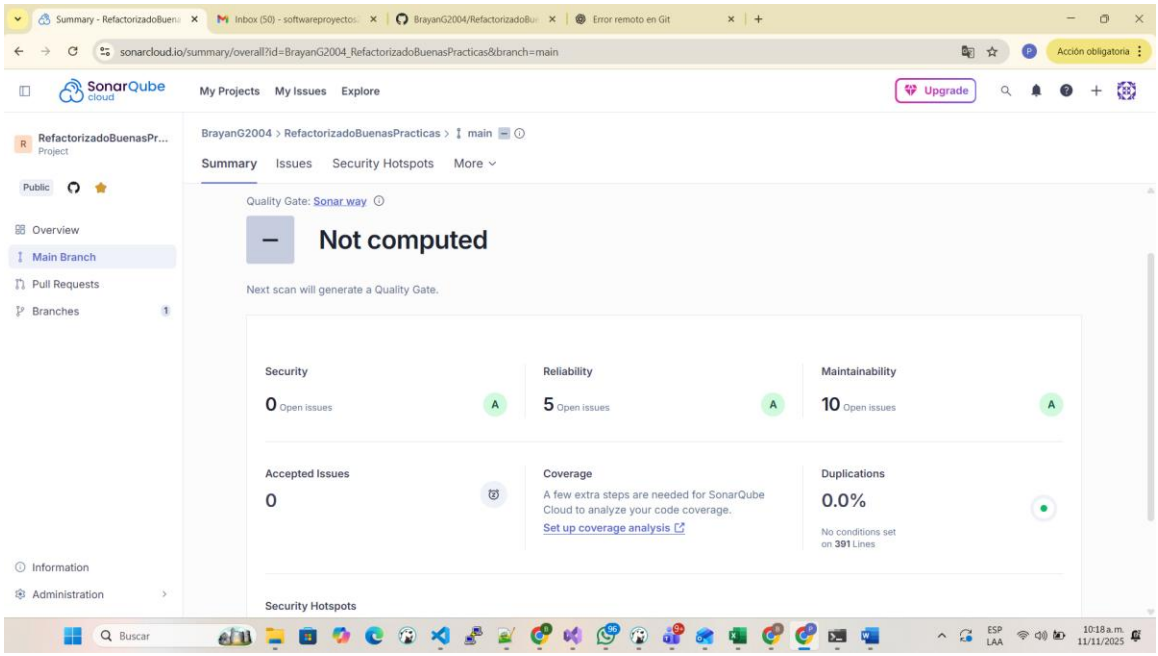
Taller de Auditoría Profesional – Versión Refactorizada (Después)

1. Evidencia de Auditoría

Captura del Dashboard principal (Summary) de la versión Refactorizada.



2. Análisis Comparativo de Calidad:

Tabla Comparativa de Métricas: Cree una tabla que resuma las clasificaciones y número de Issues para Fiabilidad, Mantenibilidad, Seguridad y Duplicación en ambas versiones.

Avances del Proyecto					
Versión	**Security**	**Reliability**	**Maintainability**	**Accepted Issues**	**Duplications**
1	0	5	36	0	0%
2	0	5	10	0	0%

A continuación se presentan los cinco hallazgos principales detectados durante la auditoría del código en la versión refactorizada (después). Cada hallazgo incluye su descripción, impacto y un espacio designado para las capturas de pantalla de SonarCloud.

