Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

**Lint Report**



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2.

Curso 2023 – 2024

| **Fecha** | **Versión** |
| --- | --- |
| 21/05/2024 | v1.0.0 |

| **Grupo: C1.011** | | |
| --- | --- | --- |
| **Autores por orden alfabético** | **Rol** | **Descripción del rol** |
| Abouri, Mohamed - Y7156458E | Developer | Persona encargada de desarrollar el código. |
| Cabello Ranea, Rafael -32094496C | Operador | Encargado de las tareas de campo, de las instalaciones y del mantenimiento de los sistemas de la empresa. |
| Calderón Rodríguez, Francisco Javier - 78233219F | Project Manager | Persona encargada de tomar decisiones de diseño y vigilar el correcto desarrollo |
| Delgado Pallares, David - 29519510E | Tester | Persona encargada de realizar pruebas sobre el código. |
| Ramírez Lara, Iván- 77852382G | Developer | Persona encargada de desarrollar el código. |

**Control de Versiones**

| **Fecha** | **Versión** | **Descripción** |
| --- | --- | --- |
| 21/05/2024 | v1.0.0 | Desarrollo de la primera versión. |
|  |  |  |
|  |  |  |
|  |  |  |

**Índice de contenido**

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

2[.](#_heading=h.30j0zll) Contenido 2

3[. Conclusiones](#_heading=h.1fob9te) 3

4[. Bad smells](#_heading=h.1fob9te) 4

**1. Introducción**

Este documento presenta una lista de los “bad smells” (códigos problemáticos o antipatrón) identificados por SonarLint en nuestro proyecto. SonarLint es una herramienta de análisis estático que detecta posibles errores, problemas de calidad y malas prácticas en el código. Sin embargo, no todos los problemas detectados necesariamente afectan la funcionalidad o el rendimiento de nuestro software de manera significativa.

En las siguientes secciones, se enumeran los “bad smells” reportados, proporcionando una descripción detallada de cada uno, así como la ubicación específica en el código. Además, se ofrece una justificación para cada caso, explicando por qué consideramos que estos problemas son inocuos y no requieren intervención inmediata.

El propósito de este documento es asegurar la transparencia en el mantenimiento del código, demostrar nuestra comprensión de las alertas generadas por SonarLint y justificar las decisiones tomadas respecto a la necesidad de corregir o no cada uno de estos problemas. Con ello, buscamos mantener un equilibrio entre la calidad del código y la eficiencia en el desarrollo, evitando cambios innecesarios que no aporten un valor relevante al proyecto.

**2. Contenido**

| **Clase** | **Línea** | **Descripción** | **Justificación** |
| --- | --- | --- | --- |
| AnyPublishedProjectListService | 38 | Replace this assert with a proper check | Este bad smell no ha sido debido a nuestra implementación, puesto que la línea que da problemas es necesaria tenerla en nuestras clases. |
| AnyPublishedProjectShowService | 43 | Replace this assert with a proper check | Este bad smell no ha sido debido a nuestra implementación, puesto que la línea que da problemas es necesaria tenerla en nuestras clases. |

****

**Advertencia**:

Todos los “code smells” no pertenecientes a clases implementadas por el estudiante 1 no han sido analizados, puesto que no son responsabilidad de este la mala implementación de código o él no usó de buenas prácticas, por lo que este solo puede garantizar que para la realización de su código se ha usado el feedback recibido en clase y las buenas prácticas comentadas en estas.

