Universidad de Sevilla  
 Escuela Técnica Superior de Ingeniería Informática

**Documentación de la entrega D03**

**Informe del Lint Report**



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

Curso 2023 – 2024

|  |  |
| --- | --- |
| **Fecha** | **Versión** |
| 26/4/2024 | v1r2 |

|  |  |  |
| --- | --- | --- |
| **Grupo de prácticas: C1.033** | | |
| **Autores por orden alfabético** | **Rol** | **Correo electrónico** |
| Aguayo Orozco, Sergio - 25604244T | Desarrollador | ahydul1@gmail.com |
| García Lama, Gonzalo - 47267072W | Desarrollador, Tester | gongarlam@alum.us.es |
| Huecas Calderón, Tomás - 17476993Y | Desarrollador | tomhuecal@alum.us.es |
| Fernández Pérez, Pablo - 54370557Y | Desarrollador,  Analista | pablofp.33@gmail.com |
| Youssafi Benichikh, Karim -28823709V | Desarrollador,  Operador,  Manager | karyouben@alum.us.es |

## Repositorio: <https://github.com/karyouben/Acme-SF-D03>

**Índice de contenido**

[**1. Resumen ejecutivo 2**](#_d5mdjbdbxjy1)

[**2. Tabla de versión 2**](#_qy8e39e8j64k)

[**3. Introducción 2**](#_dghwxxs0hw5m)

[**4. Contenido 3**](#_45jzfp9onkwb)

[**5. Conclusiones**](#_7188hylxrer9) **6**

[**6. Bibliografía**](#_2bezepxcuzge) 7

# 1. Resumen ejecutivo

Este informe detalla el análisis llevado a cabo para el código realizado en la entrega 3 de dp2. Este análisis proporciona una mejora de la calidad del código, identificando posibles malas prácticas, malos olores (code smells) o incluso algunos bugs.

# 2. Tabla de versión

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 25/3/2024 | v1r0 | creación del documento de análisis individual |
| 16/4/2024 | v1r1 | Añadido resumen ejecutivo e introducción |
| 26/4/2024 | v1r2 | Añadido contenido y conclusiones |
|  |  |  |

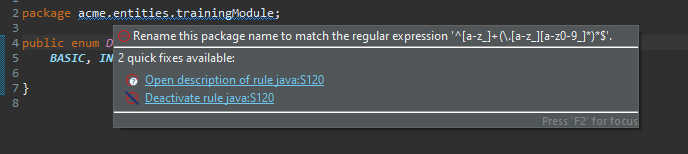
# 3. Introducción

En esta tercera del proyecto se han realizado un análisis exhaustivo de todo el código implementado en las entidades, servicios, controladores y querys, además del front, para mejorar la calidad y seguir las buenas practicas durante el desarrollo de dicho proyecto.  
  
Estas buenas prácticas no solo mejoraran la legibilidad del código, sino que asegurara la calidad del mismo.

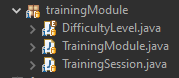
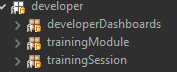
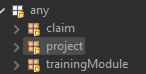
# 4. Contenido

Al analizar el proyecto con SonarLint, se detectaron los siguientes malos olores:

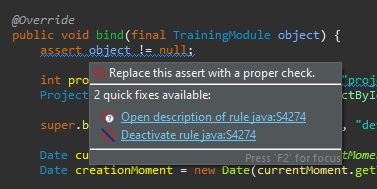
**-Nombres de paquetes sin seguir una convención de nombre.**



Este mal olor (code smell) en mi código esta presente en el nombrado de los paquetes trainingModule, trainingSession y developerDashboards del features y en el trainingModule en la entidad además del trainingModule del any, pues uso la nomenclatura camelCase pero debería escribir los paquetes en minúscula.

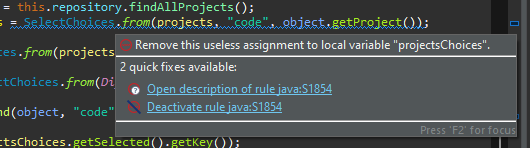
  

**-Asserts comprobando parámetros de un método público:**

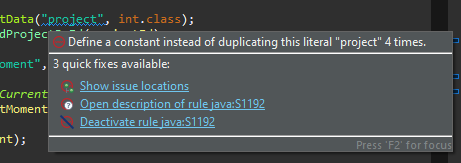
****

Este mal olor (code smell) en mi código está presente en los paquetes trainingModule, trainingSession , developerDashboards y authenticated del features y en el trainingModule del any, pues uso una clase publica para comprobar el assert en lugar de usar una en la clase privada, este codeSmell no es algo grave, ya que prefiero usar el assert de una clase compartida.  
  
  
**-uso de variables que no se usan**

Este tipo de variables que no se usan generan código basura y dificultan la lectura del código por lo que es indispensable eliminarlos para tener un código limpio y de calidad

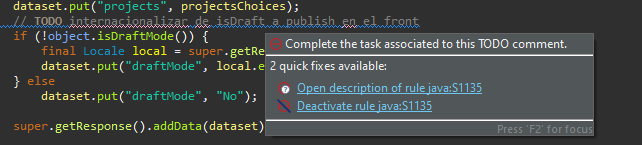
  
  
**-strings repetidos**

Uso de strings repetidos en las clases.



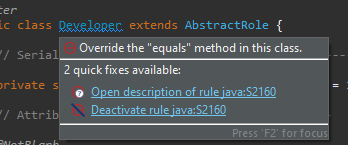
**-Comentario TODO**

Uso del comentario // TODO para implementar una internacionalización en el front para buenas practicas de código en la siguiente entrega, esto es un comentario realizado para tener en cuenta un fix futuro, por lo que no debería considerarse como un code Smell, pero es un buen aviso o recordatorio.

****

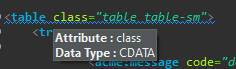
**-Uso del método equals de una clase publica**

Esto como tal no lo considero tampoco un code Smell ya que es mejor coger el equals de una clase compartida que me interesa que tenga las mismas características en el equals, en este caso del abstractRole.

****

**-Uso del atribute CDATA**

Este no entiendo bien porque sale, ya que lo uso para crear tablas en jsp, así que entiendo que será otro code smell que no debería catalogarse como tal.

****

# 5. Conclusiones

La implementación de herramientas de análisis estático de código, como el lint, ha tenido un impacto significativo en la calidad del código de nuestro proyecto. El uso sistemático del lint ha permitido no solo mantener un alto estándar de calidad, sino también fomentar mejores prácticas de codificación entre los miembros del equipo. Con ello ha habido una disminución en la cantidad de errores de sintaxis y problemas de estilo, lo que ha llevado a un código más limpio y eficiente.

Con todo esto, también se ha incrementó la coherencia en todo el código base, lo cual facilita la legibilidad del código y la colaboración. La automatización del lint como parte de nuestro proceso de integración continua es algo vital para identificar problemas antes de que el código llegue a producción, mejorando así nuestra capacidad de respuesta y agilidad.

En futuros proyectos, planeamos integrar el lint desde las primeras fases de desarrollo para cultivar una base de código robusta desde el comienzo. Aunque también hay que tener en cuenta que no todo lo que encuentra este programa como code smell es realmente un mal olor, por lo que hay que ser analítico de saber cuando realmente un fragmento de código es un code smell o es realmente necesario, como el uso del //TODO de comentario para próximas implementaciones.

# 6. Bibliografía

Intencionalmente en blanco.