**Instalación y configuración de servicios de mensajería instantánea (OpenFire, XMPP, Strophe.Js, NODEJS).**

**Autores: Adrián Lecrevisse y David Martín Huerta**

1. **Abstract .**

La comunicación digital se basa en la mensajería instantánea, favoreciendo la interacción en tiempo real entre los usuarios. Este proyecto pone en marcha un sistema de mensajería que utiliza Node.js, Express.js, SQLite, Socket.io, React y Vite, ofreciendo una solución actualizada y eficaz para la transmisión de mensajes.

El procesamiento de datos, la comunicación con la base de datos y otros servicios se realiza con Node.js y Express.js. El almacenamiento de mensajes en SQLite y la gestión de conexiones en tiempo real mediante Socket.io

La interfaz de usuario está desarrollada con React y Vite. La comunicación en tiempo real con el procesamiento de datos, la comunicación con la base de datos y otros servicios se realiza mediante Socket.io, asegurando la sincronización de los mensajes.

La aplicación garantiza una funcionalidad fácil, ideal para aplicaciones de chat en tiempo real.

1. **Introducción.**

Hoy en día, la mensajería instantánea se ha convertido en una herramienta fundamental para la comunicación de los seres humanos, permitiéndonos intercambiar información en tiempo real. En el día a día usamos aplicaciones como WhatsApp, Instagram, Twitter.. Las cuales utilizan este tipo de mensajería.

Este proyecto tiene como objetivo el desarrollo de una aplicación de mensajería en tiempo real, utilizando herramientas para garantizar una comunicación efectiva entre usuarios. Para ello, hemos construido una arquitectura basada en los siguientes componentes:

La parte de la interfaz gráfica, con la que interactúan los usuarios (Frontend), lo hemos construido con React y Vite, proporcionando una interfaz de usuario simple para una buena experiencia en la comunicación.

La parte que controla la lógica de negocio, procesa las solicitudes de los clientes y los mensajes, en definitiva la parte que se encarga de que la lógica de la aplicación funcione (Backend), está desarrollada con Node.js y Express.js.

Para almacenar los mensajes e información de los usuarios hemos utilizado SQLite, una base de datos que está integrada en todos los teléfonos móviles y en la mayoría de computadoras.

La comunicación en tiempo real es implementada mediante Socket.io. Es una biblioteca que permite la transmisión de datos en tiempo real entre clientes y servidor.

Esta aplicación permite a los usuarios enviar y recibir mensajes instantáneamente, manteniendo una conexión estable y segura. Además, la herramienta Vite que es utilizada para el desarrollo del Frontend, permite una carga rápida y Socket.io permite una comunicación en ambos sentidos sin necesidad de actualizar la aplicación.

1. **Explicación del modelo.**

Node.js: Es un entorno de ejecución de JavaScript basado en el motor V8 de Chrome, que puede ejecutarse en varios sistemas operativos sin necesidad de modificaciones en el código. Permite desarrollar aplicaciones rápidas y escalables.

Características:

1. Utiliza un modelo de entrada/salida no bloqueante. Cuando el servidor tiene que hacer tareas como esperar por operaciones como leer archivos, consultar bases de datos.. El servidor puede seguir atendiendo otras consultas mientras realiza dichas tareas. Esto permite que el servidor maneje muchas solicitudes simultáneamente sin tener problemas de rendimiento.
2. Gracias a el uso de WebSockets mediante la biblioteca Socket.io, Node,js es adecuado para aplicaciones de mensajería instantánea.
3. Node.js cuenta con un sistema de paquetes muy amplio. El gestor de paquetes npm permite instalar paquetes o bibliotecas de código de manera sencilla.

Ejemplo:

npm install -g pnpm

Con npm se puede encontrar una amplia variedad de librerías.

