COMPREHENSIVE TECHNICAL DOCUMENTATION: DELIVERY 2

Governance Pipeline & Tech Debt Audit

Project: Pdds-ISA-Project (Spring PetClinic)

Engineer: Pablo Pineda & Christian Martinez

Submission Date: February 16nd, 2026

1. Governance Infrastructure (CI Workflow)

An automated quality control system was implemented using GitHub Actions to ensure software integrity. This system acts as a 'Quality Gate' that evaluates every code change before final integration.

1.1 Tools and Technologies:

- **GitHub Actions:** CI/CD orchestrator for executing governance rules.
- **SonarCloud:** Static code analysis and metrics centralization.
- **Checkstyle:** Coding standards auditing (Sun Checks).
- **JaCoCo:** Unit test coverage measurement.
- **Maven:** Dependency management and build automation.

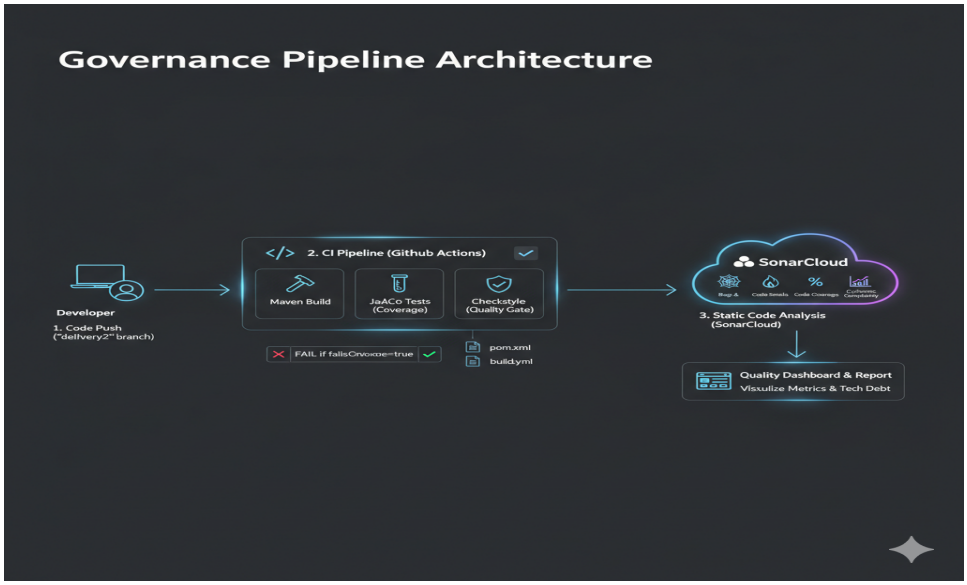

Figure 1.1: Governance Pipeline Architecture Diagram.

1.2 Specific Configurations:

- **build.yml:** Redundant workflows were removed, and the analysis command was configured to force visibility on the SonarCloud main branch (-Dsonar.branch.name=main).
- **pom.xml:** Plugin integration with 'failsOnError=false' to allow a comprehensive debt report without interrupting the dashboard visibility flow.

2. Tech Debt Audit (Hotspots Identification)

Using pipeline data, an audit was conducted to identify 'Hotspots': files where high complexity and change frequency (churn) generate operational risks.

2.1 Identified Hotspots (Cyclomatic Complexity > 20):

- **OwnerController.java:** Identified as a critical point due to excessive search and edit logic (Fat Controller).
- **PetController.java:** High structural fragility caused by tight coupling in validation management.
- **Owner.java:** Entity with structural complexity exceeding recommended standards for data models.

