Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

SONAR LINT REPORT INDIVIDUAL - D03



**C1.027**

**Repositorio**: <https://github.com/josrojrom1/DP2-G27-Acme>

**Autor:**

Rojas Romero, José Joaquín

josreimun@alum.us.es

**Fecha:** 25/04/2024

**Tabla de contenidos**

[**Resumen**](#_heading=h.gjdgxs) **2**

[**Tabla de revisión**](#_heading=h.30j0zll) **2**

[**Introducción**](#_heading=h.1fob9te) **2**

[**Contenidos**](#_heading=h.3znysh7) **3**

[**Bibliografía**](#_heading=h.3dy6vkm) **3**

# Resumen

Este documento se trata de un listado de registros de malos olores con la herramienta Sonar Lint. En caso de detectar malos olores, se decidirá su corrección o no en función de si es importante o no el registro en cuestión para el correcto funcionamiento del código y su legibilidad.

# Tabla de revisión

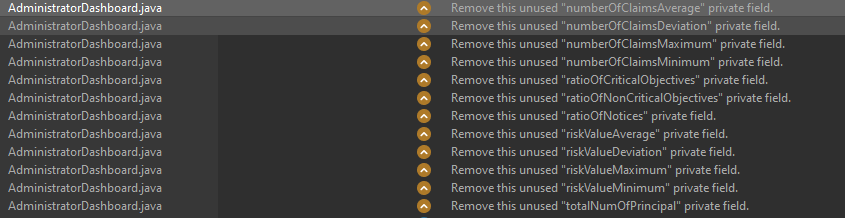
| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| V1 | 25/04/2024 | Creación y desarrollo del documento |

# Introducción

Con motivo de detectar malos olores, y su posible corrección, se han aportado capturas del análisis ejecutado con Sonar Lint y se ha realizado un análisis de cada captura con la decisión al respecto.

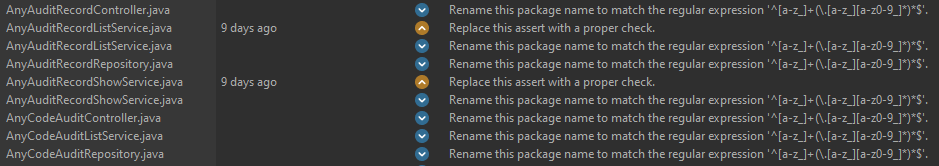
# Malos olores detectados tras análisis y valoración

Se ha ejecutado un análisis del proyecto individual completo, obteniendo los siguientes malos olores, los cuales se mostrarán mediante capturas del propio log generado por la herramienta:

**Captura 1**. AdministratorDasboard.java bad smells

En la **Captura 1** podemos comprobar un aviso relacionado con cada una de las variables declaradas en la clase ***AdministratorDashboard*** del proyecto, debido a que no se están usando.

**Valoración**: Debido a que la ejecución del análisis se ha hecho sobre mi proyecto individual, han aparecido estos avisos relacionados con tareas de desarrollo grupales relacionadas con el *dashboard* del rol *Administrator*, las cuales están en un estado más avanzado en el proyecto grupal de GitHub, por lo que no se tendrán en cuenta dichos malos olores al desaparecer tras la posterior inclusión de mi rama individual a la rama grupal, y por tanto, ser empleadas en las consultas pertinentes.



**Captura 2**. AnyAuditController, Service, Show and List bad smells

En la **Captura 2** observamos varios avisos idénticos relacionados con el seguimiento de una expresión regular a la hora de nombrar las clases de cualquier rol principal sobre auditorias y registros de auditoría. También se muestran (con la flechita naranja) dos malos olores relacionados con el uso de ***asserts*** en los métodos de los servicios.

**Valoración**: El nombre de las clases sigue las recomendaciones indicadas por el profesor, donde empiezan por mayúsculas (el único caso que rompe la expresión regular recomendada por el análisis) al indicar el rol en concreto que puede realizar los servicios especificados, por lo que no se tendrán en consideración dichos malos olores. Por otro lado, los malos olores que nos recomiendan reemplazar los ***asserts*** por ***checks*** no serán tenidos en cuenta, pues en el framework que empleamos es el modo en el que se comprueban todos los objetos usados en los respectivos métodos son distintos de ***null*** y no se emplean para nada más. Esta valoración será tenida en cuenta en todos los malos olores relacionados con este caso en el análisis, por lo que se mostrará solo en este ejemplo de la **Captura 2,** al existir dichos malos olores varias veces en todos los servicios desarrollados.

**Captura 3**. Constants suggestion in *authorise, bind* and *unbind* methods from all services

En la **Captura 3** observamos un aviso en rojo que nos sugiere definir constantes para los parámetros que se pasan en los métodos *authorise*, *bind* y *unbind* de todos los servicios, por lo que solo se mostrará esta captura para representar todas las apariciones de dicha advertencia para mejorar la legibilidad del documento.

**Valoración:** En la manera en la que están diseñados los métodos que reciben parámetros como *masterId* para extraerlos de la petición o ingresarlos en otro método que lo requiera (como los atributos de una entidad que son pasados a una vista o devueltos en un rollback), se espera que dichos parámetros se pasen en formato cadena, esto es, entre comillas dobles. Si creamos una constante podríamos estar añadiendo una posible vulnerabilidad al tener que controlar perfectamente el uso que se le da a dicha variable. Por estos motivos y por que se comentó en clase que no hay que declarar constantes en estos casos, se han ignorado dichos code smells relacionados con el uso de constantes.

**Captura 4**. Replace Collectors.toList() to Stream.toList()

En la **Captura 4** observamos un aviso en naranja que nos sugiere reemplazar el uso del método *Collectors.toList()* a Stream.toList().

**Valoración:** Se ha probado a aplicar la sugerencia, pero se producía un error en tiempo de compilación. Esto se debía a que el final del método implementado en la clase *AnyCodeAuditShowService.java* para calcular la puntuación (*mark*) requiere de un tipo *Collectors* que se le pasa al método .*collect* de dicha función. Debido a esto no se ha tenido en cuenta el mal olor y se ignorará en próximos entregables.

Finalmente, a modo de resumen, comentar que no ha hecho falta añadir cambios en el código tras el análisis del proyecto individual con Sonar Lint, pues no había ningún mal olor preocupante. Para realizar el análisis se ha tenido en cuenta las respuestas en el foro realizadas sobre las poca información teórica acerca de los malos olores y el uso de Sonar Lint. Para ver los detalles puede ingresar al siguiente enlace al foro con más información: <https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_85092_1&nav=discussion_board&conf_id=_405265_1&forum_id=_234042_1&message_id=_415711_1>

# Bibliografía

-Documento 08 Annexes, de los contenidos de la plataforma virtual de esta asignatura.