Universidad Nacional de San Agustín de Arequipa

Ingeniería de Software II

Clean Code & SonarQube

Integrantes :
- Aquise Santos, Angela Margarita
- Chullunquía Rosas, Sharon Rossely
- Jara Huilca, Arturo Jesús
- Rosas Arotaype, Oscar Eugenio
- Vilca Alvites, Cecilia del Pilar

Profesor :
- Sarmiento Calisaya, Edgar

6 de octubre de 2020

Índice

1. Conocimientos Previos

1.1. Clean Code

Libro que se divide en tres partes. El primero describe los principios, patrones y prácticas para escribir código limpio. La segunda parte consta de varios estudios de casos de complejidad creciente. Cada estudio de caso es un ejercicio de limpieza de código, de transformar una base de código que tiene algunos problemas en una que sea sólida y eficiente. La tercera parte es la recompensa: un solo capítulo que contiene una lista de heurísticas y "smellsrecopilados al crear los estudios de caso. El resultado es una base de conocimientos que describe la forma en que pensamos cuando escribimos, leemos y limpiamos código. [1]

1.2. SonarQube

SonarQube (conocido anteriormente como Sonar [2]) es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.

1.3. Issues

Al ejecutar un análisis, SonarQube plantea un problema cada vez que un fragmento de código rompe una regla de codificación. El conjunto de reglas de codificación se define a través del Perfil de Calidad asociado para cada idioma del proyecto. [3]

1.3.1. Issue Types

Hay tres tipos de problemas:

- **Bug :** Un error de codificación que romperá su código y debe corregirse de inmediato.
- **Vulnerability :** Un punto de su código que está abierto a ataques.
- **Code Smell :** Un problema de mantenimiento que hace que su código sea confuso y difícil de mantener.

1.3.2. Issue Severity

Cada problema tiene una de cinco gravedad:

- **Blocker :** Error con una alta probabilidad de afectar el comportamiento de la aplicación en producción: pérdida de memoria, conexión JDBC no cerrada.El código DEBE ser reparado inmediatamente.

- **Critical :** O un error con baja probabilidad de afectar el comportamiento de la aplicación en producción o un problema que representa una falla de seguridad. El código DEBE ser revisado inmediatamente.
- **Major :** Defecto de calidad que puede impactar enormemente en la productividad del desarrollador: código descubierto, bloques duplicados, parámetros no utilizados, ...
- **Minor :** Defecto de calidad que puede afectar ligeramente la productividad del desarrollador: las líneas no deben ser demasiado largas, las declaraciones de çambio"deben tener al menos 3 casos, ...
- **Info :** Ni un error ni un defecto de calidad, solo un hallazgo.

1.4. Smell

Una forma de ver los *smells* es con respecto a los principios y la calidad: "Los *smells* son ciertas estructuras en el código que indican una violación de los principios fundamentales del diseño y tienen un impacto negativo en la calidad del diseño". Los *code smells* no suelen ser errores ; no son técnicamente incorrectos y no impiden que el programa funcione. En cambio, indican debilidades en el diseño que pueden ralentizar el desarrollo o aumentar el riesgo de errores o fallas en el futuro. [4] El *Smelly code* hace (probablemente) lo que debería, pero será difícil de mantener. En el peor de los casos, será tan confuso que los encargados del mantenimiento pueden introducir errores sin darse cuenta. Los ejemplos incluyen código duplicado, código descubierto por pruebas unitarias y código demasiado complejo. [5]

2. Relación entre Clean Code y SonarQube

Reporte que describe la relación entre prácticas descritas en Clean Code y reglas implementadas en plugins de SonarQube para el lenguaje C++ (Link).

Referencias

[1] Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882, 2008.

[2] Freddy Mallet, *SONAR is becoming SONARQUBE*. http://sonar.15.x6.nabble.com/SONAR-is-becoming-SONARQUBE-td5010134.html.

[3] SonarQube Docs. 8.5, *Issues*. https://docs.sonarqube.org/latest/user-guide/issues/#:~:text=While%20running%20an%20analysis%2C%20SonarQube,each%20language%20in%20the%20project..

[4] Suryanarayana, Girish, *Refactoring for Software Design Smells* . https://en.wikipedia.org/wiki/Code_smell, 2014.

[5] SonarQube, *Detect Tricky Issues* .https://www.sonarqube.org/features/issues-tracking/.