Lint Report

Acme AirNav Solutions, Inc

Group Number: C1.066
Repository: `https://github.com/mquirosq/DP2-C1.066`

Members:
María Quirós Quiroga, marquiqui@alum.us.es
Guillermo Rodríguez Narbona, guirodnar@alum.us.es
Ignacio Mora Pérez, ignmorper1@alum.us.es
Daniel Herrera Urbano, danherurb@alum.us.es
Alejandro Parody Quirós, aleparqui@alum.us.es

May 26, 2025

Contents

Executive Summary

This document presents the analysis of the code smells identified by SonarLint in the implementation of the group requirements [1]. The purpose of this analysis is to assess each reported issue and determine whether corrective action is necessary.

The findings demonstrate that while the reported smells may technically violate certain coding conventions or best practices, in this our context, an educational, locally executed project, they do not negatively affect functionality, readability, security, or maintainability. In many cases, the design choices reflect deliberate trade-offs made to preserve domain clarity, reduce unnecessary complexity, or support developer intent.

This document provides transparency into the development process and serves as a formal reference for quality assurance going forward.

Revision History

Revision	Date	Description
1.0	2025-05-26	Initial document

1. Introduction

The purpose of this document is to present and analyze the code smells reported by SonarLint in the section of the project corresponding to the group requirements [1]. Each reported issue is examined and justified to demonstrate why it is considered innocuous and has not been addressed through code changes.

This report is structured to ensure transparency in the decision-making process and to serve as documentation for future reference, supporting maintainability and quality assurance.

2. Bad smells

The bad smells detected by Sonar Lint are the following:

- Rename the `SystemConfiguration` package to follow Java's convention of using single, lowercase words for package names.
  - **Justification:** While the Java convention is to use a single, lowercase word for package names, the term `SystemConfiguration` is used to express a meaningful concept, the configuration of the system. Splitting or simplifying it would reduce semantic clarity. So, the name should be preserved for better maintainability and understanding.
- Rename the `IATACode` attributes and variables to follow Java's camelCase convention.
  - **Justification:** Although the Java convention uses camelCase for variable names, the term `IATACode` represents a well-known domain term, the IATA code. Changing its capitalization would reduce readability and clarity. Keeping the original name is aligned with the domain and enhances comprehension.
- Catch `Exception` instead of `Throwable` in the money exchange API call.
  - **Justification:** In this case, catching `Throwable` is intentional, as the external API may have unpredictable behavior. This approach ensures that the application handles any unexpected failures from the API gracefully and maintains stability.
- Catch `Exception` instead of `Throwable` in the `ServiceAdvisor`.
  - **Justification:** This use of `Throwable` is intentional. The goal is to prevent the failure of the entire request if the query to obtain a random service fails. This query is used in the footer display and is not absolutely necessary for the correct functioning of the system. This enhances stability and user experience, event if this non-crititcal component encounters an unexpected issue.

- Remove the word "picture" from the alternative description text (alt attribute) of the service image in `footer.jsp`.
  - **Justification:** The word "picture" in this context is part of a variable name rather than the description of the picture. The actual alternative visualization is referenced using this variable, so it does not make sense to remove it as it is not redundant as SonarLint flagged it.
- Avoid storing database passwords in `platform-development.properties`, `platform-testing.pr` and `platform-production.properties`.
  - **Justification:** This is a local educational project, and it is not intended for deployment in real environments. Credentials are only used for localhost access and no security-sensitive data is at risk. So the passwords are not a security breach and may remain in the files.
- Define constants instead of duplicating string literals such as `"airline"` and `"confirmation"` .
  - **Justification:** These literals are short, descriptive and domain-specific, making their usage clear and self-explanatory in the context. Introducing constants would add unnecessary abstraction and likely require a large centralized constants class, reducing code clarity rather than improving it.
- Override the `equals` method in the `Aircraft`, `Airline`, `Airport`, `BannedPassenger`, `Review`, `SystemConfiguration`, and `Service` classes.
  - **Justification:** Overriding `equals` is important when object are compared by content. However, in our case, these entities are uniquely identified by their unique identifier (`id`). Therefore, the default behavior of identity comparisson is enough and overriding `equals` is unnecessary.
- Replace `assert` statements in validators with explicit condition checks.
  - **Justification:** In validators, assertions are used to verify conditions that should never be violated under correct usage. When violated, it is typically due to developer errors. In addition, they are disabled by default at runtime, ensuring no performance impact in production. Replacing them with runtime checks could lead to unnecessary overhead, as they are only there for development purposes.

3. Conclusions

The analysis recorded in this document provides a comprehensive evaluation of the code smells identified by SonarLint in the implementation of the group requirements [1]. For each reported issue, a detailed justification explaining why it is considered harmless and has intentionally been kept is provided.

The analysis provides a clear rationale for maintaining these elements, ensuring ensuring transparency in the decision-making process. Moving forward, this document will serve as the reference for quality assurance, helping to prevent unnecessary modifications and reducing the risk of confusion during future reviews.

References

[1] *00 - Requirements – Group*, Enseñanza Virtual, 2025. Available on Enseñanza Virtual in the course inside the Project Statement folder.