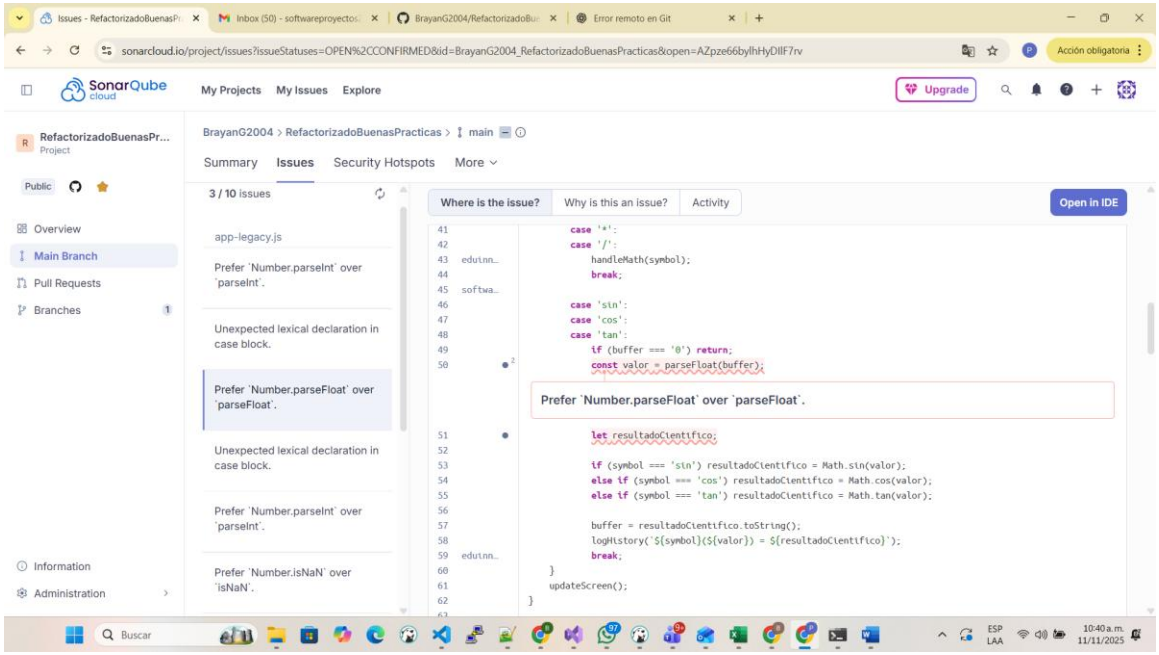
Hallazgo 1:

Descripción:

SonarCloud detectó el uso de la función global parseFloat() en lugar de la función estándar Number.parseFloat().

Impacto:
El uso de la función global puede generar comportamientos inesperados si el entorno redefine parseFloat, afectando la **fiabilidad** y **consistencia** del código.

Solución aplicada:
Se reemplazó parseFloat() por Number.parseFloat(), mejorando la compatibilidad y las buenas prácticas de programación en JavaScript.

