UNIVERSIDAD PRIVADA DEL VALLE

FACULTAD DE SISTEMAS Y TECNOLOGÍA INFORMÁTICA

DEPARTAMENTO DE INGENIERIA EN SISTEMAS INFORMÁTICOS

Métricas Web Tuberculosis

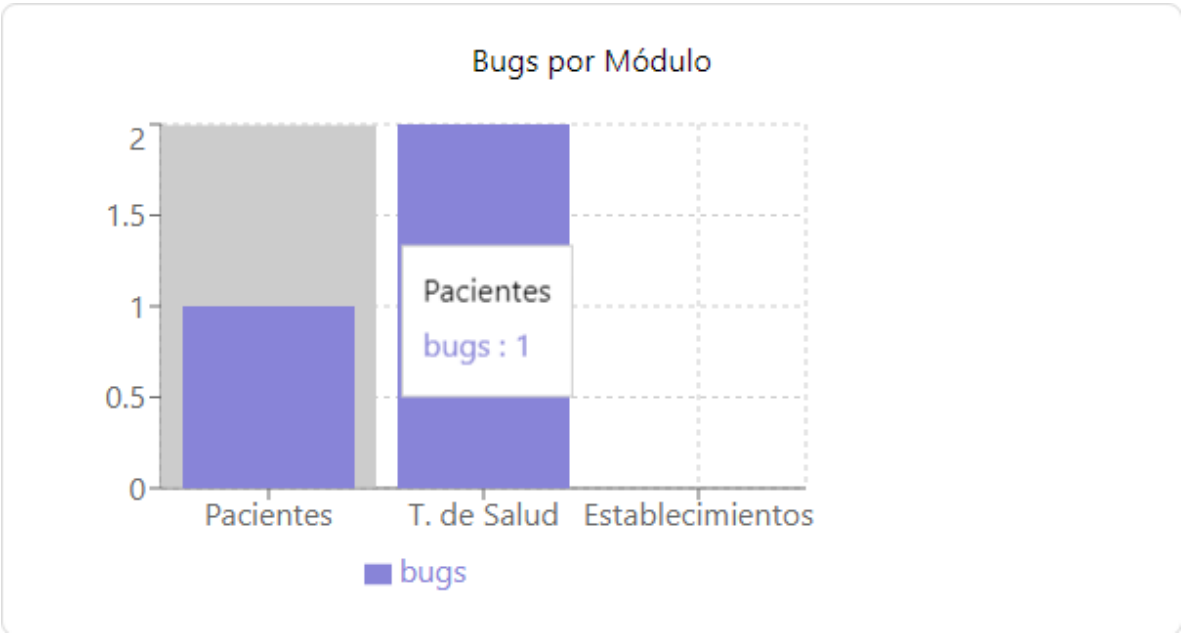
INTEGRANTES: Luis David Ancieta Sejas
Diana Abigail Arostegui Ramos
Erik Cuba Rodriguez