**3.** [**Conclusiones**](#_heading=h.1fob9te)

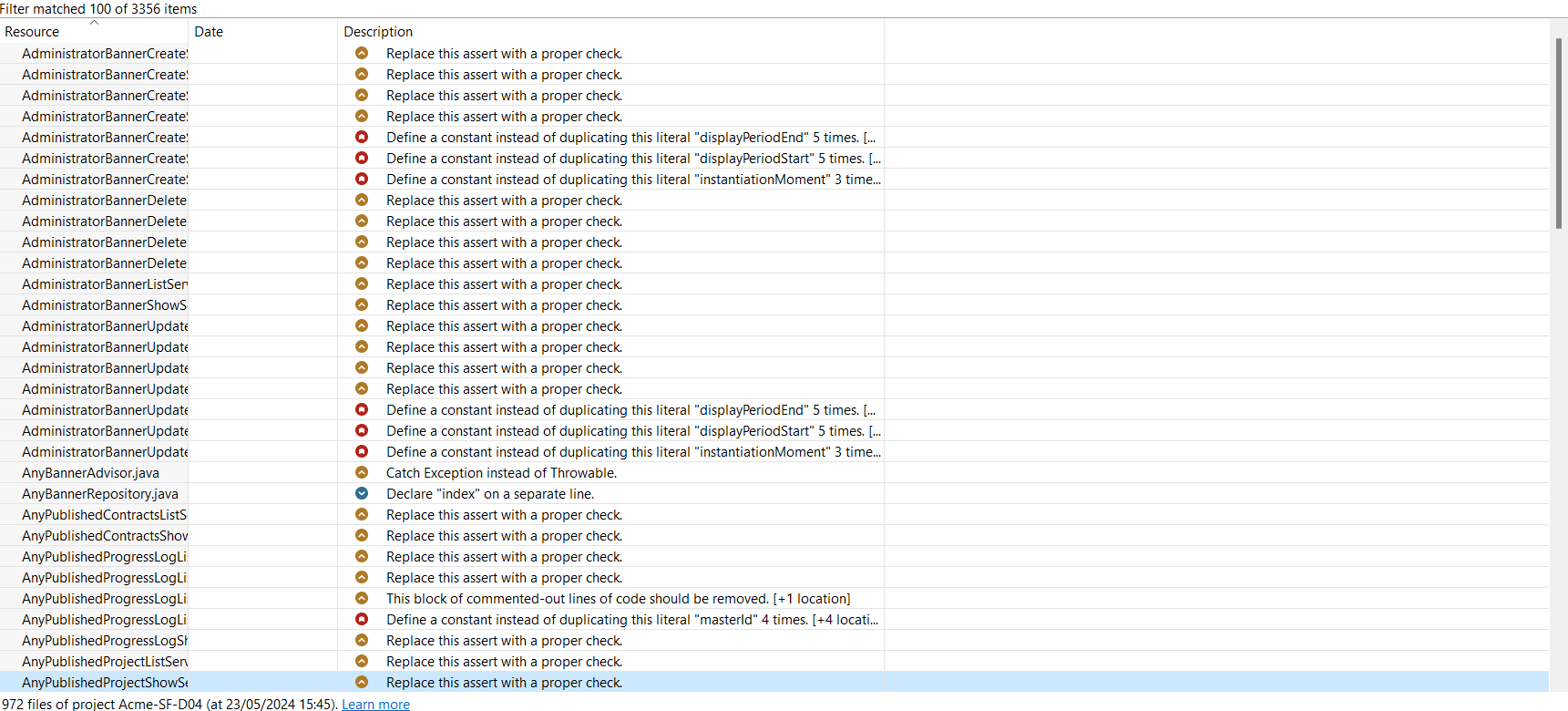
En conclusión, aunque nuestro código no está completamente libre de “code smells”, es importante destacar que muchos de estos no son el resultado de malas prácticas o errores de nuestra parte. Las líneas de código que generan estas alertas son, en su mayoría, necesarias para el correcto funcionamiento del framework que estamos utilizando, y se han implementado siguiendo las especificaciones y directrices proporcionadas por los docentes en clase.

