 23:40	@ June 13, 2022 0:20	0	
Probar el módulo de traducción en Frontend	Very low	Idea 	Not Started					
Create Relational diagram	High	Documentation  Initialization 	Complete	Sprint 01	@ June 11, 2022 22:18	@ June 12, 2022 2:30	3	
Create the model classes with Cayenne Modeler	Medium	Initialization 	Cancelled	Sprint 01	@ June 13, 2022 0:20	@ June 13, 2022 0:45	0	Migration to JPA
Create ER diagram	High	Documentation  Initialization 	Complete	Sprint 01	@ June 11, 2022 13:15	@ June 11, 2022 22:15	3	
Sistema de menciones	Very low	Idea  Research 	Not Started					
How To Build React with Java Backend For Production by Bhargav Bachina Bachina Labs Medium	Very low	Idea  Research 	Complete					
How To Develop and Build React App With Java Backend by Bhargav Bachina Bachina Labs Medium	Very low	Idea  Research 	Complete					

Bibliografía

- [1] Gant Laborde. *Deep NN for NSFW Detection*. (Accedido el 05/03/2023). URL: https://github.com/GantMan/nsfw_model.
- [2] *Sueldos — Indeed.com*. <https://es.indeed.com/career/salaries?from=gnav-homepage>. (Accedido el 08/06/2023).
- [3] Refactoring Guru. *Design Patterns - What is a Design Pattern?* (Accedido el 29/09/2022). URL: <https://refactoring.guru/design-patterns/what-is-pattern>.
- [4] Refactoring Guru. *Classification of patterns*. (Accedido el 30/09/2022). URL: <https://refactoring.guru/design-patterns/classification>.
- [5] 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>.
- [6] 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>.
- [7] *Docker overview — Docker Documentation*. (Accedido el 21/12/2022). URL: <https://docs.docker.com/get-started/overview/>.
- [8] *Volumes — Docker Documentation*. (Accedido el 21/12/2022). URL: <https://docs.docker.com/storage/volumes/>.
- [9] *Overview — Prometheus*. (Accedido el 21/12/2022). URL: <https://prometheus.io/docs/introduction/overview/>.
- [10] *Spring Boot app metrics - with Prometheus and Micrometer - Tutorial Works*. (Accedido el 22/12/2022). URL: <https://www.tutorialworks.com/spring-boot-prometheus-micrometer/>.
- [11] *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/>.
- [12] *Spring Boot 2.1 System Monitor — Grafana Labs*. URL: <https://grafana.com/grafana/dashboards/11378-justai-system-monitor/>.
- [13] *Visualizations — Grafana documentation*. URL: <https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/>.
- [14] *Overview — Grafana Loki documentation*. URL: <https://grafana.com/docs/loki/latest/fundamentals/overview/>.
- [15] *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>.
- [16] *NSFW JS*. <https://nsfwjs.com/>. (Accedido el 05/03/2023).
- [17] Thomas Hamilton. *Cyclomatic Complexity in Software Testing (Example)*. <https://www.guru99.com/cyclomatic-complexity.html>. (Accedido el 10/06/2023). Mayo de 2023.

- [18] *ZSH - THE Z SHELL*. <https://zsh.sourceforge.io/>. (Accedido el 14/10/2022).
- [19] *Oh My Zsh - a delightful & open source framework for Zsh*. <https://ohmyz.sh/>. (Accedido el 14/10/2022).
- [20] *Duck DNS - install*. (Accedido el 17/10/2022). URL: <https://www.duckdns.org/install.jsp>.