Este proyecto es una aplicación que permite a los usuarios registrarse, compartir enlaces web interesantes, y votar en los enlaces compartidos por otros usuarios. La aplicación está construida en Node.js, y utiliza una base de datos MySQL para almacenar la información del usuario y los enlaces.

La aplicación consta de varias partes:

\*\*Middleware\*\*: Se utiliza para manejar la autenticación de los usuarios. El middleware "auth.js" verifica el token JWT proporcionado por el usuario en la cabecera de autorización de cada solicitud. Si el token es válido, la solicitud puede continuar al próximo middleware o controlador. Si el token es inválido, se detiene la solicitud y se envía un mensaje de error.

\*\*Modelos de Datos\*\*: Los modelos de datos ("userModel", "linkModel" y "voteModel") manejan todas las interacciones con la base de datos. Cada modelo tiene funciones para crear, buscar y eliminar datos. Los modelos también pueden tener funciones adicionales para operaciones específicas, como buscar un usuario por correo electrónico.

\*\*Controladores de Rutas\*\*: Los controladores de rutas manejan las solicitudes HTTP a diferentes endpoints de la API. Cada ruta se asocia con una función específica que procesa la solicitud y devuelve una respuesta. Por ejemplo, la ruta de registro maneja las solicitudes POST para registrar nuevos usuarios, mientras que la ruta de inicio de sesión maneja las solicitudes POST para iniciar sesión.

Para comprobar el correcto funcionamiento de la aplicación, utilizamos el software Postman. Postman es una herramienta que permite enviar solicitudes HTTP a una API y ver las respuestas. Con Postman, podemos probar cada endpoint de la API para asegurarnos de que está funcionando correctamente y devolviendo las respuestas esperadas. Además, Postman nos permite probar la autenticación y la autorización, enviando tokens JWT en las cabeceras de autorización de nuestras solicitudes.

Explicación paso a paso de la creación de la API

1. \*\*Configuración del entorno (.env)\*\*: Este archivo almacena las variables de entorno, como la cadena de conexión de la base de datos (DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME), el secreto del JWT para la autenticación (JWT\_SECRET), y el puerto donde se ejecutará la aplicación (PORT).

2. \*\*Index.js\*\*: Este es el archivo principal de la aplicación. Aquí se inicia el servidor de Express y se configuran los middleware necesarios.

3. \*\*Incorporación de dependencias\*\*: Se importan los módulos necesarios, como express, body-parser y dotenv.

4. \*\*Creación del servidor de Express\*\*: Se crea una nueva instancia de Express para manejar las solicitudes HTTP.

5. \*\*Configuración de body-parser\*\*: Este middleware se utiliza para analizar el cuerpo de las solicitudes HTTP entrantes en un formato utilizable. En este caso, se configura para analizar tanto JSON como datos de formularios URL codificados.

6. \*\*Importación de rutas\*\*: Se importan las rutas definidas en otros archivos para manejar diferentes endpoints de la API.

7. \*\*Aplicación de rutas\*\*: Las rutas importadas se aplican a la aplicación Express.

8. \*\*Iniciación del servidor\*\*: La aplicación comienza a escuchar en el puerto especificado, listo para aceptar solicitudes entrantes.

9. \*\*Middleware de Autenticación (auth.js)\*\*: Este módulo verifica si las solicitudes al servidor incluyen un token JWT válido en la cabecera de autorización. Si el token es válido, se extraen los datos del usuario y se adjuntan a la solicitud antes de pasar al siguiente middleware.

10. \*\*Extracción del token\*\*: Se extrae el token de la cabecera de autorización.

11. \*\*Verificación del token\*\*: Se utiliza la biblioteca jsonwebtoken para verificar la firma del token.

12. \*\*Asociación del usuario a la solicitud\*\*: Si el token es válido, se extraen los datos del usuario y se buscan en la base de datos. Los datos del usuario se adjuntan a la solicitud.

13. \*\*Control de errores\*\*: Si el token es inválido o no se encuentra el usuario, se devuelve una respuesta de error adecuada.

14. \*\*Modelo de Usuario (userModel.js)\*\*: Este módulo define las operaciones de la base de datos relacionadas con los usuarios.