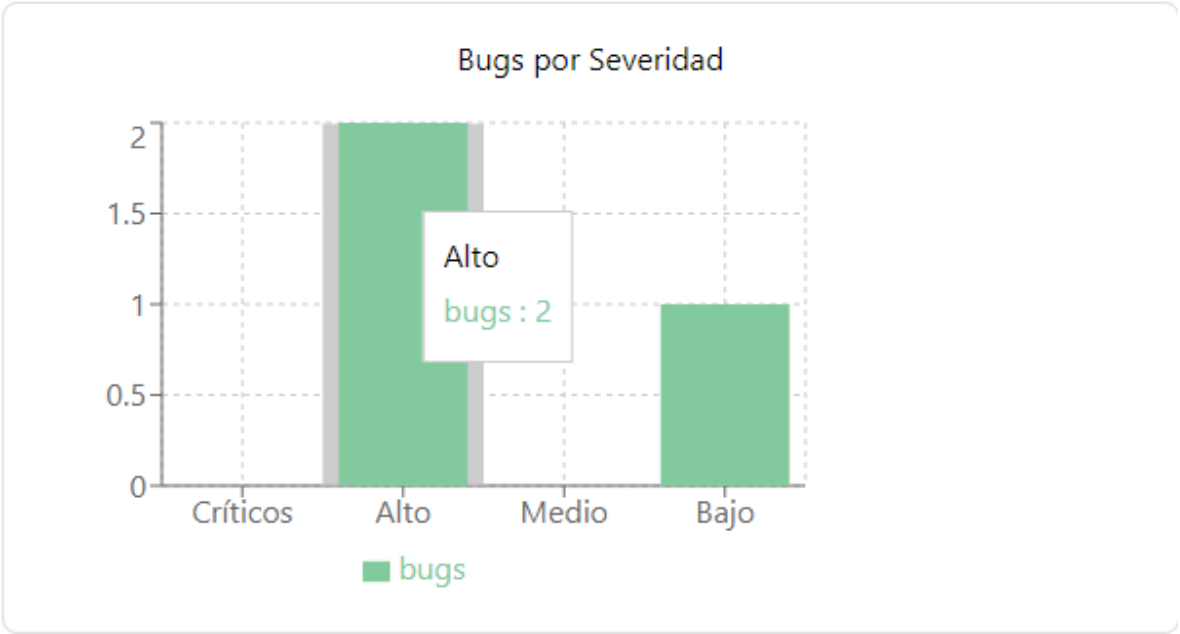
DOCENTE: Ing. Nibeth Mena Mamani

SEMESTRE: II-2024

Cochabamba – Bolivia

Bugs:





METRICAS DE CALIDAD DE CODIGO CON LA HERRAMIENTA ESLINT

1. **Analizar el Proyecto**

Usamos el siguiente comando: **npx eslint .**

```
PS C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB> npx eslint .
>>
Warning: React version not specified in eslint-plugin-react settings. See https://github.com/jsx-eslint/eslint-plugin-react#configuration .

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\components\Footer.js
  7:44  error  Unknown property 'ap-type' found  react/no-unknown-property

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\components\Layout.js
  6:19  error  'children' is missing in props validation  react/prop-types

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\components\LayoutPersonalSalud.js
  6:19  error  'children' is missing in props validation  react/prop-types

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\components\Signin.js
   6:19  error  'usuario' is missing in props validation            react/prop-types
   6:28  error  'setUsuario' is missing in props validation         react/prop-types
   6:40  error  'contrasenia' is missing in props validation        react/prop-types
   6:53  error  'setContrasenia' is missing in props validation     react/prop-types
   6:69  error  'handleLogin' is missing in props validation        react/prop-types
  15:11  error  'handleKeyPress' is assigned a value but never used  @typescript-eslint/no-unused-vars

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\ActualizarPaciente.js
  95:13  error  'response' is defined but never used  @typescript-eslint/no-unused-vars

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\AñadirPaciente.js
  87:13  error  'response' is defined but never used  @typescript-eslint/no-unused-vars

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\RegistrarPersonalSalud.js
  55:13  error  'response' is assigned a value but never used  @typescript-eslint/no-unused-vars
```

```
  87:13  error  'response' is defined but never used  @typescript-eslint/no-unused-vars

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\RegistrarPersonalSalud.js
  55:13  error  'response' is assigned a value but never used  @typescript-eslint/no-unused-vars

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\server\Conection.js
    1:17  error  A `require()` style import is forbidden    @typescript-eslint/no-require-imports
    1:17  error  'require' is not defined                   no-undef
    2:15  error  A `require()` style import is forbidden    @typescript-eslint/no-require-imports
    2:15  error  'require' is not defined                   no-undef
    3:14  error  A `require()` style import is forbidden    @typescript-eslint/no-require-imports
    3:14  error  'require' is not defined                   no-undef
    4:16  error  A `require()` style import is forbidden    @typescript-eslint/no-require-imports
    4:16  error  'require' is not defined                   no-undef
    5:7   error  'path' is assigned a value but never used  @typescript-eslint/no-unused-vars
    5:14  error  A `require()` style import is forbidden    @typescript-eslint/no-require-imports
    5:14  error  'require' is not defined                   no-undef
   11:7   error  'PORT' is assigned a value but never used  @typescript-eslint/no-unused-vars
  272:17  error  'result' is defined but never used         @typescript-eslint/no-unused-vars
  369:11  error  'result' is defined but never used         @typescript-eslint/no-unused-vars
  382:53  error  'result' is defined but never used         @typescript-eslint/no-unused-vars
  768:35  error  'result' is defined but never used         @typescript-eslint/no-unused-vars
  798:12  error  'error' is defined but never used          @typescript-eslint/no-unused-vars

✖ 29 problems (29 errors, 0 warnings)
```

Resultados del Análisis:

- **Total de Problemas Detectados:** 29 errores, 0 advertencias

Detalle de Errores por Categoría:

1. Propiedades Desconocidas en Componentes React (react/no-unknown-property)

- **Archivo:** Footer.js
  - Línea 7:44 → Propiedad desconocida 'ap-type' encontrada.
- **Impacto:** Estas propiedades desconocidas pueden provocar comportamientos inesperados en la UI.
- **Acción Recomendada:** Corregir o eliminar propiedades inválidas.

