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

**Autores: Adrián Lecrevisse y David Martín Huerta (DNI:70844244N)**

Índice:

1. Abstract………………………………………………………………………………Página 3
2. Introducción………………………………………………………………………..Página 3
3. Explicación del modelo………………………………………………………….Página 4
4. Resultados…………………………………………………………………………..Página 10
5. Conclusiones……………………………………………………………………….Página 12
6. Referencias…………………………………………………………………………Página 12
7. **Abstract .**

La comunicación digital se basa en la mensajería instantánea, permitiendo la interacción en tiempo real entre los usuarios. Este proyecto se desarrolla un sistema de mensajería que utiliza Node.js, Express.js, SQLite, Socket.io, React y Vite, ofreciendo una solución 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, desarrollada con **React y Vite**, ofrece una experiencia fluida y rápida, optimizando la carga y actualización de la aplicación sin necesidad de recargar la página.

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, es decir, 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 las 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 lo que la convierte en una gran herramienta para crear aplicaciones como chats , juegos en línea…
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.

¿Por qué es rápido?

Node.js usa un solo hilo de ejecución para manejar las solicitudes, sin embargo, eso no significa que solo pueda hacer una cosa a la vez. Cuando llega una solicitud, si es rápida (como devolver un texto), Node.js la procesa de inmediato. Si es una tarea más pesada, en lugar de bloquear el hilo, Node.js la delega a otro proceso y sigue atendiendo otras solicitudes. Node.js utiliza una biblioteca que se llama Worker Threads, esta biblioteca le permite usar múltiples hilos en segundo plano para realizar este tipo de tareas más pesadas. Gracias a estas características, Node.js es una de las tecnologías más utilizadas para el desarrollo del backend (lógica de negocio).

**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.

1. Al estar basado en Node.js, Express aprovecha que Node.js es asíncrono y no bloqueante, lo que permite manejar muchas conexiones simultáneamente sin perder rendimiento.

Ejemplo:

Utilizamos Express para crear un servidor HTTP:





Texto

El contenido generado por IA puede ser incorrecto.

Se ejecuta en un determinado puerto y puede manejar peticiones HTTP.

**¿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.

**Socket.io:**

Es una biblioteca de JavaSript que permite comunicación en tiempo real entre el servidor y el cliente a través de WebSockets. Su principal ventaja es que permite una comunicación bidireccional, es decir, tanto el servidor como el cliente pueden enviar y recibir datos en cualquier momento, sin necesidad de recargar la página:Forma, Rectángulo

El contenido generado por IA puede ser incorrecto.

Funcionamiento:

1. El servidor escucha conexiones de clientes usando Socket.io.
2. Los clientes se conectan al servidor para enviar y recibir mensajes en tiempo real.
3. **Se establece un canal de comunicación bidireccional**, lo que lo hace ideal para aplicaciones de chats.

Ejemplos:

1. Configuración del servidor:

En el servidor, Socket.io se inicializa y escucha conexiones de clientes:

Texto

El contenido generado por IA puede ser incorrecto.

Cuando se conecta un usuario, el servidor imprime el siguiente mensaje:

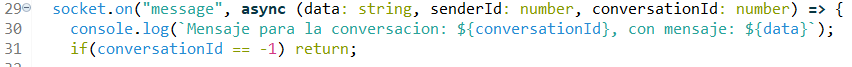


1. Comunicación bidireccional entre el cliente y el servidor:

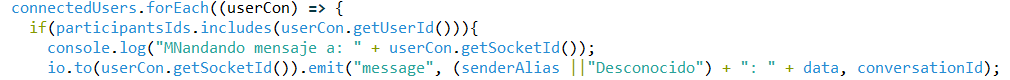
Cuando un usuario envía un mensaje en un chat, el cliente envía ese mensaje a el servidor:



El servidor recibe el mensaje:



Posteriormente, el servidor envía el mensaje a todos los participantes de esa misma conversación:



Y por último el cliente recibe el mensaje y los muestra por pantalla:

Texto

El contenido generado por IA puede ser incorrecto.

**React:**

React es una biblioteca de JavaScript que se utiliza para crear interfaces de usuario. React ayuda a desarrollar aplicaciones cuyos datos cambian constantemente ya que permite actualizar la interfaz de manera eficiente cuando el estado de uno de los componentes cambia.

Componentes en React:

React está basado en componentes reutilizables, esto quiere decir que la interfaz de usuario se divide en pequeñas piezas reutilizables. Un componente puede ser un botón, por ejemplo:



Se correspondería con el siguiente botón:



TSX (TypeScript + JSX) en React:

TSX es una extensión de Typescript, permite escribir estructuras similares a HTML dentro del código de Typescript, esto facilita la creación de interfaces dinámicas y hace que el código sea más sencillo.

React y el DOM:

A la hora de actualizar objetos, React maneja el DOM(Document Object Model) de manera optimizada a través del Virtual DOM. El virtual DOM es una copia del DOM real, React lo utiliza para hacer cambios de manera más optimizada. Cuando un estado cambia, React genera una nueva versión del Virtual DOM y la compara con la versión anterior del Virtual DOM. Después React compara el nuevo Virtual DOM con el anterior y al final React actualiza solo las partes necesarias del DOM real, sin tener que modificar el resto de la página.

