

## Reporte de Auditoría

A continuación, se precisan las diferentes fallas en la aplicación. Del lado izquierdo tenemos el error y al lado derecho la posible corrección de mantenimiento, la solución es opcional, pero se debe aumentar la calidad del código para la siguiente auditoría.

### Reporte de auditoría proyecto sin refactorizar



Los 36 problemas visibles se presentan varias veces a través del código y se dividen entre los siguientes:

1. **Cambio en la definición de variables:** Gran parte de los cambios que se deben hacer se enfocan en el manejo de variables. Es recomendable el uso de **let** para definir variables. Es importante revisar todo el módulo para hacer todas las correcciones correspondientes.

Seleccione los problemas para realizar acciones en bloque.

☐ Cambiar    Números selectos    Navegue hasta el problema    36 números    Esfuerzo de 2 horas y 45 minutos

app-legacy.js

☐ Variable inesperada, utilice let o const en su lugar. Intencionalidad

Mantenibilidad Alto    malas prácticas es2015 +

☐ Abierto    No asignado    L2    Esfuerzo de 5 minutos    Hace 19 minutos    Mal olor del código    Crítico

```
/ app-legacy.js    Consulte todos los problemas en este archivo.

1  125094...    // MÓDULO DE CALCULADORA v2 - LEGADO
2             var buffer="0";
               Variable inesperada, utilice let o const en su lugar.
3
4             var memoria=0;
5             var último_operador;
6             var histórico= [];
7             constante MÁXIMO_ELEMENTOS_HISTORIAL=5; // El dev anterior al menos puso una const
```

2. Prefiera `Number.parseInt` a `parseInt`: La idea es cambiar el uso de métodos. Se debe hacer el cambio de `parseInt` por `Number.parseInt()` haciendo el llamado a la clase `Number`.

☐ Prefiera `Number.parseInt` a `parseInt`. Consistencia

Fiabilidad Medio

Mantenibilidad Bajo

convención es2015 +

☐ Abierto

No asignado

L20

Esfuerzo de 2 minutos

Hace 20 minutos

Mal olor del código

Menor

```
16     buffer="0";memoria=0;último_operador= nulo;
17     romper;
18     caso '=':
19         si(último_operador== nulo){ devolver;}
20         flushOperationAndLog(parseInt(buffer));
```

Prefiera `Number.parseInt` a `parseInt`.

3. Prefiera `Number.isNaN` a `isNaN`: Se recomienda priorizar los métodos y propiedades estáticos numéricos sobre sus equivalentes globales.

☐ Prefiera `Number.isNaN` a `isNaN`. Consistencia

Fiabilidad Medio

Mantenibilidad Bajo

convención es2015 +

☐ Abierto

No asignado

L94

Esfuerzo de 2 minutos

Hace 27 minutos

Mal olor del código

Menor

```
92     }
93     función clic en el botón(valor){
94         si(isNaN(parseInt(valor))){ símbolo de manejo(valor);} demás { número de identificador(valor);}
95     }
96
```

Prefiera `Number.isNaN` a `isNaN`.

4. Se esperaba un bucle `for-of` en lugar de un bucle `for` con esta iteración simple. El uso de `for` a secas hace menos visible el código se recomienda usar un `for-of`.

☐ Se esperaba un bucle `for-of` en lugar de un bucle `for` con esta iteración simple. Consistencia