2. Validación de Props Faltantes (react/prop-types)

- **Archivos:**
  - Layout.js (Línea 6:19): children falta en la validación de props.
  - LayoutPersonalSalud.js (Línea 6:19): children falta en la validación de props.
  - Signin.js:
    - Línea 6:19: usuario falta en la validación de props.
    - Línea 6:28: setUsuario falta en la validación de props.
    - Línea 6:40: contrasenia falta en la validación de props.
    - Línea 6:53: setContrasenia falta en la validación de props.
    - Línea 6:69: handleLogin falta en la validación de props.
- **Impacto:** Falta de validación de props puede llevar a errores en tiempo de ejecución.
- **Acción Recomendada:** Implementar PropTypes para validar correctamente las propiedades.

3. Variables No Usadas (@typescript-eslint/no-unused-vars)

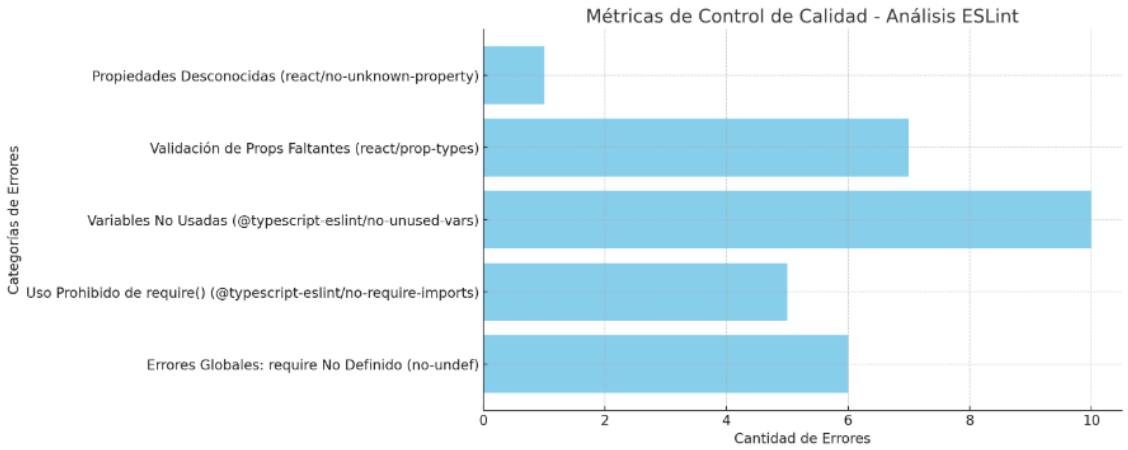
- **Archivos:**
  - ActualizarPaciente.js (Línea 95:13): response definido pero no utilizado.
  - AñadirPaciente.js (Línea 87:13): response definido pero no utilizado.
  - RegistrarPersonalSalud.js (Línea 55:13): response asignado pero no utilizado.
  - Conection.js:
    - Varias líneas con variables como result, path, y error definidas pero no utilizadas.
- **Impacto:** Mantener variables no utilizadas aumenta la complejidad del código y reduce su claridad.
- **Acción Recomendada:** Eliminar variables no utilizadas o usarlas si son necesarias.

- **4. Uso Prohibido de require() en TypeScript (@typescript-eslint/no-require-imports)**
- **Archivo:** Conection.js
  - Varias líneas utilizan require() en lugar de import.
- **Impacto:** La mezcla de estilos puede provocar problemas de compatibilidad y dificultar la migración hacia ES Modules.
- **Acción Recomendada:** Cambiar require() por import para mantener consistencia.

5. Errores Globales: require No Definido (no-undef)

- **Archivo:** Conection.js
  - Línea 1-5: require no está definido.
- **Impacto:** Estos errores ocurren porque el entorno Node.js no está configurado correctamente en ESLint.

Grafica:



