PROYECTO PROGRAMACIÓN SERVICIOS Y PROCESOS

SERVICIO DE MENSAJERÍA ONLINE MULTIHILO

Desarrollar un sistema cliente – servidor de mensajes en tiempo real que permita a los usuarios intercambiar mensajes.

Los mensajes deben entregarse inmediatamente si el receptor está conectado y, si no lo está, deben almacenarse en una base de datos hasta que el receptor se conecte nuevamente.

Funcionalidades principales

Autenticación de usuarios

Los usuarios han de iniciar sesión con su nombre de usuario y contraseña. La comunicación en este caso entre el cliente y el servidor ha de ser segura.

El servidor debe verificar las credenciales almacenadas en una base de datos (motor de base de datos libre).

Mensajería en tiempo real

Los usuarios conectados pueden enviarse mensajes en tiempo real.

El servidor gestionará la comunicación entre los clientes utilizando hilos independientes.

Mensajes pendientes

Si un receptor no está conectado, el servidor almacenará el mensaje en la base de datos.

Cuando el receptor se conecte, recibirá todos los mensajes pendientes.

Estados de conexión

El sistema debe mantener un estado online u offline para cada usuario.

Este estado debe actualizarse en la base de datos.

Historial de mensajes:

Los usuarios pueden consultar su historial de mensajes con otros usuarios.

Requisitos técnicos

Servidor

El servidor debe manejar múltiples conexiones de clientes simultáneamente utilizando hilos (multithreading).

El servidor se conectará a una base de datos, motor a libre elección, para gestionar usuarios y mensajes.

El servidor ha de estar programado en Java.

Cliente

La interfaz del cliente puede desarrollarse en cualquier tecnología.

El cliente debe conectarse al servidor mediante sockets.

Base de datos

Se deja a elección del estudiante la elección del motor de base de datos a utilizar.

Ha de almacenar la información de usuarios, mensajes y estados de conexión.

Componentes del sistema

Servidor Java

Gestionará conexiones de clientes mediante hilos.

Manejará la lógica de envío y recepción de mensajes.

Guardará y consultará datos en la base de datos

Base de datos

Se propone la siguiente estructura:

Usuarios, donde se almacena información de los usuarios.

Mensajes, donde se almacenan los mensajes enviados entre usuarios.

Sesiones, donde se rastrea los estados de conexión de los usuarios.

Cliente Java

Permite al usuario,

Iniciar sesión

Enviar mensajes a otros usuarios

Recibir mensajes en tiempo real

Consultar el historial de mensajes

Crear nuevos usuarios de chat

Diseño de Base de datos (propuesta)

Usuarios

IdUser (int, pk), identificador único del usuario

name (varchar), nombre del usuario

password (varchar), clave encriptada del usuario.

last_Online (datetime), última vez que el usuario estuvo conectado

Mensajes

idMessage (int pk), identificador único del mensaje

idTransmitter (int fk), identificador del emisor del mensaje.

idReceiver (int fk), identificador del receptor del mensaje

msgText (text), texto del mensaje

state (enumerado pendiente y entregado, 'pending', 'delivered')

timeStamp (datetime) fecha y hora del mensaje

Sesiones

idSession (int pk), identificador único de la sesión.

userId(int fk), identificador usuario de la sesión.

online(boolean), indicador si el usuario está conectado o no.

modified(datetime), última vez que se actualizó la sesión.

Flujo del sistema

1.- Conexión y autenticación

Los clientes se conectan al servidor mediante sockets.

El servidor autentica al usuario consultando la base de datos.

2.- Gestión de mensajes

Si ambos usuarios están conectados:

El servidor entrega el mensaje directamente al receptor mediante sockets.

Si el receptor no está conectado

El servidor guarda el mensaje en la base de datos con el estado "pendiente"

3.- Recepción de mensajes pendientes

Cuando un usuario inicia sesión, el servidor consulta la base de datos por mensajes con estado “pendiente” y los envía al cliente.

4.- Estados de conexión

Al conectarse, el estado del usuario en la base de datos se actualiza online.

Al desconectarse, se actualiza a offline.

Otros

1.- Implementar cifrado en las comunicaciones para proteger los datos.

2.- Confirmación de lectura de mensajes

- Añadir un estado de "leído" para los mensajes. Esto permite al emisor saber si el receptor ha visto el mensaje.

- Implementación:

- Agregar un nuevo estado al campo `state` de la tabla `Mensajes`, como `read` (leído).

- Cuando el cliente receptor muestra el mensaje al usuario, envía una confirmación al servidor.

- El servidor actualiza el estado del mensaje a `read` en la base de datos y notifica al emisor si está conectado.

3.- Grupos de chat

- Permitir que varios usuarios participen en un chat grupal.

- Implementación:

- Crear una nueva tabla `Grupos` y una tabla `MiembrosGrupo` que asocie usuarios con grupos.

- Los mensajes de grupo se almacenan en la misma tabla `Mensajes`, pero con un `idGroup` en lugar de `idReceiver`.

- El servidor envía los mensajes a todos los miembros conectados del grupo.

4.- Bloqueo de usuarios

- Permitir que los usuarios bloqueen a otros usuarios para no recibir mensajes no deseados.

- Implementación:

- Crear una tabla `UsuariosBloqueados` que almacene pares de usuarios (bloqueador y bloqueado).

- Antes de entregar un mensaje, el servidor verifica si el receptor ha bloqueado al emisor

5.-