Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería Informática del Software

Diseño y Pruebas II

Curso 2021/2022

LINT REPORT

Repositorio: https://github.com/Ginpasfer/Acme-Recipes

Grupo de Prácticas	*S07*
Estudiantes	**Rol**
Pastor Fernández, Ginés	Project Manager Developer Operator Tester
Giráldez Álvarez, Pablo	Developer Analyst Tester
Rijo Hernández, Badayco	Developer Tester
Solís Miranda, Antonio Manuel	Developer Tester
Paradas Borrego, Álvaro	Developer Tester

Índice

1. Resumen ejecutivo 3
2. Tabla de revisiones 3
3. Introducción 3
4. Contenido 3
5. Conclusión 5
6. Bibliografía 5

1. Resumen ejecutivo

El presente documento tiene como finalidad reflejar aquellos “malos olores (*Bad Smells*)” detectados en el proyecto **Acme-Recipes,** mediante el uso de la herramienta **SonarLint** integrada en el entorno de desarrollo de Eclipse. En este informe se indicarán los “malos olores”, junto con la solución de estos, además se indicarán los “malos olores” que no se han corregido, así como la justificación de dicha acción.

2. Tabla de revisiones

Versión	Fecha	Autor	Descripción de cambios
1.0	17/08/2022	Badayco Rijo Hernández	- Creación del documento

3. Introducción

En el presente documento se expondrá como, mediante el uso de la extensión de SonarLint, se han detectado los “Bad Smells” que existían durante el desarrollo del proyecto. La calidad y la seguridad del proyecto es un aspecto importante, pero sucede que la mayoría de las veces durante la programación del proyecto surgen ciertos errores a nivel de calidad.
Por ello, se ha decidido utilizar el Plugin **SonarLint** para el IDE de Eclipse, que hace el análisis estático y encuentra errores en el nivel de tiempo de compilación. Como desarrolladores se nos proporcionará un feedback temprano sobre el código para realizar una corrección lo antes posible, para asegurar una rápida resolución de los problemas internos de calidad; en definitiva, un modelo que se basa en medir la calidad desde el principio, en lugar de considerarse después de los hechos. SonarLint trabaja fuera de línea y detecta problemas de calidad de forma espontánea que ayudan a actuar de forma proactiva.
El mecanismo de control de calidad de SonarLint soporta toda la base de datos de reglas necesarias y requeridas. Las sugerencias y las descripciones de los errores son muy significativas. Ciertamente, ayudará a aumentar la calidad del código y a disminuir el esfuerzo de revisión de un revisor. A continuación, se presentan los puntos sobre la ayuda que proporciona SonarLint:

- **Orientación clara:** Descubre los problemas en el contexto con descripciones de reglas, una guía clara y ejemplos de código para aprender cómo solucionarlos.

- **Flujo de datos de problemas:** Resalta todas las ubicaciones afectadas en el código permitiendo tener una visión clara de los problemas que se deben abordar.

- **Análisis rápido y preciso:** El análisis de alta precisión sobre la marcha significa menos falsos positivos y resultados consistentes y confiables.

Por tanto, después de esta introducción, en este informe se presentará un espacio de *Contenido* donde se desarrollará como se ha analizado el proyecto, cuáles han sido los errores detectados, así como la solución que se ha aportado a cada “Bad Smell”. Así mismo, se mostrarán una serie de ilustraciones sobre el código analizado para remarcar lo que se quiere expresar.
Finalmente, se concluirá con el aprendizaje que ha realizado el equipo del proyecto sobre el uso de esta clase de herramientas.

4. Contenido

En el presente apartado se indicará como se ha llevado a cabo el análisis del proyecto con la herramienta de SonarLint, así como cuáles han sido los errores encontrados y la solución a los mismos.
En primer lugar, para el análisis del código, se han subido los cambios a la rama de *develop*, ya que de esta manera podremos a analizar la rama completa. Una vez hecho, se introduce el proyecto en el entorno de desarrollo de Eclipse y pulsando el botón izquierdo del ratón sobre el proyecto,

aparece una opción que se llama **SonarLint.** En dicha opción, aparece un desplegable en el que hay que darle a la opción de **Analyze.** Una vez hecho, se analizará todo el código del proyecto y después de unos minutos aparecerán los “Bad Smells” encontrados en el proyecto por parte de la herramienta como se aprecia en la **ilustración 1**.
Después de un análisis de los “Bad Smells” encontrados, así como en los archivos donde estos se hallaban, se pudo apreciar que el único “Bad Smell” correspondiente al proyecto se ubicaba en el fichero **AuthenticatedSystemConfigurationShowService.java.** Se observó que el “Bad Smell” se trataba de variable local que luego se devolvía (**ilustración 2**). A continuación, se mostrará un fragmento de la descripción (traducida) que proporcionó SonarLint respecto a esto:

Las variables locales no deben ser declaradas y luego devueltas o lanzadas inmediatamente

Declarar una variable sólo para devolverla o lanzarla inmediatamente es una mala práctica.