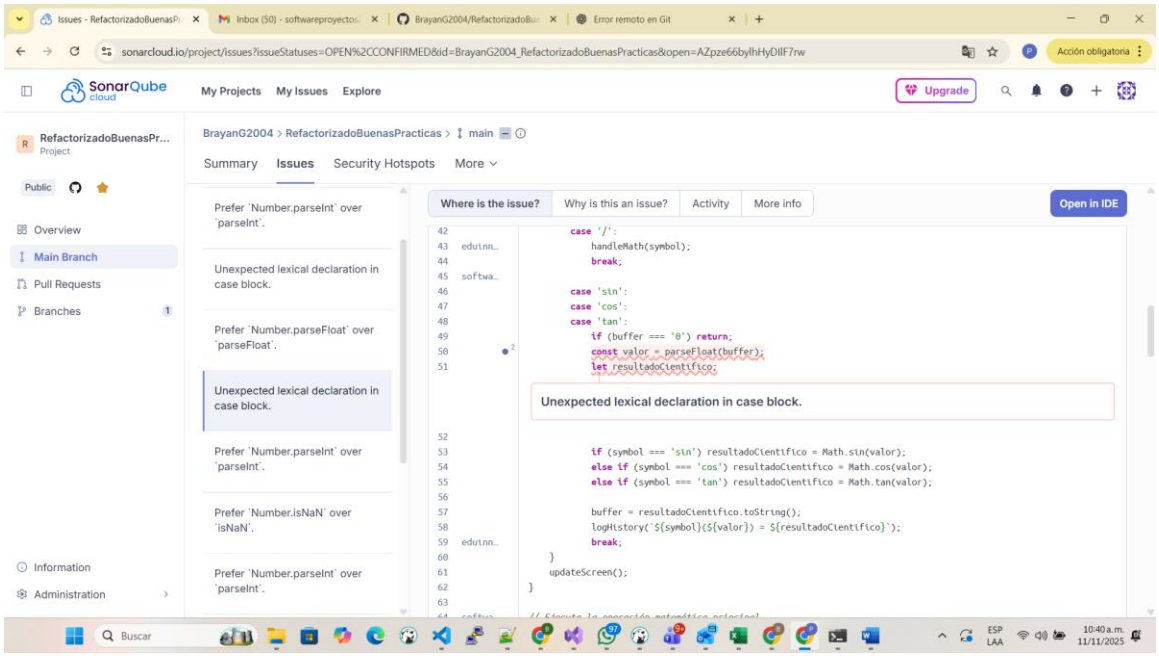


Hallazgo 2:

Descripción:
SonarCloud reportó una **declaración léxica inesperada** dentro de un bloque case en una estructura switch. Se detectó el uso de let dentro del case, lo que puede producir errores de alcance (scope) en JavaScript.

Impacto:
Declarar variables con let o const directamente dentro de un case puede causar errores en la ejecución del bloque, afectando la **fiabilidad** del código.

Solución aplicada:
Se encapsuló la declaración dentro de un bloque { } o se movió fuera del case correspondiente, garantizando un alcance adecuado y evitando errores de referencia.



Hallazgo 3:

Descripción:

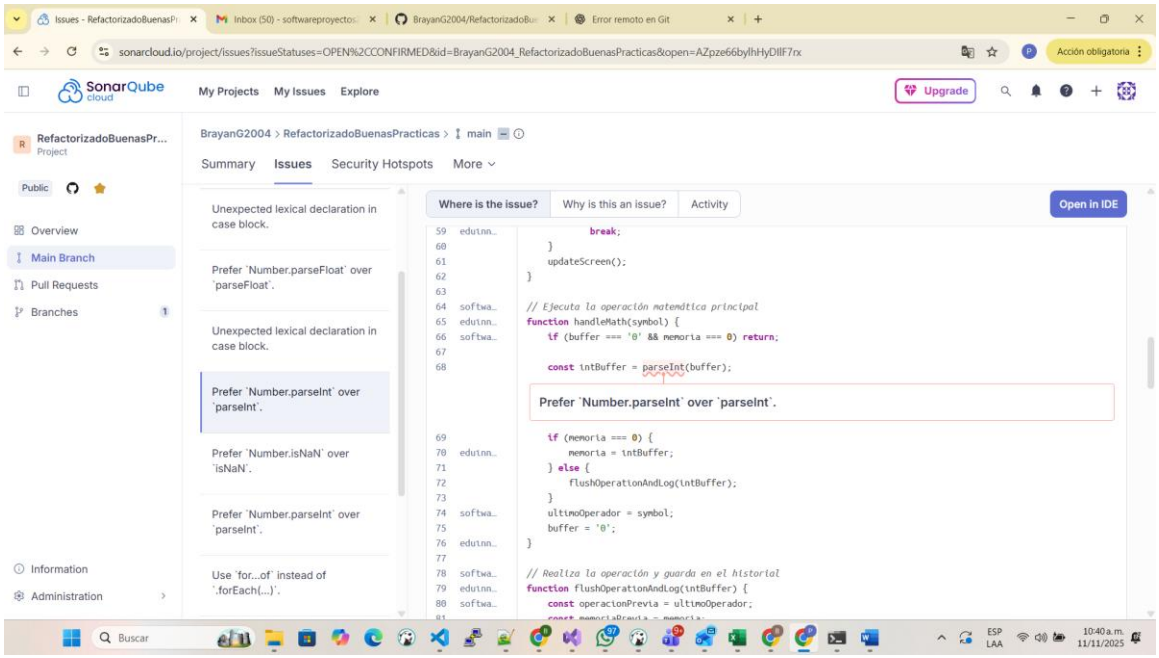
El análisis de SonarCloud identificó el uso de la función global parseInt() en lugar de la versión estándar Number.parseInt().