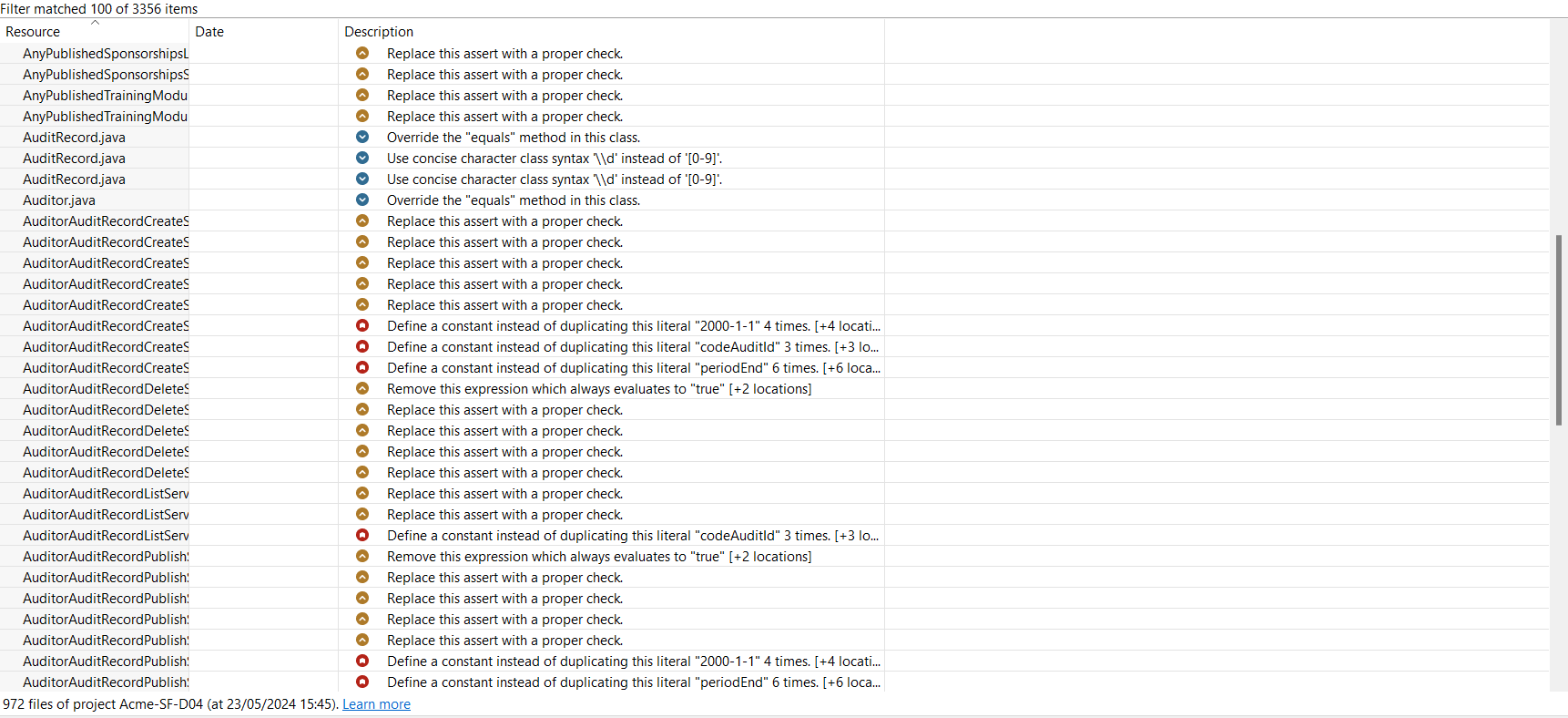
La decisión de mantener estos “code smells” está respaldada por una evaluación cuidadosa de su impacto en la funcionalidad y rendimiento de la aplicación. En muchos casos, las modificaciones sugeridas por SonarLint podrían comprometer la estabilidad o el comportamiento esperado del software, por lo que hemos optado por priorizar la adherencia a los requisitos del framework y las prácticas recomendadas.

