Middleware

import passport from 'passport';

Esta línea importa el módulo passport, que es una biblioteca de autenticación para Node.js. Se utiliza para manejar la autenticación de usuarios de manera sencilla.

export function authMiddleware(req, res, next) {

Esta línea define y exporta una función llamada authMiddleware. Esta función es un middleware que se utilizará en una aplicación Express para manejar la autenticación de usuarios. Los parámetros req, res y next son objetos de la solicitud, la respuesta y una función que llama al siguiente middleware en la cadena respectivamente.

passport.authenticate('jwt', { session: false }, (err, user, info) => {

Aquí se llama a passport.authenticate con la estrategia 'jwt' (JSON Web Token) para autenticar la solicitud. La opción { session: false } indica que no se debe crear una sesión de usuario (es decir, no se usa sesión de servidor, útil en aplicaciones sin estado). La función de callback (err, user, info) se ejecuta después de que passport intenta autenticar al usuario. Recibe tres parámetros: err (errores de autenticación, si los hay), user (el usuario autenticado, si la autenticación fue exitosa) y info (información adicional sobre la autenticación).

if (err) {

return next(err);

}

Si hubo un error durante la autenticación (err no es null), se pasa el error al siguiente middleware usando next(err). Esto permite manejar el error adecuadamente en la aplicación.

res.locals.isAuthenticated = !!user;

Esta línea establece una variable isAuthenticated en res.locals, la cual indica si el usuario está autenticado (true si user es verdadero, false si no). res.locals es un objeto que contiene variables locales para la solicitud y la respuesta, y se puede utilizar para pasar datos a la vista.

req.user = user.user || null;

Aquí se asigna el objeto de usuario autenticado a req.user. Si user.user es válido, req.user se establece en ese valor; de lo contrario, se establece en null. Esto asegura que req.user siempre exista y pueda ser utilizado más adelante en la cadena de middlewares o en las rutas.

next();

Finalmente, se llama a next() para pasar el control al siguiente middleware en la cadena. Esto permite que la solicitud continúe siendo procesada por otros middlewares o rutas.

})(req, res, next);

Esta es la ejecución inmediata de la función passport.authenticate. La autenticación se lleva a cabo inmediatamente con los parámetros req, res y next pasados desde authMiddleware.

En resumen, este middleware utiliza Passport para autenticar a los usuarios mediante JWT. Si la autenticación es exitosa, establece información sobre la autenticación en res.locals y req.user, y luego pasa el control al siguiente middleware en la cadena. Si hay un error, lo pasa al siguiente middleware de error.

porque es !!user en res.locals.isAuthenticated = !!user;

El uso de !!user en la línea res.locals.isAuthenticated = !!user; se emplea para convertir el valor de user en un booleano. Aquí te explico paso a paso:

Valor de user: Después de la autenticación, user puede tener varios valores posibles:

Un objeto de usuario válido si la autenticación fue exitosa.

null o undefined si la autenticación falló o no se proporcionó un token.

Primer ! (negación): Aplicar el operador ! a user convierte cualquier valor "truthy" en false y cualquier valor "falsy" en true. Por ejemplo:

!true se convierte en false

!false se convierte en true

!null se convierte en true

!{} (un objeto no vacío) se convierte en false

Segundo ! (doble negación): Aplicar un segundo ! al resultado anterior convierte el valor nuevamente a su forma booleana correcta. Por ejemplo:

!!true se mantiene true

!!false se mantiene false

!!null se convierte en false

!!{} se convierte en true

Por lo tanto, !!user garantiza que user se convierta en un valor booleano:

true si user es un objeto (autenticación exitosa).

false si user es null o undefined (autenticación fallida).

Así, res.locals.isAuthenticated = !!user; se asegura de que isAuthenticated sea siempre un booleano que refleja si el usuario está autenticado o no. Esta práctica de usar !! es común en JavaScript para convertir valores a booleanos de forma explícita.