Impacto:

La función global parseInt puede ser sobrescrita o comportarse de manera inconsistente en algunos contextos, afectando la **seguridad** y **fiabilidad** del código.

Solución aplicada:

Se reemplazó parseInt() por Number.parseInt(), asegurando un código más robusto y conforme a las recomendaciones del estándar ECMAScript.

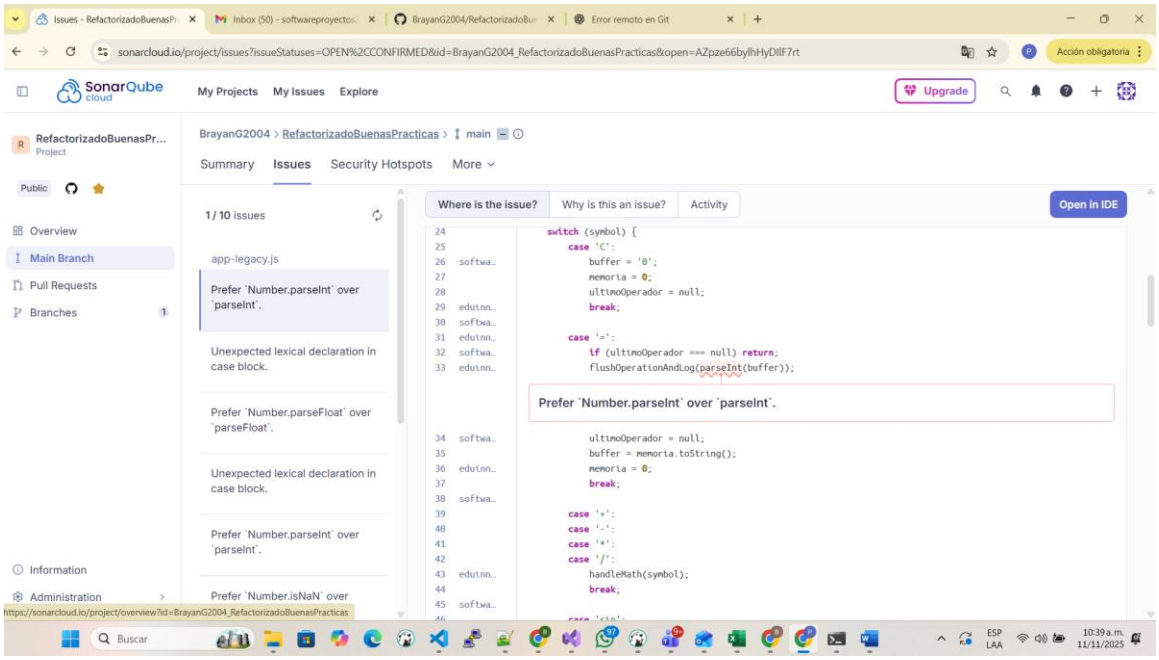


Hallazgo 4:

Descripción:
SonarCloud detectó el uso de la función global parseInt() dentro de una estructura switch, específicamente en el bloque que maneja operaciones matemáticas.

Impacto:
El uso de la función global parseInt() puede causar inconsistencias si el entorno redefine esta función o si el código depende del contexto global, afectando la **fiabilidad** y **estabilidad** de la aplicación.

Solución aplicada:
Se reemplazó el uso de parseInt() por Number.parseInt(), asegurando una implementación más segura y conforme con las recomendaciones modernas de JavaScript (ES6+). Esto mejora la mantenibilidad y reduce el riesgo de errores en entornos complejos.

