PS17 – Informe de calidad: Historia de usuario "Visualizar logo en arranque"

Resumen

En el presente documento, se detalla el análisis de calidad del código implementado que da funcionalidad a la historia de usuario *Visualizar logo en arranque.*

Document ID:	PS17/00/2017-QR003-US242580-VisualizarLogoEnArranque
Departamento:	Calidad y Auditoría, dentro del proyecto integrado
Tipo:	Análisis de calidad
Privacidad:	CONFIDENCIAL
Estado:	ENTREGABLE
Versión:	1.0.2
Fecha:	15/11/2017
Autores:	Oslé García, Luis
Revisores:	Martínez, Javier; Cerezo, Elsa

HISTORIAL DE CAMBIOS

Versión	Fecha	Cambio	Responsable
1.0.0	15/11/2017	Creación del documento y redacción de los apartados.	Oslé García, Luis
1.0.1	15/11/2017	Revisada gramática y estructura.	Martínez Vila, Javier
1.0.2	15/11/2017	Revisar contenido del documento.	Cerezo Fernández, Elsa

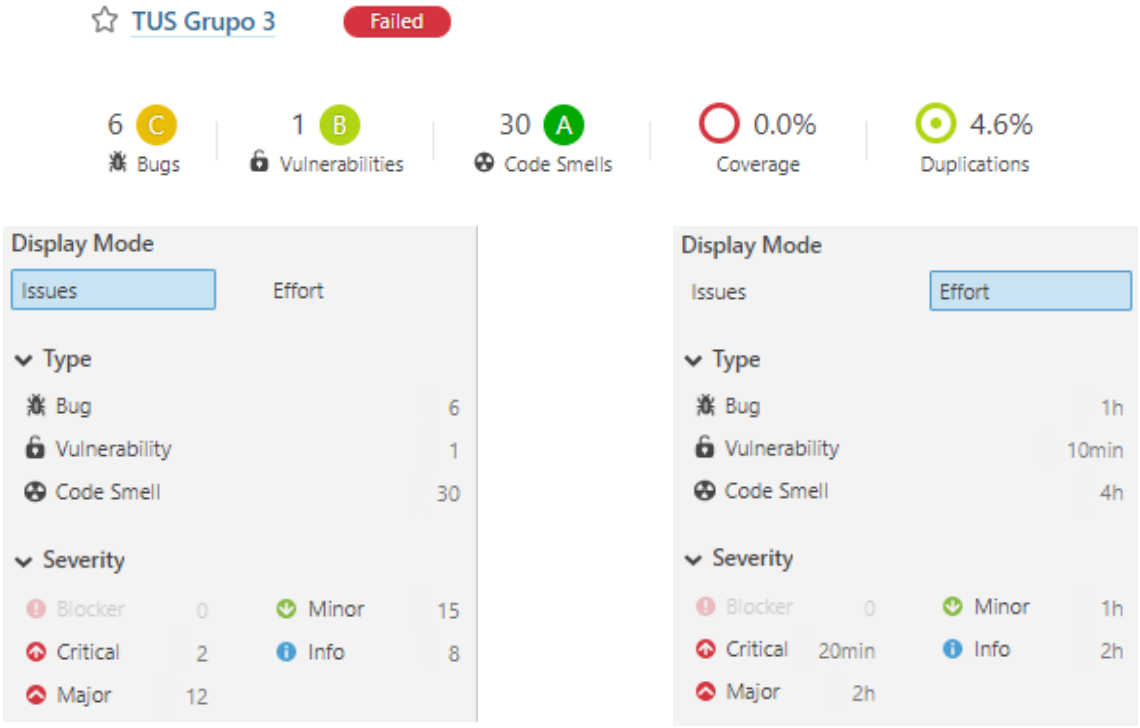
1. Introducción

El presente documento contiene el proceso de análisis de calidad llevado a cabo para la historia de usuario **#242580 – Visualizar logo en arranque**.

2. Análisis de calidad

2.1. Análisis inicial

Tras subir el proyecto al servidor *SonarCloud*, el análisis llevado a cabo por dicha herramienta arrojó los siguientes resultados:



Como puede observarse en las capturas expuestas previamente, el análisis arroja un resultado bastante negativo. El código recoge seis horas de deuda técnica repartida entre los siguientes fallos: seis bugs, una vulnerabilidad y treinta code smells. Analizando más en detalle los problemas encontrados, puede observarse que existen dos problemas críticos, doce de severidad mayor, quince de severidad menor y ocho informativos.

Las acciones correctivas y los errores tratados se enumerarán en el apartado siguiente, junto a las decisiones que se han tomado al respecto.

2.2. Acciones correctivas

Al afrontar la adopción de medidas correctivas, he optado por dar prioridad en primer lugar a los bugs, en segundo lugar, a la vulnerabilidad y, por último, a los code smells. Teniendo en cuenta ese orden, dentro del mismo he priorizado aquellos problemas de

severidad mayor frente a los de menor importancia. A continuación, se enumeran los problemas analizados y corregidos.