A pesar de esto hay una alternativa más eficiente a npm llamada pnpm que optimiza la instalación de dependencias reutilizando paquetes globales en lugar de duplicarlos en cada proyecto, esto reduce considerablemente el espacio en disco. En cambio npm instala las dependencias en la carpeta node\_modules del proyecto y si se instala una dependencia en otro proyecto, se vuelve a descargar e instalar.

1. Node.js permite manejar múltiples conexiones simultáneas sin bloquear lo que la convierte en una gran herramienta para crear aplicaciones como chats(ya que implican conexiones en tiempo real), juegos en línea…

Express.js: Express es un entorno de trabajo para Node.js que facilita la creación de servidores web y APIs, ofreciendo una forma sencilla de manejar solicitudes HTTP y gestionar la lógica del servidor.

Características:

1. Interfaz sencilla y fácil de usar

* Express proporciona una API limpia y sencilla para crear servidores web, manejar rutas y procesar solicitudes HTTP.
* Permite crear aplicaciones con pocas líneas de código, lo que mejora la productividad de los desarrolladores.

1. Definición de rutas para diferentes tipos de solicitudes HTTP

* Permite manejar fácilmente diferentes tipos de solicitudes HTTP como GET, POST, PUT, DELETE, entre otros.

1. Express puede conectarse fácilmente con bases de datos como SQLite, MongoDB, MySQL.
2. Seguridad.

* Express proporciona varias características para mejorar la seguridad de las aplicaciones web, como la protección contra ataques.

**¿Por qué usamos Express.js junto a Node.js?**

Node.js por sí solo proporciona un entorno de ejecución para JavaScript en el servidor, pero Express.js simplifica el proceso de desarrollo ya que ofrece una estructura organizada para manejar rutas y solicitudes, herramientas para agregar funcionalidades sin reescribir código desde cero y mejor gestión de errores y seguridad en APIs.

SQLite: SQLite es un sistema gestor de bases de datos fácil de usar. A diferencia de otros sistemas de bases de datos como MySQL o PostgreSQL, SQLite no requiere un servidor de bases de datos dedicado. Todo lo que necesita es un archivo de base de datos almacenado en nuestro ordenador. Este archivo contiene toda la información y las tablas de la base de datos, lo que hace que sea sencillo de administrar, transportar y respaldar.

Carácterísticas de SQLite:

1. Servidor embebido: No se necesita un servidor separado, la base de datos se guarda como un archivo y se accede directamente desde el código.
2. Lenguaje completo: Proporciona un lenguaje SQL completo, esto permite realizar todo tipo de consultas para manejar los datos de manera efectiva.
3. Fácil de integrar: SQLite es muy fácil de integrar en aplicaciones de Node.js, Express.js… Se puede acceder a él usando bibliotecas como sqlite3.

Tablas empleadas en la base de datos:

1. Conversation: Esta tabla almacena la información de las conversaciones ya sean chat de grupo o individuales. Campos:

* Id(INTEGER) (PK): Identificador único de la conversación.
* Name(STRING): Nombre de la conversación.

1. Message: En esta tabla guardamos los mensajes que se encuentran dentro de cada conversación. Campos:

* Id(INTEGER) (PK): Identificador único del mensaje.
* User\_id(INTEGER) (FK): Usuario que envió el mensaje.
* Conver\_id(INTEGER)(FK): Identifica a qué conversación pertenece el mensaje.
* Content(STRING): Texto del mensaje enviado.
* Send\_time(DATETIME): Fecha en la que envió el mensaje.

1. Participant: Esta tabla determina qué usuarios participan en cada conversación. Campos:

* Conver\_id(INTEGER)(FK): Identificador de la conversación.
* User\_id(INTEGER)(FK): Identificador del usuario que participa en la conversación.

1. Users\_identification: Almacena la identidad de los usuarios de la aplicación.

* Id(INTEGER)(PK): Identificador del usuario.
* Alias(STRING): Nombre del usuario.

1. Sqlite\_sequence: Esta es una tabla del propio SQLite, es usada para manejar que los valores en el resto de tablas establecidos como AUTOINCREMENT sean únicos.