Lint Report

Acme AirNav Solutions, Inc

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

Student #2:
María Quirós Quiroga, marquiqui@alum.us.es

May 25, 2025

Contents

Executive Summary

This document presents the analysis of the code smells identified by SonarLint in the implementation of requirements assigned to Student #2 [1]. The purpose of this analysis is to assess each reported issue and determine whether corrective action is necessary.

Upon thorough review, all remaining code smells are either stylistic, context-dependent or intentionally designed to meet specific architectural or functional needs of the project. For each issue, a clear justification explaining why it is considered harmless and no changes are necessary is included.

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	2025-05-25	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 requirements for Student #2 [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 `bookingRecord` packages for administrator and customer features 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 `bookingRecord` is used to express a meaningful domain concept, the intermediate relationship between bookings and passengers. Splitting or simplifying it would reduce clarity. So, the name should be preserved to reference the domain model.
- Catch `Exception` instead of `Throwable` in the recommendation 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.
- Define constants instead of duplicating string literals such as `"country"`, `"flight"`, `"travelClass"` and `"passenger"`.
  - **Justification:** These literals are short, descriptive and domain-specific, making their usage clear and self-explanatory in the context. Introducing constants would add unnecessary indirection and likely require a large centralized constants class, which would increase complexity without significant benefit.
- Override the `equals` method for the `Booking`, `BookingRecord`, `Customer`, `Passenger`, and `Recommendation` classes.
  - **Justification:** Overriding `equals` is usually necessary when object equality is based on its properties. In our case, object equality is determined only by the unique identifier (`id`). Therefore, the default behaviour is enough, and overriding `equals` is unnecessary.

- Replace `assert` statements in validators with explicit condition checks.
  - **Justification:** In validators, assertions are used to very 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 requirements for Student #2 [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] *02 - Requirements – Student #2*, Enseñanza Virtual, 2025. Available on Enseñanza Virtual in the course inside the Project Statement folder.