Informe de Linting

Integrantes del grupo C1.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* Adrián Cabello Martín ([adrcabmar@alum.us.es](mailto:adrcabmar@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))
* Natalia Olmo Villegas ([natolmvil@alum.us.es](mailto:natolmvil@alum.us.es))

Fecha: Sevilla, 26 de Abril 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc164969475)

[Historial de Versiones 4](#_Toc164969476)

[Introducción 5](#_Toc164969477)

[Listado de bad smells 6](#_Toc164969478)

[Literal repetido demasiadas veces 6](#_Toc164969479)

[Override “equals” method 7](#_Toc164969480)

[Replace assert with a proper check 8](#_Toc164969481)

[Conclusiones 9](#_Toc164969482)

[Bibliografía 10](#_Toc164969483)

# Resumen del Informe

Este informe incluye describe los malos olores del código producido por el estudiante 1 en el tercer entregable. El análisis se ha hecho con ayuda de la herramienta SonarLint (plug-in de Eclipse). Sólo se explicarán aquellos malos olores o bad smells inofensivos para el proyecto. Los de mayor severidad se habrán corregido y no será necesario mencionarlos en el presente informe.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.1 | Estructura del informe. Resumen. Bibliografía. | 25/04/2024 | María de la Salud Carrera Talaverón |
| V1.0 | Listado de bad smells. | 25/04/2024 | María de la Salud Carrera Talaverón |
| V1.1 | Introducción. Conclusión. Bibliografía. | 25/04/2024 | María de la Salud Carrera Talaverón |

# Introducción

Este informe de “Linting” consiste en describir y analizar los malos olores encontrados en el código del estudiante 1, con ayuda de la herramienta SonarLint. Para cada code smell se menciona su causa, sus posibles soluciones, su localización en el código y una justificación de por qué no se han solucionado.

En concreto, se han encontrado tres tipos de malos olores: repetición excesiva de un literal, no sobreescritura del método “equals” y uso del tipo primitivo assert en lugar de una sentencia condicional. Cada uno de ellos está descrito en un apartado.

Finalmente, se proporciona una conclusión del informe.

# Listado de bad smells

Sólo se van a analizar los bad smells implementados en el código de las tareas del estudiante 1. Los introducidos en el código original del proyecto, por los otros estudiantes o en las tareas grupales no se tendrán en cuenta.

## Literal repetido demasiadas veces

Este bad smell se produce cuando, en un mismo documento, repetimos un mismo literal demasiadas veces, en lugar de definir una constante que almacene el valor de dicho literal y utilizar esa constante.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteSolucionar este mal olor ayudaría a futuras modificaciones del código, pues, si fuera necesario modificar el literal que se repite, sólo habría que cambiar su valor en la definición de la constante. De lo contrario, habría que modificar cada una de las apariciones del mismo.

Se puede encontrar bad smells de este tipo en los siguientes archivos: ManagerAssignationCreateService, ManagerAssignationDeleteService, ManagerProjectPublishService, entre otros.

Como se puede observar en la imagen, la mayoría de las veces que aparece este bad smell es en los servicios de creación, actualización, publicación y borrado de entidades, puesto que se repiten numerosas veces los nombres de los atributos de estas. Se ha optado por no solucionar este bad smell porque, aunque se ganaría en mantenibilidad, se perdería en comprensión del código.

## Override “equals” method

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteEste tipo de bad smell se da, sobre todo, en las entidades del proyecto, puesto que, en general, en Java se debería sobrescribir el método “equals” en todas las clases que se creen, para personalizar el tratamiento de varios objetos de la misma entidad. Sin embargo, en la mayoría de casos no es necesario hacerlo, puesto que se suele necesitar que dos objetos compartan los mismos valores de sus atributos para ser iguales.

En nuestro caso, las entidades que hemos definido extienden a AbstractEntity, que ya tiene definido un método equals personalizado. Por tanto, este “bad smell” en realidad ya estaría cubierto.

## Replace assert with a proper check

Imagen que contiene Calendario

Descripción generada automáticamenteEste tipo de bad smell se da cuando utilizamos el tipo primitivo “assert” para definir una condición o predicado, en lugar de utilizar una sentencia “if” que ejecute el bloque de código en caso necesario o que lance una excepción.

Lo encontramos casi exclusivamente en los métodos de nuestros servicios. Se ha optado por no corregir este mal olor debido a que es así cómo viene explicado en las transparencias de la asignatura.

# Conclusiones

Cabe destacar que, aunque el número de malos olores encontrados en el análisis ha sido significativo, eran todos de únicamente tres tipos distintos, lo que lleva a concluir que esta métrica del código ha sido positiva.

Durante el análisis, se solucionó un bad smell diferente y, por tanto, no aparece mencionado en este informe.

# Bibliografía

Documentación de SonarLint.