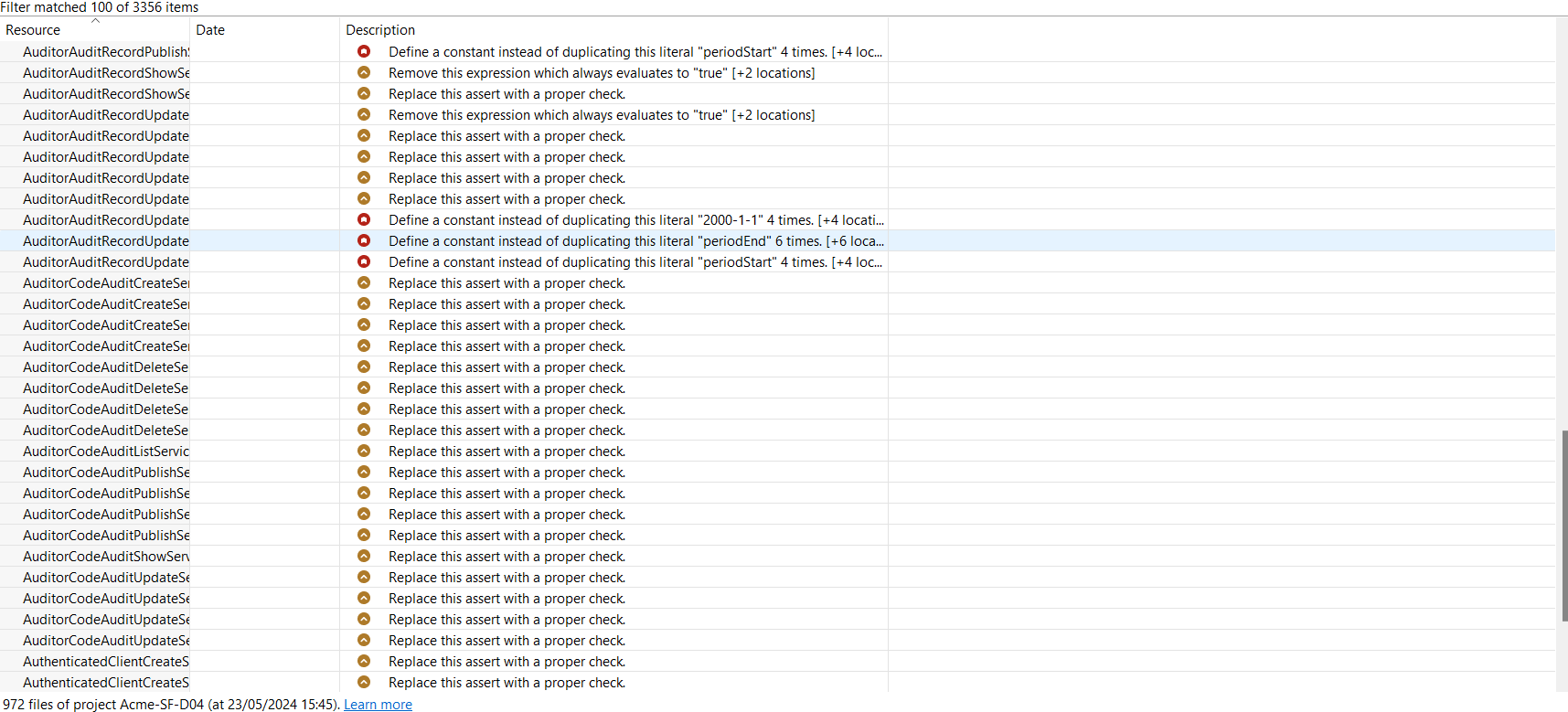
Nuestro compromiso con la calidad del código sigue siendo una prioridad, y continuamos revisando y mejorando nuestro trabajo de manera constante. La presencia de estos “code smells” no afecta negativamente la eficiencia o la fiabilidad de nuestra aplicación, y hemos tomado las medidas necesarias para asegurar que cualquier posible riesgo esté adecuadamente mitigado. Agradecemos su comprensión y estamos abiertos a discutir cualquier preocupación o sugerencia adicional para mejorar nuestro proceso de desarrollo.

**4.** **Bad Smells**

Como se demuestra en las siguientes imágenes, la mayoría de los “code smell” provienen de clases implementadas por otros estudiantes, es por esto por lo que el análisis de los “code smells” del estudiante 1 es escueto en comparación con otros estudiantes.



****

****

****