01-06-2022

Lint Report

D04 – Diseño y Pruebas II

*Lint report*

Grupo E6-06 – Version 4.0

CERRATO SÁNCHEZ, LUIS   
ESCALERA MARTÍN, REGINA  
GUITÉRREZ CONTRERAS, ERNESTO   
MARIAN STEFAN, BOGDAN  
MARTÍNEZ SUÁREZ, DANIEL JESÚS

https://github.com/danmarsua1/Acme-One.git

**Índice**

[Índice 1](#_heading=h.17dp8vu)

[Resumen 2](#_heading=h.3rdcrjn)

[Historial de versiones 2](#_heading=h.44sinio)

[Introducción 2](#_heading=h.26in1rg)

[Contenido 2](#_heading=h.lnxbz9)

[Conclusiones 2](#_heading=h.35nkun2)

[Bibliografía 2](#_heading=h.1ksv4uv)

**Resumen**

Este documento es el informe ‘lint’ de nuestro proyecto de la asignatura de Diseño y Pruebas II.

En dicho informe vamos a mostrar un análisis sobre problemas de calidad que tengamos en la aplicación desarrollada.

**Historial de versiones**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción de cambios** |
| 01/06/2022 | 1.0 | Creación del documento |
| 02/06/2022 | 2.0 | Añadido contenido y conclusiones |

**Introducción**

Como dijimos en el resumen, vamos a indicar aquellos errores ó elementos a mejorar que tenemos en el proyecto, pueden ser:

-Errores de sintaxis

-Código poco intuitivo

-El uso de malas prácticas

Una de las primeras herramientas que se desarrollaron para este cometido fue Lint, que se utilizaba para detectar código sospechoso en programas escritos en lenguaje C.

Para nuestro caso, vamos a utilizar la herramienta SonarLint desde eclipse, una extensión IDE que identifica y ayuda a solucionar los problemas de calidad y seguridad, además proporciona una guía de corrección para poder presentar un código limpio.

**Contenido**

Se adjunta una captura acerca del resultado del análisis de la herramienta SonarLint sobre el proyecto Acme-One definitivo.



**Conclusiones**

Tras realizar el análisis, hemos comprobado que tenemos 46 bugs (algo menos de la mitad de ellos son de nivel de criticidad alto) sobre 625 ficheros del proyecto completo, donde se encontraban una cantidad más elevada en anteriores entregas; No obstante, los que se muestran no se han podido llegar a solucionar.

A modo de conclusión final, haciendo una observación superficial del análisis, se puede finalizar con que el análisis, en todo caso, no es concluyente negativamente. La razón de esto es porque, incluso haciendo foco en los “code smells” críticos que categoriza SonarLint, estos se pueden solucionar de manera simple, además de que no suponen ninguna falla de seguridad importante.

**Bibliografía**

“Intentionally blank”