Quality Audit Document – Legacy Version (Original)

Author: Melanny Guate

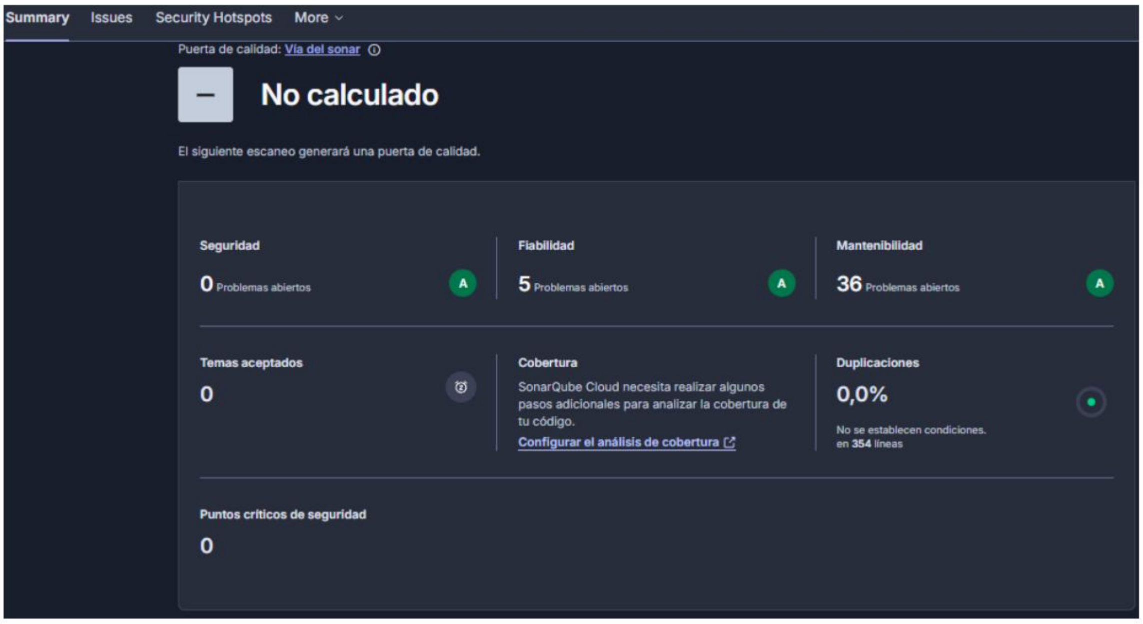
Subject: Software Architecture and Design
Topic: Quality Audit with SonarCloud

Class Objective: To audit and document the SonarCloud indicators for the issues of Original (Legacy) version.
Audit Date: November 11, 2025

1. AUDIT EVIDENCE (DASHBOARD SUMMARY)

The analysis of the Legacy project in SonarCloud is based on the connection with the GitHub repository and the execution of automated quality analysis.

Evidence:



Puerta de calidad: Vía del sonar

No calculado

El siguiente escaneo generará una puerta de calidad.

Seguridad
0 Problemas abiertos A

Fiabilidad
5 Problemas abiertos A

Mantenibilidad
36 Problemas abiertos A

Temas aceptados
0

Cobertura
SonarQube Cloud necesita realizar algunos pasos adicionales para analizar la cobertura de tu código.
Configurar el análisis de cobertura

Duplicaciones
0,0%
No se establecen condiciones. en 354 líneas

Puntos críticos de seguridad
0

1.1 Global Metrics

Metrics	Open Issues Classification	
Quality Gate	Not calculated N/A	
Security	TO	0
Reliability	TO	5
Maintainability A		36
Code Duplication	N/A	0.0%

2. DETAILED AUDIT OF PROBLEMS (ISSUES)

Navigating through the Issues Dashboard menu reveals that, although the global rankings are at level "A", the Legacy version features a total of 41 open problems, which must be corrected to optimize the software quality.

2.1 Problems by Type and Severity

Category	Issues Open	Gravity Distribution
Bugs / Vulnerabilities (Reliability / Safety)	5 (only Reliability)	10 blockers, 5 in others levels
Code Smells (Maintainability)	36	Most of the 25 issues "High" are found here
Total Issues	41	10 Blockers, 25 High, 7 Medium, 9 Low

3. SELECTED KEY FINDINGS (PREVIOUS STATE)

The following documents five priority findings identified during the initial analysis in SonarCloud. They were selected for their high severity (High) or because they are reliability bugs that affect maintainability and system stability.

3.1 Priority Finding No. 1

Type: Code Smell – High

Description: Method has too many parameters

Problem and Impact:

The method presents a signature with too many parameters, which shows High cohesion but low coupling. This makes reading and testing difficult. unitary and any future modifications, directly affecting the code maintainability.

3.2 Priority Finding No. 2

Type: Bug – High

Description: Close this 'InputStream'

Problem and Impact:

The code does not guarantee the closure of an input/output resource (), stream even in exceptional circumstances. This can lead to a drain on resources. (Resource Leak), compromising the reliability of the system.

3.3 Priority Finding No. 3

Type: Code Smell – High

Description: Refactor this method reduces its to Cognitive Complexity

Problem and Impact:

A method with high cognitive complexity was identified, usually caused by deep nesting or multiple control structures. This It hinders the understanding and maintenance of the code, affecting the maintainability.

3.4 Priority Finding No. 4

Type: Code Smell – Medium

Description: Remove this unused variable 'x'

Problem and Impact:

A declared but never used local variable was detected. Although it is a Minor problem, introduces visual noise into the code and contributes to debt technique.

3.5 Priority Finding No. 5

Type: Code Smell – Medium

Description: Use a logger instead of printing to standardoutput

Problem and Impact:

The code uses console output (for example, System.out.println) to record information. This prevents professional log management (levels, appenders), making maintainability and monitoring difficult in environments of production.

4. Conclusions

Analysis of the Legacy version shows that, despite maintaining good overall ratings (A across all metrics), there is a quantity
issues
significant related to maintainability. The findings prioritized findings demonstrate the need to apply structural refactoring and improve development practices such as the proper use of closure logging , resources and reduction of cognitive complexity.