Documentación de tecnologías utilizadas

Para este test de chat, he realizado un Stack MERN de JavaScript usando MongoDB como sistema de base de datos, Express como framework para nodejs, React para las interfaces de usuario y NodeJs entorno de ejecución (servidor), como tecnologías bases para el desarrollo de la misma.

MongoDB → https://www.mongodb.com/
Express → https://expressjs.com/es/4x/api.html
React → https://reactjs.org/
Node → https://nodejs.org/en/

También me he apoyado en el uso de paquetes externos para ayudar en ciertas funcionalidades.

Para la parte del **Servidor(Backend)**

Dependencias:
Cors → https://www.npmjs.com/package/cors
Dotenv → https://www.npmjs.com/package/dotenv
Express → https://www.npmjs.com/package/express
Jsonwebtoken → https://www.npmjs.com/package/jsonwebtoken
Mongoose → https://www.npmjs.com/package/mongoose
Passport → https://www.npmjs.com/package/passport
Passport-local → https://www.npmjs.com/package/passport-local
Passport-jwt → https://www.npmjs.com/package/passport-jwt
Socket.io → https://www.npmjs.com/package/socket.io

Dependencias de desarrollo:
Concurrently → https://www.npmjs.com/package/concurrently
Morgan → https://www.npmjs.com/package/morgan
Nodemon → https://www.npmjs.com/package/nodemon

Para la parte del **Cliente(Frontend)**

Dependencias:
Bootstrap → https://getbootstrap.com/
Momentjs → https://momentjs.com/
ReactRouterDOM → https://www.npmjs.com/package/react-router-dom
ReactNotifications → https://www.npmjs.com/package/react-notifications
ReactStrap → https://reactstrap.github.io/
Redux → https://es.redux.js.org/
Socket.io-client → https://www.npmjs.com/package/socket.io-client

Resumen del uso de cada dependencia

1. **Cors:** controlar las peticiones que llegan al servidor de nodejs.
2. **Dotenv:** manejar las variables de entorno en el servidor de nodejs.
3. **Express:** framework utilizado para manejar las Apis del servidor de nodejs.
4. **Jsonwebtoken:** generar cadena de caracteres encryptados para una sesión de usuario.
5. **Passport:** permitir poder usar autenticación en la aplicación.
6. **Passport-local:** crear autenticación en la aplicación de forma local.
7. **Passport-jwt:** autenticar la cadena de caracteres que anteriormente fue encryptada usando jsonwebtoken y regresar la información correspondiente al usuario si es valido.
8. **Socket.io:** poder escuchar eventos en el servidor y emitir respuestas a los clientes conectados.
9. **Concurrently:** permitir poder ejecutar 2 servidores al mismo tiempo uno para React y otro para Nodejs, utilizando un mismo comando de arranque.
10. **Morgan:** ver por consola cuando un usuario realiza una petición al servidor de nodejs.
11. **Nodemon:** refrescar cada cambio que se realiza en el servidor de nodejs.
12. **Bootstrap:** permitir usar estilos de css de este framework para las interfaces de usuario.
13. **Momentjs:** aplicar un formato a las fechas que se obtienen para los mensajes del chat.
14. **ReactRouterDOM:** controlar las rutas de las interfaces de usuario.
15. **ReactNotifications:** mostrar alertas en las interfaces de usuario.
16. **ReactStrap:** reducir el código de las interfaces de usuario para el uso de bootstrap.
17. **Redux:** administrar los estados de la aplicación.
18. **Socket.io-client:** permitir conectar un cliente con el servidor para escuchar o emitir eventos en el chat.

La aplicación se encuentra en los siguientes enlaces:

https://github.com/wilsalas/TestCondorLabs

https://chatcondorlabs.herokuapp.com/