

# Clean Code

Para la generación de informes de clean code para nuestro proyecto decidimos usar ESLint debido a que es una de las herramientas de análisis estático más usadas globalmente para JavaScript y en nuestro caso la mayor parte del código realizado es con este lenguaje de programación.

```
o [icon] [icon] [icon] ~/V/chess_with_sockets/Chess_With_Sockets [icon] [icon] main !2 [icon] npm install -D eslint [icon]
```

Lo primero que se hace es instalar y configurar ESLint en el proyecto.

```
✓ How would you like to use ESLint? · problems
✓ What type of modules does your project use? · esm
✓ Which framework does your project use? · none
✓ Does your project use TypeScript? · javascript
✓ Where does your code run? · browser
The config that you've selected requires the following dependencies:

eslint, globals, @eslint/js
✓ Would you like to install them now? · No / Yes
✓ Which package manager do you want to use? · npm
🔄 Installing...
```

Después se ejecuta por carpetas, en primer lugar se corrió la carpeta Server, donde se configuran las rutas y sockets del proyecto.

```
> npx eslint server

/home/roger/VisualStudioCode/chess_with_sockets/Chess_With_Sockets/server/routes/routes.js
 13:14  error  'games' is not defined  no-undef

/home/roger/VisualStudioCode/chess_with_sockets/Chess_With_Sockets/server/server.js
 18:1   error  'games' is not defined  no-undef
 26:26  error  '.__dirname' is not defined  no-undef
 32:28  error  '.__dirname' is not defined  no-undef
 33:45  error  '.__dirname' is not defined  no-undef

/home/roger/VisualStudioCode/chess_with_sockets/Chess_With_Sockets/server/sockets/io.js
 19:18  error  'games' is not defined  no-undef
 20:17  error  'games' is not defined  no-undef
 32:24  error  'games' is not defined  no-undef

✖ 8 problems (8 errors, 0 warnings)
```

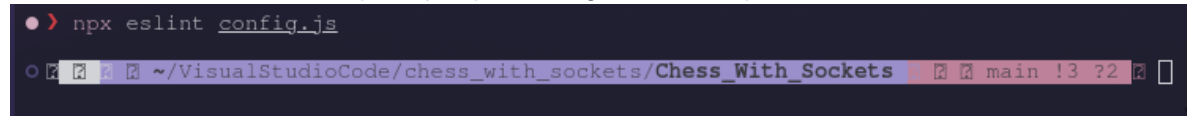
Aquí marca varios errores que no debería marcar, pero tiene sentido que lo haga, en primer lugar debido a que ninguna de estas variables están declaradas, pero esto ocurre porque se necesitan en la lógica del código, por ejemplo “games” es una lista de juegos en base al código, al crear un juego con “x” código, la variable games se estará declarado con un elemento correspondiente al código de juego, de la misma manera algo parecido ocurre con “\_dirname”.

Para el código principal del proyecto ocurre algo parecido, aquí se definen varios variables que se usan en la lógica del programa, como “ChessBoard” para crear el tablero de ajedrez, pero este se usa e importa en base a otro script, lo mismo ocurre con las otras variables.

```
/home/roger/VisualStudioCode/chess_with_sockets/Chess_With_Sockets/front/public/js/index.js
 4:16  error  'Chess' is not defined  no-undef
 5:15  error  '$' is not defined      no-undef
 6:12  error  '$' is not defined      no-undef
 9:38  error  'position' is defined but never used  no-unused-vars
 9:48  error  'orientation' is defined but never used  no-unused-vars
15:10  error  'playerColor' is not defined  no-undef
15:68  error  'playerColor' is not defined  no-undef
38:5   error  'socket' is not defined      no-undef
43:1   error  'socket' is not defined      no-undef
104:9  error  'Chessboard' is not defined   no-undef
105:5  error  'playerColor' is not defined  no-undef
113:5  error  'socket' is not defined      no-undef
118:1  error  'socket' is not defined      no-undef
123:1  error  'socket' is not defined      no-undef

✖ 14 problems (14 errors, 0 warnings)
```

Los demás archivos como por ejemplo configuración de puertos funcionan sin errores:

A screenshot of a terminal window with a dark background. The first line shows a command prompt with a red dot icon followed by the command `npx eslint config.js`. The second line shows the terminal's path `~/VisualStudioCode/chess_with_sockets/Chess_With_Sockets` and the status of the current branch `main !3 ?2`.

```
npx eslint config.js
~/VisualStudioCode/chess_with_sockets/Chess_With_Sockets main !3 ?2
```

## Opiniones del código:

- **Mateo Becerra:** Al principio me costó un poco agarrarle el ritmo a diferentes estilos de código, sin embargo, creo que el código muestra también cómo nos hemos repartido el trabajo y cómo colaboramos. Obviamente, cada uno tiene su manera de escribir, pero seguimos un estándar que hace que todo se vea más ordenado.
- **Thomás Enciso:** Los roles de cada miembro se entienden, la organización interna del código es legible y su propósito se entiende por sus nombres (de archivos o funciones) o comentarios en algunos casos, el uso de frameworks y su orden dentro del repositorio es lo que me confunde más, a veces me pierdo entre los archivos buscando algún dato o función.
- **Roger Bonilla:** Basándome en el código escrito por mis compañeros, lo que más se me ha dificultado es comprender ciertas partes debido al uso inadecuado de nombres para funciones y variables. Además, el uso de un lenguaje y frameworks nuevos ha representado un desafío adicional para entender algunos fragmentos del código. Sin embargo, cada uno ha desempeñado su rol específico de manera adecuada, estructurando su código y funcionalidades por lo general correctamente.
- **Julián Cadena:** La organización de los roles es clara, y las tareas de cada miembro del equipo están bien definidas. El código por lo general es fácil de leer, sin embargo, en algunas ocasiones, la documentación proporcionada por algunos integrantes no es suficientemente clara o comprensible.