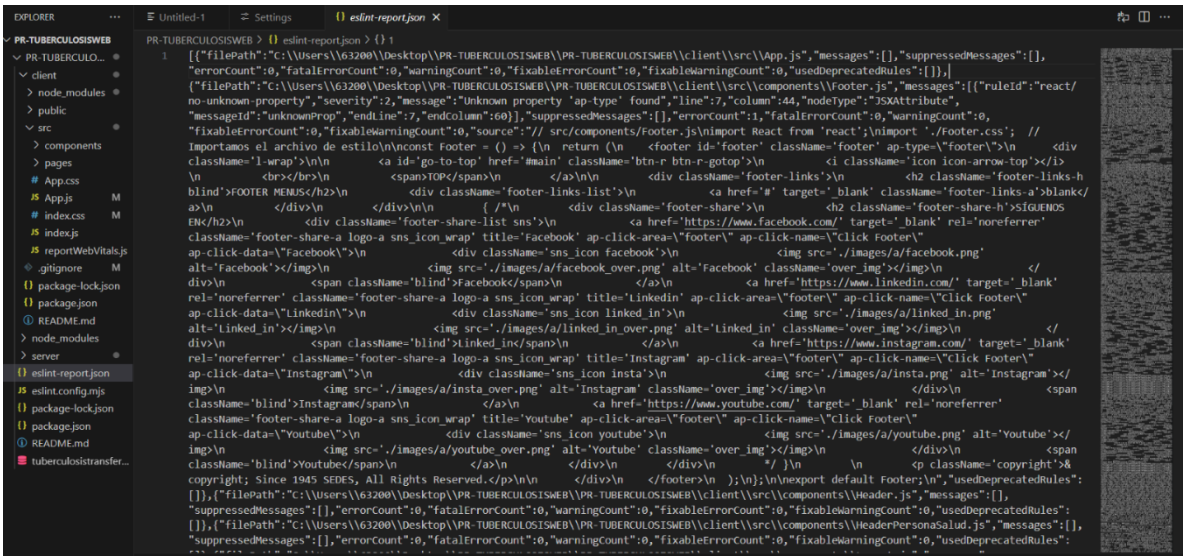
Conclusión:
El análisis realizado con ESLint detectó errores significativos que afectan la calidad del código en términos de validación de propiedades, uso de variables, y consistencia en la importación de módulos. Corregir estos problemas mejorará la legibilidad, mantenibilidad y estabilidad del proyecto.
Acciones Sugeridas:

1. Validar props en todos los componentes React.
2. Eliminar variables no utilizadas.
3. Migrar de require() a import.
4. Configurar correctamente el entorno de ESLint para Node.js.

2. Informe Detallado en Formato JSON
Usamos el siguiente comando**: npx eslint . --format json --output-file eslint-report.json**
Descripción del Reporte
El análisis del proyecto mediante ESLint generó un reporte en formato **JSON** que contiene un **listado detallado de errores y advertencias**. Este archivo es útil para identificar problemas específicos, además de ser compatible con herramientas externas para un análisis avanzado.
Métricas Principales del Reporte:

- **Listado detallado de errores y advertencias:**
  - Información exhaustiva por archivo, línea, columna y tipo de problema.
  - Clasificación de los problemas según su gravedad: errores y advertencias.
- **Información estructurada:**
  - Facilita la integración con herramientas externas.
  - Permite personalizar reportes y realizar análisis en profundidad.

Incorporación Visual del Reporte:
Dado que el archivo JSON es extenso, se presentan a continuación **capturas representativas**.



3. Informe Visual en Formato HTML

Aquí usamos dos comandos el primero para instalar el formateador HTML: **npm install eslint-formatter-html --save-dev** y el segundo para generar el informe **HTML npx eslint . --format html --output-file eslint-report.html**

Descripción del Reporte Visual

Se generó un **reporte visual en formato HTML** utilizando ESLint, que presenta de manera clara y organizada:

- **Errores y advertencias:** por archivo, regla y severidad.
- **Facilidad de navegación:** estructura interactiva para explorar cada detalle.

Ventajas del Reporte HTML

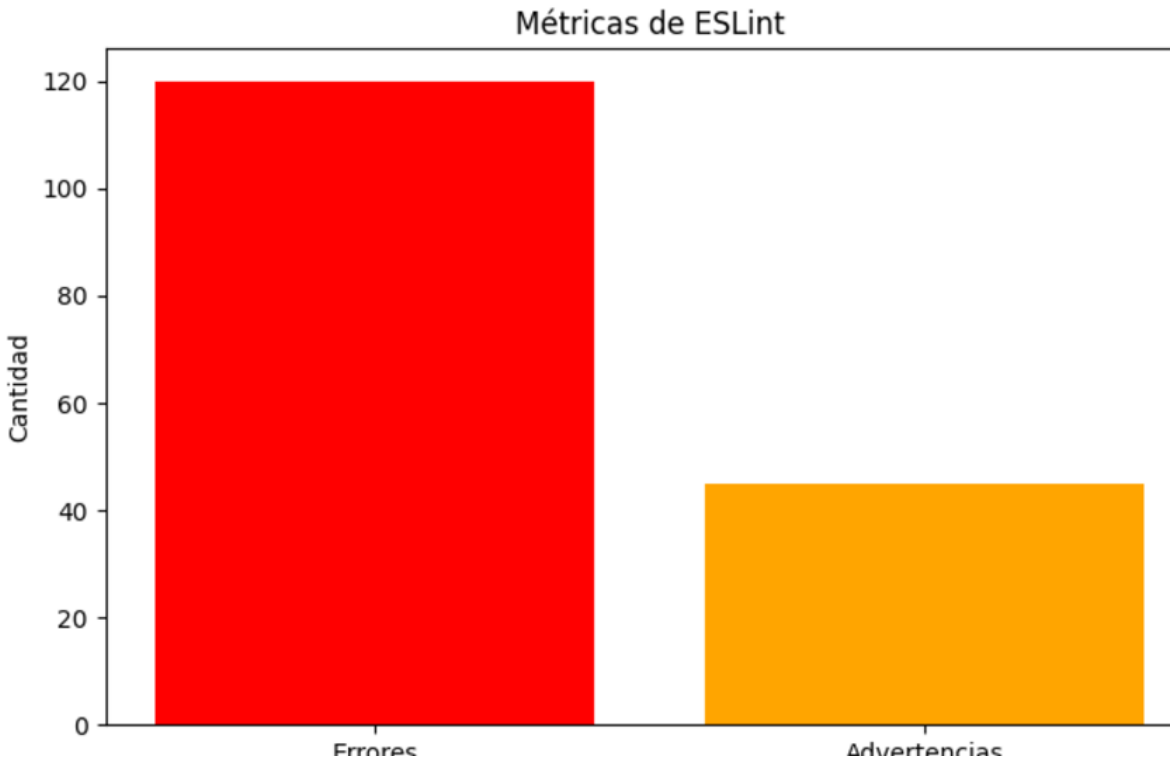
1. **Interactividad:** Permite explorar visualmente los problemas detectados.
2. **Eficiencia en la revisión:** Mejora la comprensión del estado del código.
3. **Compatibilidad:** Puede ser visualizado en cualquier navegador.

