Trabajo Práctico Final
Programación sobre Redes

Título del Proyecto : Chat distribuido seguro en Node.js

Objetivo General

El proyecto final consiste en desarrollar un chat distribuido seguro utilizando Node.js. Los estudiantes deberán implementar un sistema que permita la comunicación entre múltiples usuarios de manera concurrente, con mensajes enviados y recibidos en tiempo real, primero a través de TCP y luego migrando a WebSockets para permitir la interacción desde un navegador web.

La aplicación deberá incluir persistencia de mensajes, registrando en archivos de log todas las interacciones, así como información de conexión de los usuarios. Esto permitirá que los alumnos comprendan cómo manejar múltiples clientes, eventos concurrentes y cómo estructurar la información para que pueda ser monitoreada y auditada.

Además, se espera que los estudiantes incorporen mecanismos de seguridad en su chat. Esto incluye autenticación de usuarios mediante JSON Web Tokens (JWT) y cifrado de mensajes entre cliente y servidor, asegurando que la información sensible no pueda ser interceptada en la red. En la entrega final, se podrá extender a cifrado end-to-end, donde el servidor solo reenvía los mensajes cifrados sin poder leerlos.

El proyecto está diseñado para ser incremental: cada etapa del curso introduce un nuevo concepto o desafío. Desde la creación de un chat básico en consola hasta la implementación de un sistema web seguro y distribuido, los alumnos aplicarán prácticas de programación sobre redes, protocolos, seguridad y concurrencia. Al finalizar, se espera un sistema funcional, robusto y documentado, demostrando comprensión de los fundamentos teóricos y su aplicación práctica en un entorno real.

Entregas

Entrega Parcial

Objetivo: Implementar un chat multiusuario básico con logs.

Requisitos:

1. Servidor TCP en Node.js capaz de aceptar múltiples clientes concurrentes.

2. Cliente en consola que pueda conectarse y enviar/recibir mensajes.
3. Implementación de broadcast: los mensajes se reenvían a todos los clientes.
4. Persistencia inicial: guardar los mensajes en un archivo de log (fs).
5. Comandos básicos opcionales (/nick, /lista, /salir).

Checklist de entrega parcial:

- Código del servidor TCP.
- Código del cliente en consola.
- Logs de mensajes funcionando.
- README con instrucciones de ejecución.

Entrega Final

Objetivo: Transformar el chat parcial en un sistema **seguro y distribuido**.

Requisitos:

1. Migración a **WebSockets** para interacción web.
2. Cliente web simple en HTML + JS (multiusuario).
3. Autenticación de usuarios con **JWT**.
4. Cifrado de mensajes entre cliente y servidor (AES o end-to-end opcional).
5. Logging estructurado de conexiones, mensajes y errores (ej: winston).
6. Manejo de desconexiones y reconexiones.
7. Extensión opcional: replicación básica entre varios servidores o balanceo simulado.

Checklist de entrega final:

- Código del servidor WebSocket seguro.
- Cliente web funcional con chat multiusuario.
- Implementación de login/autenticación JWT.
- Mensajes cifrados correctamente.
- Logging estructurado (con errores y mensajes).
- Documentación de arquitectura.

Criterios de Evaluación

Aspecto	Peso
Funcionalidad básica (TCP/WebSocket)	30%
Concurrencia y manejo de clientes	20%
Seguridad y cifrado	20%
Persistencia y logging	10%
Distribución y robustez	10%
Documentación y presentación	10%

Extensiones opcionales

- Mensajes privados o grupos.
- Frontend avanzado con React o mobile.
- Contenerización con Docker.