Informe Lint

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](#_Toc167735914)

[Historial de Versiones 4](#_Toc167735915)

[Introducción 5](#_Toc167735916)

[Listado de bad smells 6](#_Toc167735917)

[Literal repetido demasiadas veces 6](#_Toc167735918)

[Replace assert with a proper check 8](#_Toc167735919)

[Conclusiones 9](#_Toc167735921)

[Bibliografía 10](#_Toc167735922)

# Resumen del Informe

Este informe describe los malos olores del código producido por el estudiante 2 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 | Informe Lint | 25/04/2024 | Natalia Olmo Villegas |

# Introducción

Este informe Lint consiste en describir y analizar los malos olores encontrados en el código del estudiante 2, con ayuda del plug-in de SonarLint para Eclipse. 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 dos tipos de malos olores: repetición en numerosas ocasiones de un literal y uso del tipo primitivo assert en lugar de una sentencia condicional. Cada uno de ellos está descrito en un apartado.

# Listado de bad smells

## 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

Descripción generada automáticamenteTexto

Descripción generada automáticamente

En este caso, vemos que se repite la palabra **budget** 7 veces. Solucionar este mal olor ayudaría a una refactorización más rápida y futuras modificaciones del código, pues, en casa de que se deba modificar el literal que se repite, sólo habría que cambiar su valor en la definición de la constante, en vez de hacerlo en sus múltiples apariciones, ocasionando posibles despistes e incoherencias si no se cambian todos.

Se puede encontrar bad smells de este tipo en las siguientes clases:

* ClientContractCreateCreateService
* ClientContractUpdateService
* ClientContracPublishService.

Este error vuelve a aparecer en las clases

* ClientProgressLogCreateCreateService
* ClientProgressLogUpdateService
* ClientProgressLogPublishService,

Pero esta vez con las palabras **completeness**, **recordId** y **registrationMoment**, respectivamente. En el listado de los informes de progreso también se duplica **masterId**

Se ha decidido no solucionar este bad smell porque, aunque se ganaría en mantenibilidad, se aumentaría la complejidad cognitiva, haciendo más difícil su comprensión.

## Replace assert with a proper check

## Imagen que contiene Calendario Descripción generada automáticamente

Este 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 en los métodos de nuestros servicios y se ha recomendado por parte del profesorado que debe ser añadido para manejar la excepción en caso de que el proyecto converja en un error irreversible(no error 500). Se ha optado por no corregir este mal olor debido a que es así cómo viene explicado en las transparencias de la asignatura.

# Conclusiones

El número de malos olores encontrados no ha sido elevado y todo eran de dos tipos únicamente: la repetición excesiva de literales y el uso inadecuado de "assert" en lugar de comprobaciones apropiadas, lo que lleva a concluir que esta métrica del código ha sido positiva.

Aunque se identificaron áreas de mejora en el código, se optó por mantener ciertas prácticas para no aumentar la complejidad de este, priorizando la facilidad de comprensión por parte de los desarrolladores.

# Bibliografía

Documentación de SonarLint.