Capturas del Reporte HTML

PR-TUBERCULOSISWEB > eslint-report.html > html

```
<!doctype html>
<html lang="en">

<head>
    <title>Eslint Report</title>
    <meta charset="UTF-8" />
    <base target="_blank" />
    <link rel="icon" href="https://eslint.org/favicon.ico" />
    <script src="https://unpkg.com/react@18.2.0/umd/react.production.min.js"></script>
    <script src="https://unpkg.com/react-dom@18.2.0/umd/react-dom.production.min.js"></script>
    <script src="https://unpkg.com/lodash@4.17.21/lodash.min.js"></script>
    <script src="https://unpkg.com/dayjs@1.10.8/dayjs.min.js"></script>
    <script src="https://unpkg.com/antd@5.18.0/dist/antd.min.js"></script>
    <script src="https://unpkg.com/pako@2.1.0/dist/pako_inflate.min.js"></script>
  <script type="module" crossorigin src="https://unpkg.com/eslint-formatter-html@2.7.2/dist/index.js"></script>
  <link rel="stylesheet" crossorigin href="https://unpkg.com/eslint-formatter-html@2.7.2/dist/index.css">
</head>

<body>
    <div id="app"></div>

        <script>
            window.EslintCwd = 'C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB';
            window.EslintCreateTime = 1732733397561;
            window.EslintResults = '237,89,219,110,219,56,16,253,21,65,47,217,5,236,108,36,89,23,251,105,131,32,139,102,55,192,26,73,122,1,234,2,165
            window.EslintRulesMeta = '189,85,239,111,211,48,16,253,87,172,124,1,166,166,249,222,79,108,12,164,73,176,78,99,192,7,132,84,55,190,166,1
        </script>

</body>

</html>
```

Gráfico:



4. **Análisis de Complejidad del Código**

Configuración en eslint.config.js:

```
export default [
  {
    files: ["**/*.js", "**/*.jsx", "**/*.ts", "**/*.tsx"],
    rules: {
      "max-lines": ["warn", 300],  // El archivo no debe exceder 300 líneas
      "max-lines-per-function": ["warn", 50]  // Ninguna función debe exceder 50 líneas
    }
  }
];
```

Y ejecutamos el siguiente comando: npx eslint .

En la ejecución de EsLint, se identificaron varias advertencias relacionadas con la longitud de las funciones y el tamaño de los archivos en el proyecto. Estas advertencias indican que algunas funciones y archivos exceden el número máximo de líneas permitidas, lo que puede dificultar la mantenibilidad y legibilidad del código.

Advertencias encontradas:

1. Función con demasiadas líneas:
    - En el archivo Conection.js:
        - Línea 203: Función con 85 líneas, superando el límite de 50 líneas permitido (Advertencia: max-lines-per-function).
        - Línea 224: Función con 63 líneas, superando el límite de 50 líneas permitido (Advertencia: max-lines-per-function).
        - Línea 339: Función con 56 líneas, superando el límite de 50 líneas permitido (Advertencia: max-lines-per-function).
        - Línea 574: Función con 59 líneas, superando el límite de 50 líneas permitido (Advertencia: max-lines-per-function).
2. Archivo con demasiadas líneas:
    - En el archivo Conection.js:
        - El archivo contiene 804 líneas, superando el límite de 300 líneas permitido (Advertencia: max-lines).