15. \*\*Creación de usuarios\*\*: Se define una función para insertar un nuevo usuario en la base de datos.

16. \*\*Búsqueda de usuarios por correo electrónico\*\*: Se define una función para buscar un usuario en la base de datos por correo electrónico.

17. \*\*Búsqueda de usuarios por ID\*\*: Se define una función para buscar un usuario en la base de datos por ID.

18. \*\*Actualización de usuarios\*\*: Se define una función para actualizar los datos de un usuario existente en la base de datos.

19. \*\*Modelo de Enlaces (linkModel.js)\*\*: Este módulo define las operaciones de la base de datos relacionadas con los enlaces.

20. \*\*Creación de enlaces\*\*: Se define una función para insertar un nuevo enlace en la base de datos.

21. \*\*Búsqueda de enlaces por ID\*\*: Se define una función para buscar un enlace en la base de datos por ID.

22. \*\*Búsqueda de enlaces por fecha\*\*: Se define una función para buscar todos los enlaces creados en una fecha específica.

24. \*\*Eliminación de enlaces\*\*: Se define una función para eliminar un enlace existente en la base de datos.

25. \*\*Modelo de Votos (voteModel.js)\*\*: Este módulo define las operaciones de la base de datos relacionadas con los votos.

26. \*\*Creación de votos\*\*: Se define una función para insertar un nuevo voto en la base de datos.

27. \*\*Búsqueda de votos\*\*: Se define una función para buscar un voto en la base de datos por ID de enlace y ID de usuario.

28. \*\*Eliminación de votos\*\*: Se define una función para eliminar un voto existente en la base de datos.

29. \*\*Funciones de Ruta\*\*: Las funciones de ruta manejan las solicitudes HTTP a diferentes endpoints. Cada ruta se asocia con una función específica dependiendo del tipo de solicitud (GET, POST, DELETE, etc.) y del endpoint.

30. \*\*Ruta de Registro\*\*: Esta ruta maneja las solicitudes POST para registrar un nuevo usuario. Toma los datos del cuerpo de la solicitud, los valida y luego los pasa al modelo de usuario para crear un nuevo usuario en la base de datos.

31. \*\*Ruta de Inicio de Sesión\*\*: Esta ruta maneja las solicitudes POST para iniciar sesión. Toma los datos del cuerpo de la solicitud, los valida, verifica que el usuario exista en la base de datos y que la contraseña coincida. Si todo está bien, genera un token de JWT y lo envía de vuelta al cliente.

32. \*\*Ruta para Crear un Enlace\*\*: Esta ruta maneja las solicitudes POST para crear un nuevo enlace. Toma los datos del cuerpo de la solicitud, los valida y luego los pasa al modelo de enlace para crear un nuevo enlace en la base de datos.

33. \*\*Ruta para Obtener Enlaces\*\*: Esta ruta maneja las solicitudes GET para obtener todos los enlaces para un día específico. Toma la fecha de la solicitud y la pasa al modelo de enlace para recuperar todos los enlaces creados en esa fecha.

34. \*\*Ruta para Eliminar un Enlace\*\*: Esta ruta maneja las solicitudes DELETE para eliminar un enlace existente. Toma el ID de la solicitud, verifica que el enlace exista y que el usuario tenga permiso para eliminarlo, y luego lo pasa al modelo de enlace para eliminar el enlace de la base de datos.

35. \*\*Ruta para Votar un Enlace\*\*: Esta ruta maneja las solicitudes POST para votar un enlace existente. Toma los datos de la solicitud, los valida y luego los pasa al modelo de votos para crear un nuevo voto en la base de datos.

36. \*\*Ruta para Editar un Perfil de Usuario\*\*: Esta ruta maneja las solicitudes PUT para actualizar los datos de un usuario existente. Toma los datos del cuerpo de la solicitud, los valida y luego los pasa al modelo de usuario para actualizar el usuario en la base de datos.

Eso es esencialmente cómo funciona la aplicación. Cada pieza interactúa con las otras para permitir a los usuarios registrarse, iniciar sesión, compartir enlaces, votar enlaces y editar sus propios datos.