Explicación del código. Seguridad basada en tokens JWT.

El archivo configura una aplicación web con Express que utiliza SQLite como base de datos local y tokens web JSON para controlar el acceso. Primero se cargan las dependencias necesarias: Express para crear el servidor, sqlite3 para administrar la base de datos, jsonwebtoken para firmar y verificar los tokens y paths para crear las rutas de los archivos. Se crea una instancia Express, se establece el puerto en el que escucha la aplicación y se define una clave secreta que se utilizará para firmar tokens.

La aplicación habilita un middleware que convierte automáticamente los cuerpos de solicitud en objetos JSON, de modo que los datos enviados por los clientes puedan leerse a través de req.body. Luego se abre o crea un archivo de base de datos SQLite llamado clientes.db en la misma carpeta que el proyecto y se genera un informe de consola que indica si la conexión fue exitosa o ocurrió un error.

En /login se implementa una ruta POST que recibe un nombre de usuario y contraseña de la solicitud. Si faltan estos datos, se devuelve un error de cliente. Cuando las credenciales coinciden con los valores esperados, se genera un token firmado con la clave secreta y con un tiempo de expiración de una hora y se envía al cliente; Si no coinciden, se devuelve un error de autenticación.

Para proteger las rutas que requieren autenticación, se define la función verifyToken. Esta función extrae el encabezado de Autorización, espera la palabra Bearer seguida de un token y, si no encuentra uno, rechaza la solicitud. Una vez que el token está presente, intenta validarlo con la misma clave secreta. Si la verificación tiene éxito, el nombre de usuario se agrega al objeto req para que las funciones posteriores puedan usarlo; Si no tiene éxito, la solicitud se rechaza con un error de autorización.

En /protected se expone una ruta de texto que solo responde si la solicitud se pasa a través de verifyToken. Después de iniciar sesión correctamente, se devuelve un mensaje de inicio de sesión permitido junto con el nombre de usuario extraído del token.

Las siguientes son las operaciones CRUD declaradas para la tabla de clientes. Todos ellos utilizan Verify Token, por lo que solo un usuario con un token válido puede crear, leer, actualizar o eliminar registros. Para crear un cliente se acepta una persona con nombre, correo electrónico, número de teléfono y dirección; Estos valores se insertan en la tabla y se devuelve el identificador asignado. Para obtener todos los clientes, se ejecuta una consulta que devuelve todas las filas de la tabla. Para obtener un cliente específico, se ejecuta una consulta con un filtro por ID y si no existe, se devuelve un mensaje de error "no encontrado". Para actualizar un cliente, si se reciben nuevos datos, se realiza una operación UPDATE y, si la consulta no afectó ninguna fila, informa que el cliente no existe. Para eliminar un cliente, se realiza una operación DELETE con el id y se informa al usuario si la fila no estaba presente.

Finalmente se inicia el servidor HTTP y la consola muestra la URL donde está disponible la aplicación. El resultado es un servicio REST que le permite autenticarse, recibir un token y usar ese token para operar en registros de la tabla de clientes, al tiempo que protege todas las acciones contra el acceso no autorizado.