Impacto y recomendaciones:

- Las advertencias sobre funciones demasiado largas sugieren que algunas funciones pueden estar realizando demasiadas tareas, lo que hace que el código sea más difícil de entender y de mantener.
- Archivos con demasiadas líneas indican que el archivo puede ser demasiado extenso, lo que hace que sea difícil localizar ciertas secciones de código, realizar modificaciones o agregar nuevas funcionalidades sin riesgos de introducir errores.

Recomendación:

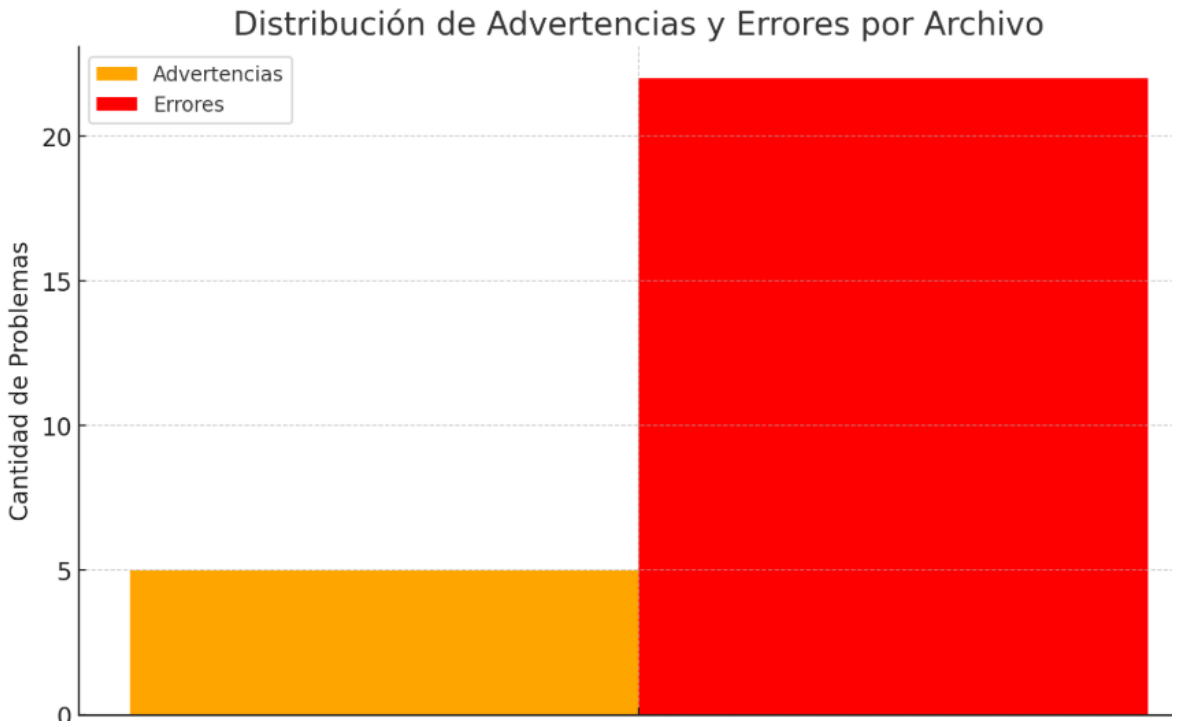
- Refactorizar las funciones largas en funciones más pequeñas y específicas.
- Dividir archivos que contienen demasiadas líneas en módulos más pequeños para mejorar la estructura del código.

Captura:

```
PR-TUBERCULOSISWEB > JS eslint.config.mjs > [o] default
1  export default [
2    {
3      files: ["**/*.js", "**/*.jsx", "**/*.ts", "**/*.tsx"],
4      rules: {
5        "max-lines": ["warn", 300],  // El archivo no debe exceder 300 líneas
6        "max-lines-per-function": ["warn", 50]  // Ninguna función debe exceder 50 líneas
7      }
8    }
9  ];
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        powershell - PR-TUBERCULOSISWEB

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\RegistrarPersonalSalud.js
  77:5  error  Parsing error: Unexpected token <

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\Transferencia.js
  99:9  error  Parsing error: Unexpected token <

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\client\src\pages\tratamiento.js
  118:5  error  Parsing error: Unexpected token <

C:\Users\63200\Desktop\PR-TUBERCULOSISWEB\PR-TUBERCULOSISWEB\server\Conection.js
  203:32  warning  Arrow function has too many lines (85). Maximum allowed is 50  max-lines-per-function
  224:45  warning  Arrow function has too many lines (63). Maximum allowed is 50  max-lines-per-function
  301:1   warning  File has too many lines (804). Maximum allowed is 300          max-lines
  339:35  warning  Arrow function has too many lines (56). Maximum allowed is 50  max-lines-per-function
  574:31  warning  Arrow function has too many lines (59). Maximum allowed is 50  max-lines-per-function

✖ 27 problems (22 errors, 5 warnings)
```

