Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



**Grado en Ingeniería Informática – Ingeniería del Software**  
**Diseño y Pruebas II**

Curso 2023 – 2024

|  |  |
| --- | --- |
| **Fecha** | **Versión** |
| 25/04/24 | 1.0 |

|  |  |
| --- | --- |
| **Grupo de Prácticas: C1.012** | |
| **Repositorio:** [https://github.com/DP2-C1-012/Acme-SF](mailto:https://github.com/DP2-C1-012/Acme-SF) | |
| **Autores por orden alfabético** | **Correo** |
| Bernal Caunedo, Álvaro Jesús | [alvbercau@alum.us.es](mailto:alvbercau@alum.us.es) |
| Caballero Hernández, Jaime | [jaicabher1@alum.us.es](mailto:jaicabher1@alum.us.es) |
| Casamitjana Benítez, Juan José | [juacasben@alum.us.es](mailto:juacasben@alum.us.es) |
| Herrera Lobo, Nicolás | [nicherlob@alum.us.es](mailto:nicherlob@alum.us.es) |
| Montoya Albitres, Ronald Alexander | [ronmonalb@alum.us.es](mailto:ronmonalb@alum.us.es) |

**Índice de contenido**

[**1.** **Resumen Ejecutivo** 2](#_Toc165018953)

[2. **Tabla de revisiones** 2](#_Toc165018954)

[**3.** **Introducción** 3](#_Toc165018955)

[**4.** **Contenido** 4](#_Toc165018956)

[**5.** **Conclusiones** 6](#_Toc165018957)

[**6.** **Bibliografía** 6](#_Toc165018958)

# **Resumen Ejecutivo**

El informe de Lint indica que el proyecto carece de problemas mayores, identificando solo algunos "code smells" menores que no obstaculizan el progreso. Estos aspectos podrían ser abordados en futuras iteraciones para mejorar la calidad del código, pero no representan riesgos significativos para el desarrollo actual.

# **Tabla de revisiones**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Revisión** | **Descripción** |
| 25/04/2024 | 1.0 | Versión Inicial |

# **Introducción**

El informe de Lint proporciona una evaluación exhaustiva del código del proyecto, destacando su robustez general y su solidez estructural. Esta revisión detallada ofrece una visión clara de la salud del código y sugiere áreas potenciales para futuras mejoras.

En este informe se aborda el contenido del código, examinando cada clase identificada con "code smells" y evaluando su relevancia para el proyecto. Por último, se incluye una sección de conclusión en la que se resumen las decisiones tomadas en base al informe.

# **Contenido**

|  |  |  |
| --- | --- | --- |
| **Tipo de Code Smell** | **Archivos** | **Análisis** |
| Use isEmpty() to check whether the collection is empty or not. | * AuditorDashboardShowService | Funcionalmente es indiferente usar isEmpty() a una comprobación booleana sobre size(). |
| Replace this usage of 'Stream.collect(Collectors.toList())' with 'Stream.toList()' | * AuditorDashboardShowService | Funcionalmente es indiferente, excepto en el caso de que se quiera una lista modificable o no, que no es el caso. |
| Refactor this method to reduce its Cognitive Complexity. | * AuditorDashboardShowService | El Código no tiene porque ser refactorizado a menos que el cliente quiera otro tipo de estadísticas por lo que cambiar la complejidad cognitiva no supone un esfuerzo aprovechable. |
| Replace the type specification in this constructor call with the diamond operator ("<>"). | * AuditorDashboardShowService | Funcionalmente es indiferente. |
| Replace this assert with a proper check. | * AuthenticatedAuditorCreateService * AuthenticatedAuditorUpdateService * AuditorAuditRecordDeleteService * AuditorCodeAuditDeleteService | Las funciones que usan assert solo están ahí para que el framework las acepte como implementadas ya que de otra forma no me dejaba dejarlas vacías ya que en estas situaciones no me hace falta implementar la función. |
| Define a constant instead of duplicating this literal. | * AuditorAuditRecordCreateService * AuditorAuditRecordUpdateService | Es major ser directamente explícito con estos literales para futuras modificaciones. |

# **Conclusiones**

En conclusión, el análisis del código revela que se han identificado muy pocos "code smells" relacionados directamente con los requisitos del proyecto. Estos hallazgos indican que el código se alinea eficazmente con las especificaciones establecidas, lo que refuerza la calidad y la coherencia del desarrollo. Dado el bajo número y la naturaleza menor de los "code smells" detectados, no se justifica la realización de acciones de mejora inmediatas. En general, la solidez del código sugiere que el proyecto está bien encaminado y puede avanzar con confianza hacia su siguiente fase. Este análisis reafirma la calidad del trabajo realizado hasta la fecha y destaca la importancia de mantener estándares elevados en el desarrollo de software.

# **Bibliografía**

Intencionalmente en blanco.