Algunos desarrolladores argumentan que esta práctica mejora la legibilidad del código, porque les permite nombrar explícitamente lo que se devuelve. Sin embargo, esta variable es un detalle interno de la implementación que no se expone a quienes llaman al método. El nombre del método debería ser suficiente para que los llamantes sepan exactamente lo que se devolverá.

Resource	Date	Description
AuthenticatedSystemConfigurationShowService.java		Immediately return this expression instead of assigning it to the temporary variable "sc".
acme.js	few seconds ago	Remove the declaration of the unused 'questionPosition' variable.
acme.js	few seconds ago	Remove the declaration of the unused 'separator' variable.
acme.js	few seconds ago	Extract this nested ternary operation into an independent statement.
acme.js	few seconds ago	Refactor this function to reduce its Cognitive Complexity from 19 to the 15 allowed. [+9 locations]
areyousure.js	few seconds ago	Refactor this function to reduce its Cognitive Complexity from 56 to the 15 allowed. [+23 locations]
areyousure.js	few seconds ago	Add the "let", "const" or "var" keyword to this declaration of "$dirtyForms" to make it explicit.
areyousure.js	few seconds ago	Add the "let", "const" or "var" keyword to this declaration of "$fields" to make it explicit.
bootstrap.bundle.js	few seconds ago	Complete the task associated to this "TODO" comment.
bootstrap.bundle.js	few seconds ago	'Popover' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'Popover' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'ScrollSpy' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'ScrollSpy' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'Tab' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'Tab' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'Tooltip' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'Tooltip' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'complete' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'hide' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'i' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'i' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	'update' is already declared in the upper scope.
bootstrap.bundle.js	few seconds ago	This function expects 1 argument, but 2 were provided. [+2 locations]
bootstrap.bundle.js	few seconds ago	Complete the task associated to this "TODO" comment.
bootstrap.bundle.js	few seconds ago	Complete the task associated to this "TODO" comment.
bootstrap.bundle.js	few seconds ago	Expected a `for-of` loop instead of a `for` loop with this simple iteration.
bootstrap.bundle.js	few seconds ago	Expected a `for-of` loop instead of a `for` loop with this simple iteration.
bootstrap.bundle.js	few seconds ago	Expected a `for-of` loop instead of a `for` loop with this simple iteration.
bootstrap.bundle.js	few seconds ago	Expected a `for-of` loop instead of a `for` loop with this simple iteration.

Ilustración 1: Fragmento de "Bad Smells" encontrados

Una vez detectado en que línea se indicaba que se realizara la corrección, se procedió con a esta. La solución era muy simple, se trataba de eliminar la declaración de esa variable y devolver directamente el valor que tenía dicha variable. Esta corrección se puede observar en la **ilustración 3,** que se puede observar la diferencia con la **ilustración 2.**

```
@Override
public SystemConfiguration findOne(final Request<SystemConfiguration> request) {
    assert request != null;
    final SystemConfiguration sc = this.repository.findInitialConfiguration();
    return sc;
}
```

Ilustración 2: Fragmento de código donde se encontró el "Bad Smell"

```
@Override
public SystemConfiguration findOne(final Request<SystemConfiguration> request) {
    assert request != null;
    return this.repository.findInitialConfiguration();

}
```

Ilustración 3: Fragmento de código una vez solucionado

El resto de “Bad Smells” que aparecen en la **ilustración 1** se detectó que procedían de ficheros del framework, por lo que se tomó la decisión por parte del equipo de desarrollo de mantener los archivos como están.

5. Conclusión

Para concluir el informe, se puede decir que el equipo de desarrollo ha aprendido la importancia que tiene el uso de herramientas como SonarLint, ya que su ayuda a la hora de proporcionar feedback de forma temprana sobre la calidad del código es muy importante para los proyectos como es el caso de Acme-Recipes. Gracias a ella se pudo detectar problemas a nivel de calidad sobre el proyecto y su solución rápida. Además, con SonarLint los miembros del equipo pueden comprender los “Bad Smells” y demás errores existentes, ya que proporciona una descripción detallada sobre estos, así como una solución recomendada.
Por lo tanto, el equipo se ha dado cuenta que SonarLint es un gran aliado en la entrega de código limpio y seguro. Por ello, el uso de dicha herramienta será parte del proyecto en todo el ciclo de desarrollo.

6. Bibliografía

Intencionalmente en blanco.