Grafico:



5. **Verificación de Consistencia de Estilo (Reglas de Estilo)**

Configuración en eslint.config.js:

```
export default [
  {
    files: ["**/*.js", "**/*.jsx", "**/*.ts", "**/*.tsx"],
    rules: {
      "quotes": ["warn", "double"],  // Preferir comillas dobles
      "semi": ["warn", "always"],    // Exigir punto y coma
      "indent": ["warn", 2]          // Usar 2 espacios para la indentación
    }
  }
];
```

Y ejecutamos el siguiente comando npx eslint .

Al ejecutar el análisis de estilo con ESLint en el proyecto, se encontraron diversos problemas relacionados con la consistencia del código y el cumplimiento de las reglas establecidas:

1. Errores de Parsing (token inesperado <): Estos errores fueron detectados en múltiples archivos dentro del código del proyecto, particularmente en los archivos de componentes y páginas del frontend. La mayoría de los errores corresponden a una mala interpretación del JSX (etiquetas < en archivos JavaScript), que puede ser causado por una configuración incorrecta del parser de ESLint para manejar JSX correctamente.

Recomendación: Verificar que el archivo de configuración de ESLint esté configurado para admitir JSX. Asegúrese de tener configurado el parser adecuado como babel-eslint o @babel/eslint-parser para que ESLint pueda interpretar correctamente el código JSX.

Ejemplos de los archivos afectados:

- client/src/App.js
- client/src/components/Footer.js
- client/src/pages/ActualizarPaciente.js
- client/src/pages/Login.js

2. Advertencias sobre el uso de comillas: Se encontraron advertencias en varios archivos debido a que no se utilizaban comillas dobles donde se esperaban, en lugar de comillas simples. Estas son reglas de estilo que se deben corregir para cumplir con las normas de estilo establecidas en el proyecto.

Recomendación: Realizar un refactor en los archivos mencionados para usar comillas dobles en lugar de comillas simples, según lo definido en las reglas de estilo. Esto ayudará a mantener la coherencia en el formato de las cadenas de texto en todo el proyecto.

Ejemplos de los archivos afectados:

- server/Conection.js
- client/src/reportWebVitals.js

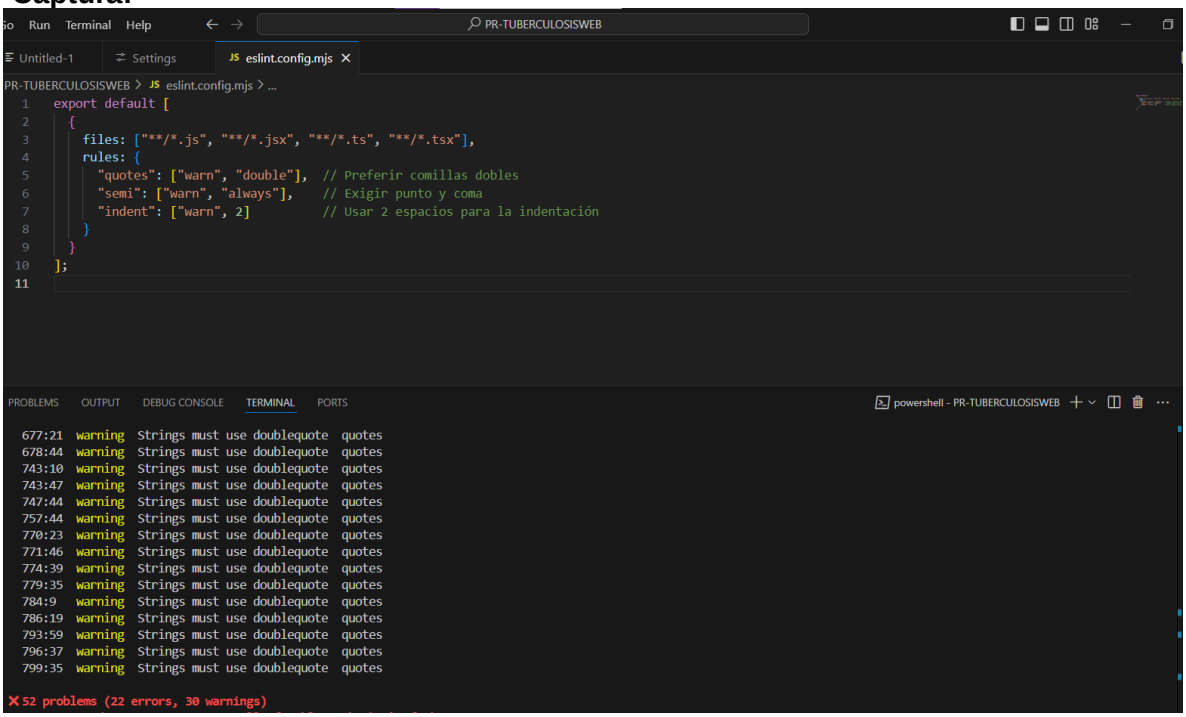
3. Archivos y funciones con demasiadas líneas: Algunos archivos y funciones exceden las líneas máximas permitidas, lo que puede afectar la mantenibilidad y legibilidad del código. Es recomendable refactorizar las funciones largas y dividir los archivos muy extensos.