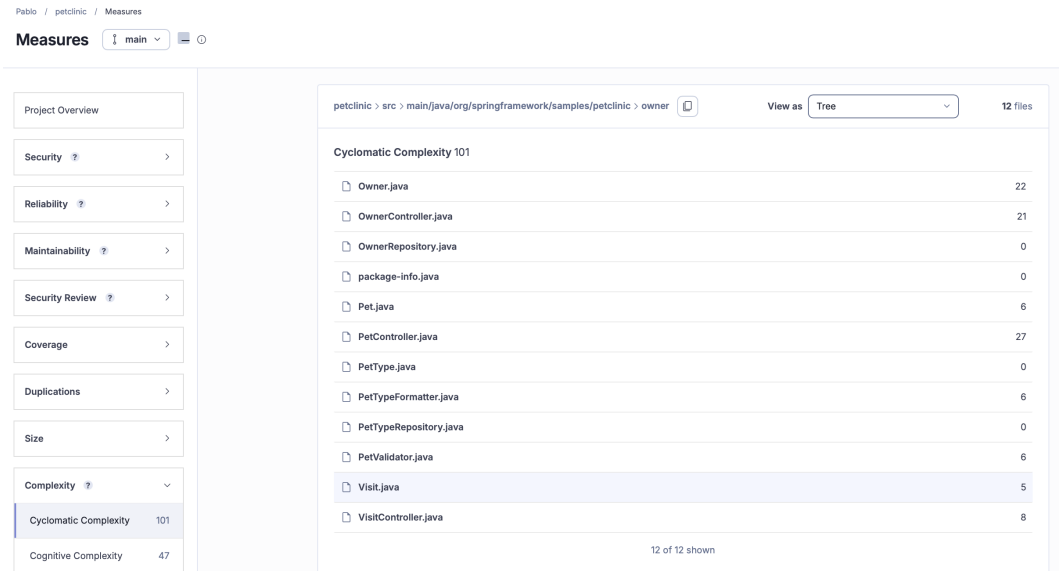

Figure 2.1: Complexity analysis and metrics in SonarCloud (mesure.png).

3. Refactoring Plan: Strangler Fig Pattern

To mitigate the identified debt, a controlled migration towards a cleaner architecture is proposed:

- **Phase 1 (Interception):** Creation of a decoupled Service Layer to extract logic from the controllers.
- **Phase 2 (Strangling):** Gradual migration of complex methods, validated at each step by the CI Pipeline and JaCoCo.
- **Phase 3 (Elimination):** Final removal of legacy code once SonarCloud metrics validate the new structure.

4. Execution Evidence

Figure 4.1: Successful Pipeline State in GitHub Actions (verde.png)

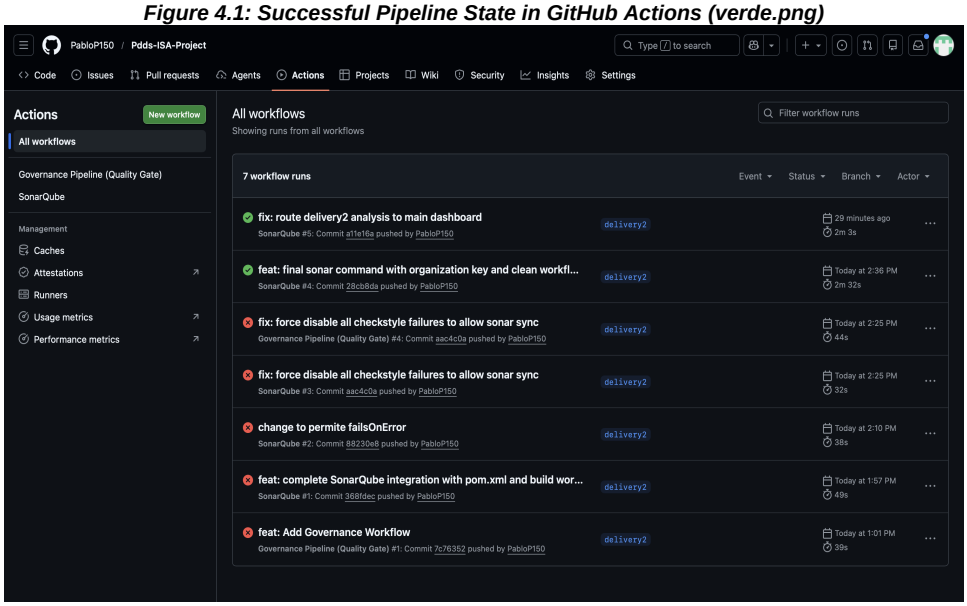

Figure 4.2: Consolidated Quality Dashboard in SonarCloud (sonar.png)