Hooks en React:

Los hooks son funciones especiales en React que permiten usar características como el manejo del estado y el ciclo de vida sin necesidad de escribir clases. Con los Hooks, los componentes funcionales pueden gestionar estados y otras funcionalidades que antes solo eran posibles en los componentes de clase.

Ejemplo: useState, se usa para manejar el estado de un componente.

**Vite:**

Es una herramienta que nos ayuda a desarrollar aplicaciones web de manera más rápida, está diseñada para frameworks como React. Vite se instala como una dependencia del proyecto y realiza lo siguiente:

* Convierte el código a una versión optimizada de tal manera que funcione en el navegador.
* Cada vez que se hace un cambio en el código de la aplicación se actualiza al instante esto se define como HMR(Hot Module Replacement).
* No compila todo el código de una sola vez, sino que lo hace cuando el navegador lo requiere.

¿Cómo usamos vite?

Para iniciar el servidor de desarrollo, simplemente usamos el comando pnpm run dev. Este comando está definido en el archivo package.json:

Texto

El contenido generado por IA puede ser incorrecto.

Se encarga de arrancar el servidor de desarrollo de Vite, cargar la aplicación rápidamente en el navegador, aplicar cambios en tiempo real sin recargar la página completa.

1. **Resultados.**

Mediante el uso de las herramientas previamente explicadas hemos logrado crear una aplicación con una interfaz de usuarios sencilla. Gracias a React y Vite, la aplicación carga rápidamente en el navegador, ofreciendo una respuesta inmediata sin necesidad de recargar la página, lo que mejora la experiencia del usuario. Socket.io ha permitido la comunicación en tiempo real, asegurando que los mensajes lleguen instantáneamente a los destinatarios, manteniendo una comunicación fluida. La aplicación permite gestionar varios usuarios a la vez, esto es posible gracias a Node.js . Además los mensajes de los usuarios se almacenan en la base de datos de SQLite, lo cual garantiza la persistencia de información a lo largo del tiempo.

Ejemplo de uso de la aplicación:

Para entrar a la aplicación primeramente debemos conectarnos a: <http://62.42.213.179:5173/> desde un nevagador.

Una vez dentro de la aplicación, se nos solicita introducir un nombre de usuario para iniciar sesión. En este caso, el primer usuario que se conecta es Alejandro:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

En otra ventana o navegador, también se realiza el mismo proceso para otro usuario, en este caso escribimos otro nombre de usuario:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Al ingresar al chat de Alejandro, en su pantalla verá una lista de usuarios conectados en la aplicación. Esta lista incluye a Pedro, entre otros usuarios. La interfaz muestra de manera clara quiénes están en línea y quién desconectados:

Imagen de la pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Para iniciar un chat, Alejandro puede pulsar sobre el nombre de Pedro en la lista de usuarios. Al hacer esto, se abre automáticamente un chat privado entre ellos. En esta pantalla, podrán enviarse mensajes en tiempo real:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Cuando Alejandro escribe un mensaje en su chat y lo envía, el mensaje aparece de inmediato en la ventana de Pedro en tiempo real. Pedro también puede responder y ver los mensajes de Alejandro en su propio chat. Pedro, al pulsar sobre el nombre de Alejandro en su lista de usuarios, puede acceder a la conversación que han tenido, y ver todos los mensajes enviados y recibidos, lo que le permite continuar la interacción en cualquier momento:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. **Conclusiones.**

En este proyecto hemos aprendido cómo herramientas como Node.js, React y Socket.io pueden juntarse para desarrollar aplicaciones en tiempo real. A lo largo del proceso de desarrollo, hemos comprendido la importancia de cada tecnología en la arquitectura de la aplicación y cómo su combinación permite ofrecer una buena experiencia a los usuarios.

El uso de Node.js y Express.js ha sido fundamental para la creación del servidor. Con Socket.io, hemos logrado una comunicación bidireccional en tiempo real, permitiendo la transmisión instantánea de mensajes. React y Vite han facilitado la construcción de la interfaz y gracias a SQLite hemos conseguido almacenar de una manera sencilla los usuarios, conversaciones y mensajes de la aplicación.

Este proyecto nos ha permitido profundizar en el desarrollo de aplicaciones web en tiempo real y comprender los retos asociados a la sincronización de datos entre múltiples clientes.

En el futuro, podrían añadirse mejoras como el envío de archivos, notificaciones o un sistema de autenticación más seguro.

1. **Referencias:**

<https://es.wikipedia.org/wiki/React>

<https://en.wikipedia.org/wiki/Vite_(software)>

<https://descubrecomunicacion.com/que-es-backend-y-frontend/>

<https://socket.io/docs/v4/>

<https://nodejs.org/es>

<https://es.wikipedia.org/wiki/Node.js>

<https://en.wikipedia.org/wiki/Express.js>

<https://www.sqlite.org/>