Mantenibilidad Bajo

torpe +

☐ Abierto

No asignado

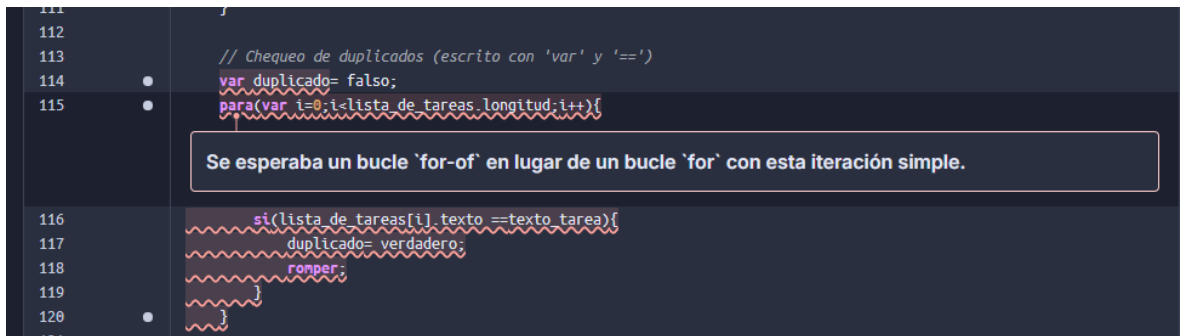
L115

Esfuerzo de 5 minutos

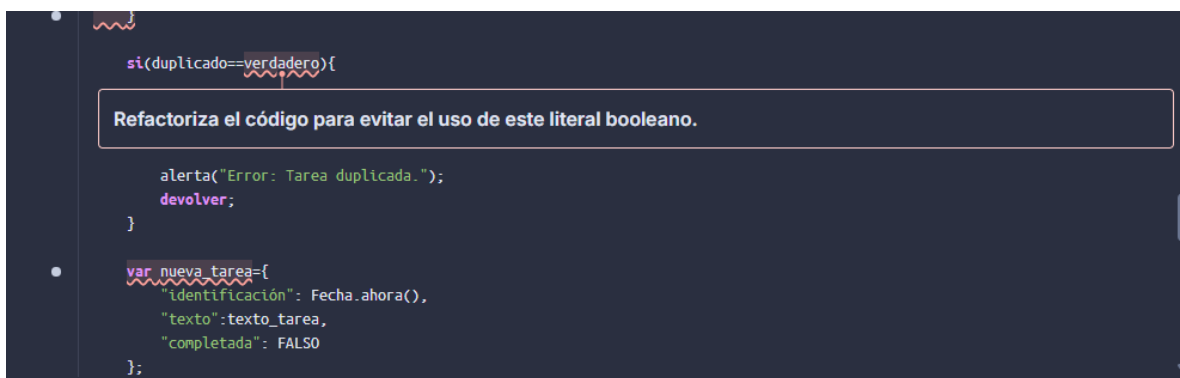
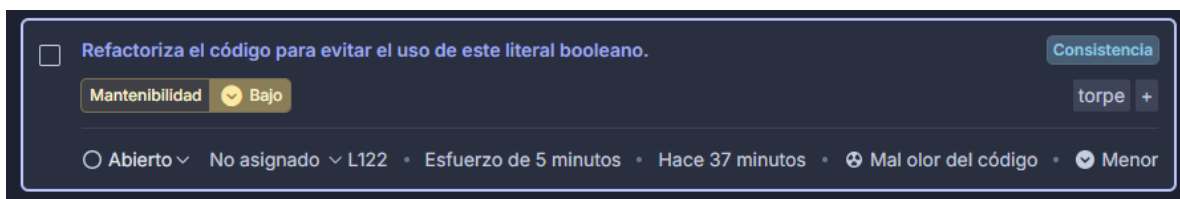
Hace 35 minutos

Mal olor del código

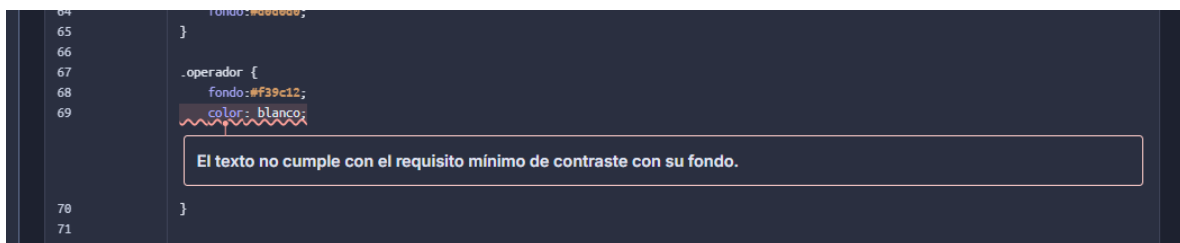
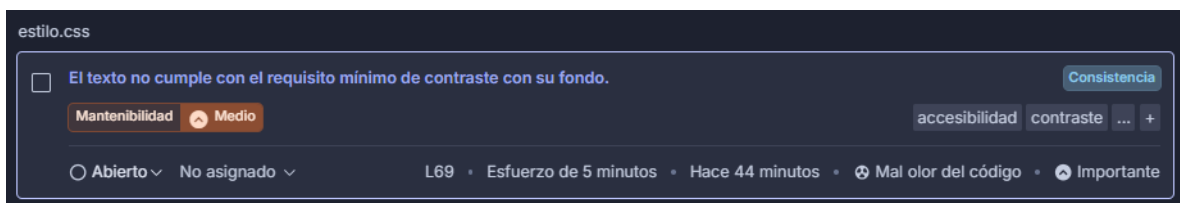
Menor



## 5. Refactoriza el código para evitar el uso de este literal booleano.



## 6. El texto no cumple con el requisito mínimo de contraste con su fondo.



<b>Métrica</b>	<b>Antes del refactor</b>	<b>Después del refactor</b>	<b>Cambio</b>
<b>Seguridad (issues)</b>	0	0	Sin cambios
<b>Fiabilidad (issues)</b>	5	5	Sin cambios
<b>Mantenibilidad (issues)</b>	36	12	Mejora significativa (-24 issues)
<b>Duplicaciones</b>	0% (en 354 líneas)	0% (en 380 líneas)	Sin cambios (más líneas, sigue en 0%)
<b>Cobertura</b>	Sin configurar	Sin configurar	Sin cambios
<b>Temas aceptados</b>	0	0	Sin cambios
<b>Puntos críticos seguridad</b>	0	0	Sin cambios