Recomendación: Refactorizar las funciones largas en unidades más pequeñas y dividir los archivos demasiado grandes en módulos más pequeños y manejables. Esto mejorará la claridad del código y facilitará su mantenimiento a largo plazo.

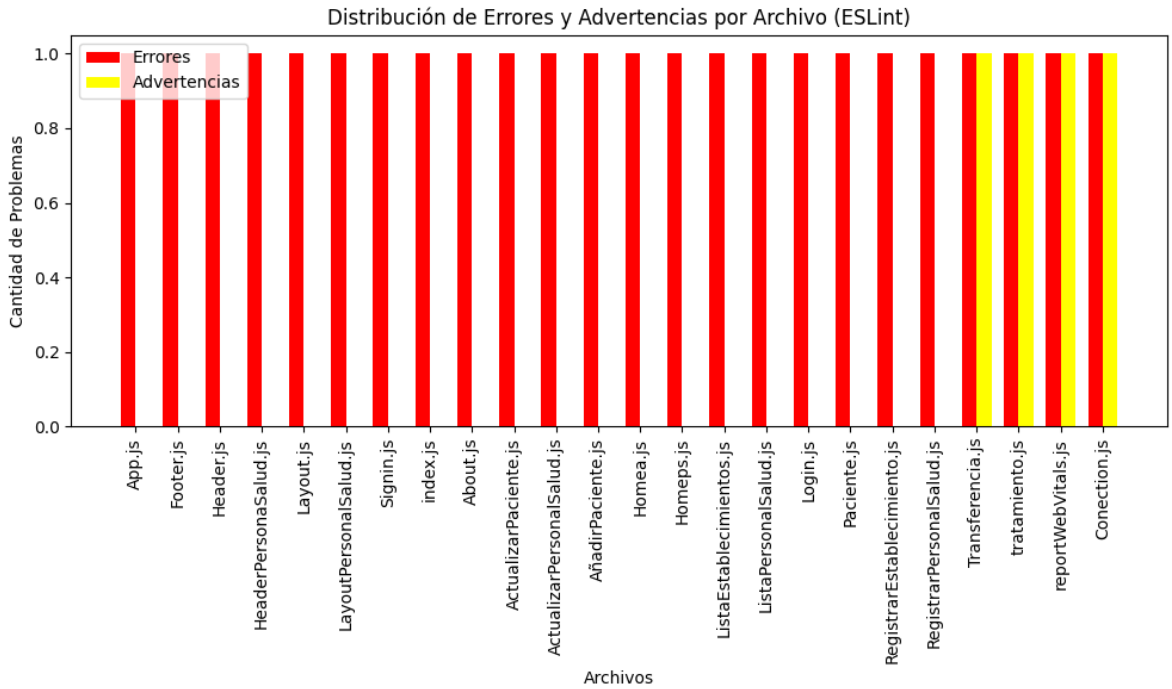
Ejemplos:

- server/Conection.js (funciones con más de 50 líneas)
- Archivos con más de 300 líneas.

Captura:



- **Grafico:**



Conclusiones

Módulo de Pacientes

Se evalúa como Bueno (4.5/5) cumple todos los requerimientos funcionales propuestos, a excepción de un bug a la hora de eliminar un paciente.

Módulo de Trabajadores de Salud

Se evalúa como regular (3/5). Aunque cumple con funcionalidades básicas, presenta problemas críticos como validaciones insuficientes en los campos, falta de botones esenciales, y errores graves como la caída del servidor y el botón de actualizar no funciona.

Módulo de Establecimiento

Se evalúa como regular (3/5) no cumple con algunos requerimientos funcionales como validar presencia de botón eliminar y editar 0.BUG