	Diseño y Pruebas 2 Documentación de la entrega D03
	Documentación de SonarLint

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Documentación de la entrega D03

Documentación de SonarLint



Grado en Ingeniería Informática – Ingeniería del Software


Diseño y Pruebas 2

Curso 2023 – 2024

Fecha	Versión
26/4/2024	v1r1

Grupo de prácticas: C1-033		
Autor	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
Yousaf Benichikh, Karim - 28823709V	Desarrollador, Operador, Mánager	karyouben@alum.us.es

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


	Diseño y Pruebas 2 Documentación de la entrega D03
	Documentación de SonarLint

Control de Versiones

Fecha	Versión	Descripción
24/4/2024	v1r0	Inicialización del documento
26/4/2024	v1r1	Finalización del documento y revisión

Índice de contenido

1. Resumen ejecutivo	2
2. Introducción	2
3. Contenido	3
4. Conclusiones	4
5. Bibliografía	4

	Diseño y Pruebas 2 Documentación de la entrega D03
	Documentación de SonarLint

1. Resumen ejecutivo

Este informe presentará una lista de problemas identificados por Sonar Lint en el proyecto, acompañada de explicaciones sobre por qué es posible que no deban preocuparnos. Utilizaremos un lenguaje simple y directo para asegurarnos de que sea fácil de entender y para garantizar un producto final de excelente calidad.

2. Introducción

Voy a emplear el plugin Lint de Sonar para Eclipse para examinar todos los archivos que he generado como Estudiante 4 para mis tareas individuales.


Concretamente, vamos a evaluar el contenido de los paquetes Java "entities.sponsorships", "features.sponsor", "features.any.sponsorship" y "features.authenticated.sponsor".

Además, analizaremos la función del Patrocinador y el panel SponsorDashboard. También someteremos a análisis los archivos adicionales como Patrocinio, Factura y Patrocinador, junto con los archivos de visualización (forms.jsp, list.jsp, y archivos i18n) que están ubicados en las carpetas "views/sponsor", "views/any/sponsorship" y "views/authenticated/sponsor".

Dado que muchos problemas de la calidad del código son compartidos por varios archivos, voy a inspeccionarlos uno por uno en lugar de hacerlo por clases.

3. Contenido

·Override the "equals" method in this class:

	Diseño y Pruebas 2 Documentación de la entrega D03
	Documentación de SonarLint

Esta sugerencia surge porque es una práctica común sobrescribir el método "equals" en las clases de Java. Sin embargo, dado que no se utiliza en nuestra implementación y se ha comentado en la clase, no es necesaria ninguna corrección.

·**Remove the unnecessary Boolean literal:**

Este bad smell se debe a las validaciones que comprueban si una entidad está publicada: `"object.isPublished() == false"`. Es cierto que puedes escribir simplemente `"!object.isPublished()"`, pero como forma parte de un assert más grande, considero que es más legible con `== false`. En cualquier caso, esto solo ocurre un par de veces y creo que vale la pena una expresión un poco más larga a favor de la comprensibilidad.

·**Use concise character class syntax '\d' instead of '[0-9]':**

Esta recomendación se refiere al código - como atributos. Sin embargo, la implementación actual que utiliza `'[0-9]'` en la expresión regular es válida y, posiblemente, más comprensible para representar rangos de números latinos. Por lo tanto, no es necesaria ninguna corrección.

·**Rename this package name to match the regular expression `'^[a-z_]+(.[a-z_][a-z0-9_]')$'`:**


Aunque se sugiere renombrar el paquete, el nombre actual se ajusta al estándar del marco de trabajo y sigue lo que fue enseñado por los profesores. Por lo tanto, no se necesita ninguna corrección.

·**Define a constant instead of duplicating this literal:**

Esta recomendación sugiere definir una constante en lugar de duplicar un valor literal varias veces en el código. Sin embargo, en este caso, duplicar el literal puede no ser un problema significativo. La justificación es que el literal se usa solo en algunos lugares (2 o 3 veces como máximo).

·**Replace this assert with a proper check:**

Se marca el assert `'object != null;'`, pero es la implementación recomendada tanto por los profesores como por el marco de trabajo. Por lo tanto, no es necesaria ninguna modificación.

	Diseño y Pruebas 2 Documentación de la entrega D03
	Documentación de SonarLint

4. Conclusiones

En conclusión, después de analizar el código, encontramos algunas áreas que podrían mejorar. Sin embargo, al examinarlas más detenidamente, muchas de estas sugerencias no parecen ser tan problemáticas. Por ejemplo, algunas recomendaciones, como cambiar el método "equals" o modificar los nombres de los paquetes, no son realmente necesarias, ya que siguen las normas y prácticas que nos enseñaron en clase.

Además, algunas decisiones que tomamos, como usar comparaciones booleanas claras o repetir ciertas partes del código, en realidad ayudan a que el código sea más fácil de entender. Es importante destacar que estas decisiones solo se tomaron en unas pocas ocasiones.

En general, es importante tener en cuenta las sugerencias para mejorar el código, pero también debemos considerar el contexto del proyecto y las normas que seguimos. En este caso, parece que las prácticas de codificación que hemos utilizado están en línea con lo que se espera para nuestro proyecto.

5. Bibliografía

Intencionalmente en blanco.