Resumen Teórico del Proyecto

Este proyecto es una aplicación backend en Node.js que simula la estructura de una plataforma colaborativa tipo Slack, utilizando MongoDB como base de datos y Mongoose como ORM.

Estructura de Archivos

package.json

Contiene la configuración principal del proyecto, scripts de ejecución, dependencias y metadatos. Define comandos para desarrollo (`dev`, `nodemon-dev`) y producción (`start`).

documentacion.md

Explica conceptos clave de Node.js, el uso de `package.json`, la estructura del directorio `/src`, y recomendaciones sobre el uso de nodemon para desarrollo.

consigna.md

Lista los repositorios y métodos CRUD que deberían implementarse para cada entidad principal del sistema (usuarios, workspaces, miembros, canales, mensajes).

src/main.js

Archivo principal de la aplicación. Aquí se inicializa la conexión a MongoDB y se definen funciones para crear usuarios, workspaces, miembros y canales, utilizando los modelos y repositorios correspondientes.

src/error.js

Define una clase personalizada de error (`CustomError`) y funciones para manejar errores en el sistema, diferenciando entre errores de cliente y servidor.

src/config/configMongoDB.config.js

Configura y realiza la conexión a la base de datos MongoDB usando Mongoose. Maneja errores de conexión y exporta la función para ser utilizada en el archivo principal.

src/models/

Contiene los modelos de datos (esquemas de Mongoose) para cada entidad: - **User.model.js**: Define el esquema de usuario con campos como nombre, email, contraseña, fechas y estado. - **Workspace.model.js**: Esquema para espacios de trabajo, con nombre, imagen, fechas y estado. - **MemberWorkspace.model.js**: Relaciona usuarios con workspaces y define el rol de cada miembro. - **Channel.model.js**: Esquema para canales dentro de un workspace, con referencia al workspace, nombre y fechas.

src/repositories/

- **user.repository.js**: Implementa el patrón repositorio para usuarios, con métodos CRUD que interactúan con el modelo `User`.

Teoría y Conceptos Clave

- **Mongoose**: Permite definir esquemas y modelos para estructurar los datos en MongoDB y facilita las operaciones CRUD.
- **Patrón Repositorio**: Separa la lógica de acceso a datos de la lógica de negocio, facilitando el mantenimiento y escalabilidad.
- **Manejo de Errores**: Se implementan clases y funciones para capturar y diferenciar errores de cliente y servidor.
- **Scripts de Desarrollo**: Se recomienda usar `nodemon` para recargar la app automáticamente en desarrollo.
