🗂️ Aprendizaje sobre el Proyecto LChat

Programación

1. Elementos fundamentales del código

- Uso de variables y constantes en Node.js y JavaScript.
- Tipos: string, number, boolean, array, objeto.
- Ejemplo:

```
const PORT = 3000;
let usuariosConectados = [];
```

2. Estructuras de control

- Condicionales: if, else.
- Bucles: for, forEach.
- Ejemplo:

```
io.on('connection', socket => {
  if (usuariosConectados.length > 0) {
    // ...
  }
});
```

3. Control de excepciones y gestión de errores

- Uso de try-catch en Node.js para manejar errores de servidor y conexión.

4. Documentación del código

- Comentarios en JS y archivos markdown (README, aprendizaje, guion).

5. Paradigma aplicado

- Programación modular y estructurada.
- Separación de lógica en archivos: main.js (Electron), server.js (servidor), public/ (frontend).

6. Clases y objetos principales

- Objetos: usuario, mensaje, sala.
- No se usan clases, pero sí objetos literales y arrays.

7. Conceptos avanzados

- Uso de WebSockets (socket.io) para comunicación en tiempo real.
- WebRTC y simple-peer para videollamadas.
- Modularidad y reutilización de funciones.

8. Gestión de información y archivos

- No hay base de datos, la información se gestiona en memoria.
- Posibilidad de ampliar con almacenamiento persistente.

9. Estructuras de datos utilizadas

- Arrays y objetos para usuarios, mensajes y salas.

10. Técnicas avanzadas

- Integración de Electron para crear app de escritorio multiplataforma.
- Comunicación entre procesos (main y renderer en Electron).

Sistemas Informáticos

1. Características del hardware

- Desarrollo y pruebas en MacBook (macOS), compatible con Windows y Linux.

2. Sistema operativo

- Multiplataforma: macOS, Windows, Linux.

3. Configuración de redes

- Comunicación en red local o internet mediante WebSockets.

4. Copias de seguridad

- Uso de Git y GitHub para control de versiones y backups.

5. Integridad y seguridad de datos

- Validación de mensajes y usuarios en el servidor.
- Uso de HTTPS recomendado en producción.

6. Usuarios, permisos y accesos

- Gestión básica de usuarios conectados.
- No hay autenticación avanzada (puede ampliarse).

7. Documentación técnica

- Archivos markdown y comentarios en el código.

Entornos de Desarrollo

1. Entorno de desarrollo (IDE)

- Visual Studio Code con extensiones para JS, Node.js y Electron.

2. Automatización de tareas

- Scripts npm para iniciar servidor y app Electron.

3. Control de versiones

- Git y GitHub.

4. Refactorización

- Mejoras periódicas en la estructura y modularidad del código.

5. Documentación técnica

- README.md, aprendizaje.md, guion.md.

6. Diagramas

- Opcional: diagramas de flujo para la arquitectura de la app.

Bases de Datos

1. Sistema gestor

- No se usa base de datos, pero se puede ampliar con MongoDB o similar.

2. Modelo entidad-relación

- No aplica actualmente.

3. Funcionalidades avanzadas

- No aplica, pero se puede añadir persistencia de mensajes.

4. Protección y recuperación de datos

- Backups en GitHub.

Lenguajes de Marcas y Gestión de Información

1. Estructura de HTML

- Uso de etiquetas semánticas en public/index.html.

2. Tecnologías frontend

- HTML, CSS, JavaScript.

3. Interacción con el DOM

- JS para mostrar mensajes, usuarios y videollamadas.

4. Validación de HTML y CSS

- Validadores online y extensiones del IDE.

5. Conversión de datos (XML, JSON)

- Uso de JSON para mensajes y usuarios.

6. Integración con sistemas de gestión

- No aplica, pero se puede ampliar.

Proyecto Intermodular

1. Objetivo del software

- Permitir chat y videollamadas en tiempo real desde escritorio.

2. Necesidad o problema que soluciona

- Facilita la comunicación instantánea y videollamadas sin depender de apps externas.

3. Stack de tecnologías

- Node.js, Express, Socket.io, WebRTC, Electron.

4. Desarrollo por versiones

- v1: Chat básico.
- v2: Videollamadas y mejoras visuales.
- v3: Empaquetado multiplataforma.