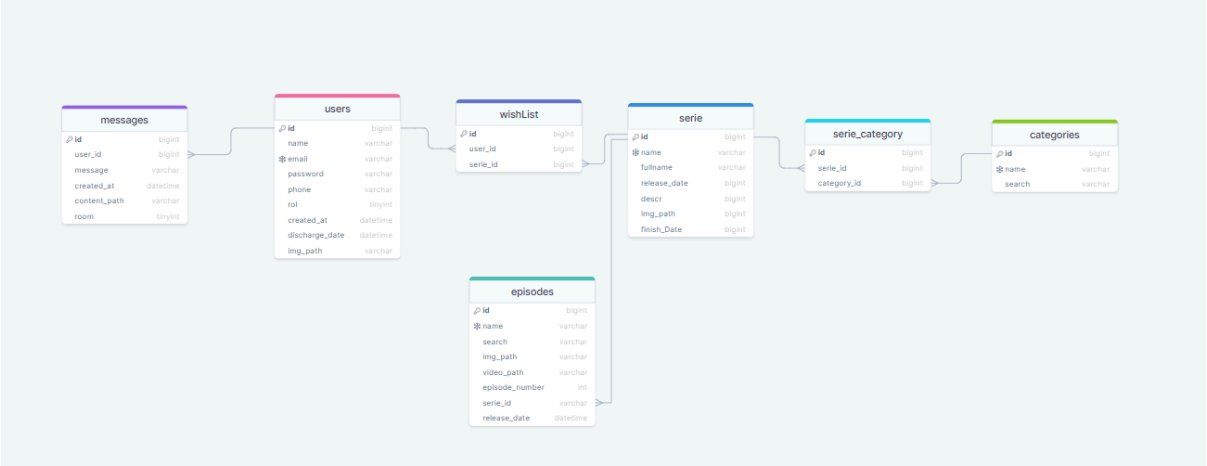
Tecnologías a usar:

Base de Datos:
Será una base de datos MySql, se valorará la posibilidad de migrarla a Postgre y pasar la tabla de mensajes a una MongoDB



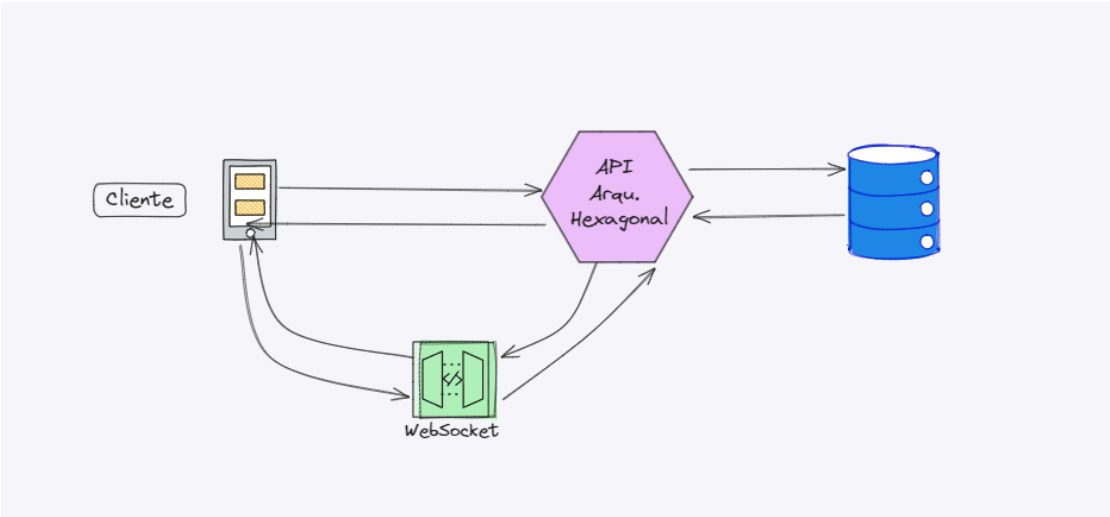
FrontEnd: Node.js+React

La tecnología principal que se usará será react 18.2 junto a tailwindcss como framework de css. También se implementarán librerías como:
React-router-dom: Para enrutamiento
React redux: Para gestionar el estado global de la aplicación
Axios: Llamadas a la api e intercepciones de respuestas y peticiones
Socket.io: Implementar el chat
Formik: Gestionar formularios, validación con la librería Yup, etc
Lodash: Mejorar el rendimiento de la aplicación.

BackEnd: Spring Framework + SpringBoot

Se usará Spring con Java2EE y JavaSE 17, tendrá una arquitectura hexagonal donde se usarán al menos 3 de los 6 puertos y adaptadores, HttpControllers, RESTClients y Entity Manager, en este caso Spring JPA, se valorará con el transcurso del proyecto añadir el adaptador de smtp, no se incluirán los adaptadores de MessageBroker Manager y CLI.
Además para manejar el flujo de la información se usará la arquitectura de Modelo de Capa de Servicio, lo que nos permitirá discernir mejor entre los puertos, adaptadores, servicios y las entidades para poder estructurar más sencillamente las capas de Infraestructura, aplicación y dominio.
Además se realizará mediante TDD, no muy en profundidad, pero se realizarán al menos 3-4 pruebas por cada controlador, servicio y 1-2 por cada repositorio y entidad.
Los archivos finalmente se guardarán en local debido a que solo poseo 3 meses de 30gb gratis en S3, aunque inicialmente se hizo con S3 implementándolo tanto en el frontend con la SDK y el CloudFront como en el backend.
Principales dependencias:

- Spring JPA: Para crear el adaptador y el puerto del EntityManager
- Spring Reactor: Para manipular flujos de datos reactivos producidos por secuencias de eventos asíncronos, principalmente para reproducir contenido conforme se va descargando de forma asíncrona, eficiente y concurrente, en mi caso es muy simple, pero implementa el patrón Observer/Observable con la clase Mono.
- Spring Websocket: Para comunicar clientes-servidor a tiempo real
- Spring Test y Mockito: Para realizar pruebas
- Spring security: Para implementar la autenticación por token y gestionar en general la seguridad de la aplicación
- Dependencias para agilizar el trabajo: Lombok, Spring Devtools, Jackson y Model Mapper



Despliegue:

Se valorarán las siguientes posibilidades

1. Docker: Creación de 2-3 contenedores para poder desplegar la aplicación
2. Docker+Kubernetes: Creación de 2-3 contenedores de docker, que se escalarán mediante VPA y HPA, en caso de optar por esta opción puede que cree un cluster aunque no sea necesario para la aplicación
3. AWS: Poseo 800h de ejecución de Servidores EC2 con RDS y EBS gratuítas y acceso a AWS Academy, si tengo tiempo para formarme en esta tecnología, puede que opte por hacerlo así.
4. Alojamiento en Plataformas como Render, Heroku o Netlify.

En este caso no se muy bien por qué opción optaré, hacerlo con docker y en Plataformas como las nombradas anteriormente sería más sencillo, aunque si tengo tiempo podría hacerlo con Kubernetes, AWS, o ambas para aprender y conocer estas plataformas tan demandadas.
La aplicación se desplegará en un servidor Tomcat

Tareas Pendientes:

Mejorar diseño, principalmente el modo claro
Seguridad del enrutamiento en el front
Implementación de Spring Security
Despliegue