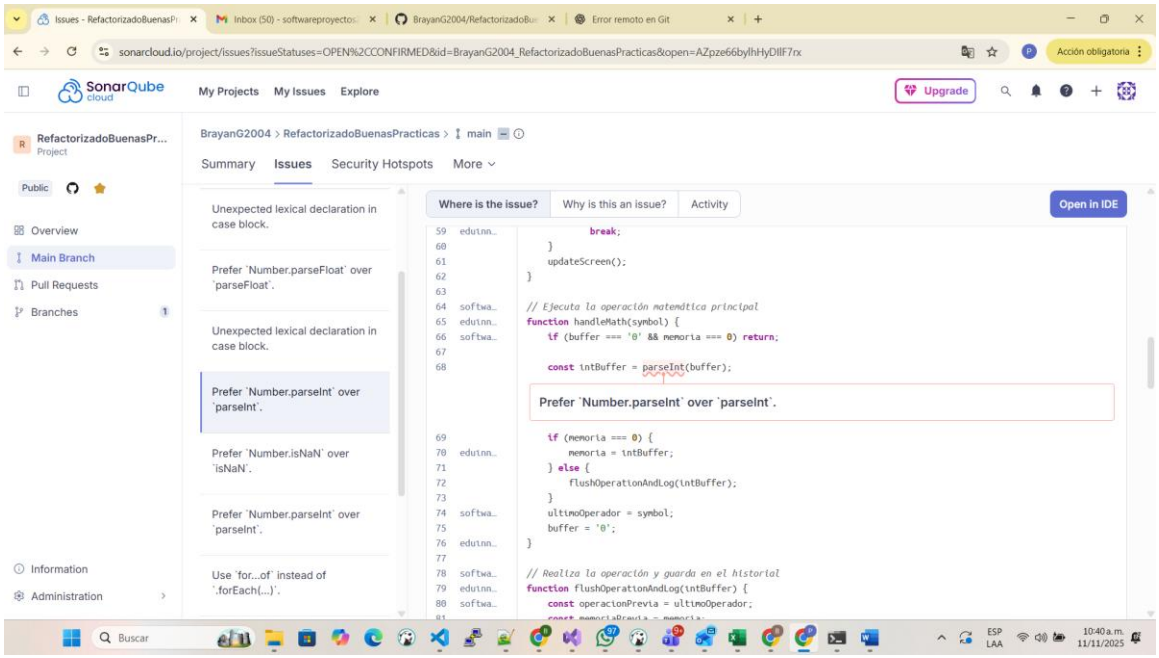


Hallazgo 5:

Descripción:
SonarQube detecta que **se está declarando una variable (let, const o class) directamente dentro de un bloque case de un switch**, sin encerrar ese bloque entre llaves { }..

Impacto:
Esto puede causar errores de **alcance (scope)** o **comportamiento inesperado**, ya que las declaraciones let y const están limitadas al bloque {} donde se declaran.
Sin las llaves, el intérprete de JavaScript puede confundir el ámbito de esas variables, generando errores en tiempo de ejecución..

Solución aplicada:
Encierra el contenido del case en un bloque { } si vas a declarar variables con let o const.


Issues - RefactorizadoBuenasPr
Inbox (50) - softwareproyectos
BrayanG2004/RefactorizadoBue
Error remoto en Git
sonarcloud.io/project/issues?issueStatuses=OPEN%2CCONFIRMED&id=BrayanG2004_RefactorizadoBuenasPracticas&open=AZpze66bylhHyDIlF7rx
Acción obligatoria
SonarQube cloud
My Projects
My Issues
Explore
Upgrade
RefactorizadoBuenasPr...
Project
Public
Overview
Main Branch
Pull Requests
Branches
Information
Administration
BrayanG2004 > RefactorizadoBuenasPracticas > main
Summary
Issues
Security Hotspots
More
Unexpected lexical declaration in case block.
Prefer `Number.parseFloat` over `parseFloat`.
Unexpected lexical declaration in case block.
Prefer `Number.parseInt` over `parseInt`.
Prefer `Number.isNaN` over `isNaN`.
Prefer `Number.parseInt` over `parseInt`.
Use `for...of` instead of `.forEach(...)`.
Where is the issue?
Why is this an issue?
Activity
Open in IDE
Prefer `Number.parseInt` over `parseInt`.
// Ejecuta la operación matemática principal
function handleMath(symbol) {
if (buffer === '0' && memoria === 0) return;
const intBuffer = parseInt(buffer);
// Realiza la operación y guarda en el historial
function flushOperationAndLog(intBuffer) {
const operacionPrevia = ultimoOperador;
Buscar
ESP
LAA
10:40 a.m.
11/11/2025