1. **Bugs y Vulnerabilities**

- En la clase *Línea*, existía un bug de severidad mayor en la línea treinta y siete: dicha instrucción puede generar un *NullPointerException*. Se ha tratado comprobando que el objeto afectado no fuese null (if… – else).
- En la clase *Parada*, existía un bug de severidad mayor en la línea setenta y uno: dicha instrucción puede generar un *NullPointerException*. Se ha tratado comprobando que el objeto afectado no fuese null (if… – else). Este error es similar al anterior, teniendo el mismo tratamiento.
- En la clase *Línea*, existía un bug de severidad menor en la línea treinta y cinco: si se sobrescribe el método equals, debe sobrescribirse el método *hashCode*. Se ha tratado sobrescribiendo el método *hashCode*.
- En la clase *Parada*, existía un bug de severidad menor en la línea sesenta y nueve: si se sobrescribe el método equals, debe sobrescribirse el método *hashCode*. Se ha tratado sobrescribiendo el método *hashCode*. Este error es similar al anterior, teniendo el mismo tratamiento.
- En la clase *Línea*, existía un bug de severidad menor en la línea treinta y cinco: era preciso comprobar que el objeto que se pasaba como parámetro era el adecuado para realizar la comprobación.
- En la clase *Parada*, existía un bug de severidad menor en la línea sesenta y nueve: era preciso comprobar que el objeto que se pasaba como parámetro era el adecuado para realizar la comprobación.
- En la clase *ParadasFragment*, línea ciento veintisiete, existía una vulnerabilidad consistente en que la variable paradas era "fácilmente accesible" por no ser estática o privada. Se ha tratado haciendo dicha variable privada en lugar de pública.

2. **Code Smells**

- En la clase *Database*, existía un code smell de severidad crítica consistente en que se duplicaba cuatro veces un String con la misma instrucción SQL (drop table if exists). Se ha solventado declarando una constante con el valor de dicho String.
- En la clase *LoadDataAsync*, existía un code smell de severidad crítica consistente en que se duplicaba cuatro veces otro *String* con el mismo valor (error). Se ha solventado declarando una constante con el valor de dicho String.
- En la clase *ParserJSON*, existía un code smell de severidad mayor consistente en que se duplicaban dos bloques de código idénticos para recorrer un array de paradas/líneas. Se ha solventado creando un método auxiliar para leer en genérico, que recibirá un parámetro indicándole el tipo de lectura a realizar.

- En la clase *ParserJSON*, existía un code smell de severidad mayor consistente en que se duplicaban dos bloques de código idénticos para leer una parada/línea. Se ha solventado creando un método auxiliar para leer en genérico, que recibirá un parámetro indicándole el tipo de lectura a realizar. Como la complejidad en nodos de decisión de dicho método sería muy elevada en caso de programarlo mediante bloques if-else, he optado por implementarlo mediante un switch.
- En las clases *ListLineasPresenter*, *ListParadasPresenter* y *LoadDataAsync*, existían code smells de severidad mayor consistentes en que aparecían algunas variables que no se utilizaban. Para solventarlo, las variables que no se utilizaban han sido eliminadas, mientras que las que sí se usaban tienen ahora observadores en lugar de ser reconvertidas en públicas.
- En la clase *LoadDataAsync*, han sido eliminadas líneas de código en las cuales existían comentarios que contenían código funcional (suponían un code smell de severidad mayor).
- En la clase *LineasFragment*, han sido eliminadas líneas de código en las cuales existían comentarios que contenían código funcional (suponían un code smell de severidad mayor).
- En la clase *ListLineasAdapter*, existe un code smell de severidad mayor que no ha sido corregido debido al esfuerzo temporal que supondría efectuar dicha corrección y las implicaciones que tendría en los test realizados.
- En las clases *Línea*, *ParserJSON*, *ListLineasPresenter*, *ListParadasPresenter*, *IListLineasView*, y *ListParadasAdapter* existían code smells de severidad menor consistentes en imports que ya no se utilizaban. Se han eliminado para solventar dichos errores.
- En la clase *LoadDataAsync* existía un code smell referente a una constante no estática que podría serlo. Para solventarlo, se ha declarado dicha constante como estática.
- Respecto a los code smells informativos, se han corregido algunos que suponían una inversión de tiempo razonable en comparación de otros que presentaban un coste mucho mayor y misma severidad, los cuales han sido obviados temporalmente.

2.3. Resultados

Tras efectuar las acciones correctivas citadas anteriormente, puede apreciarse que la deuda técnica se ha visto considerablemente reducida (ha pasado de seis a cuatro horas). Asimismo, se han reducido los problemas encontrados, de manera tal que actualmente tan solo existen once code smells, de los cuales diez son informativos y tan solo uno es de severidad mayor.

El único problema que persiste es la no cobertura del código nuevo, lo cual hace que el análisis de calidad no concluya con éxito (arroja un *failed*). Desconocemos el motivo por el cual dicho problema se mantiene a pesar de haber incluido las pruebas, por lo que se tratará posteriormente.

3. Sumario

En este documento se ha recogido el proceso seguido, los pasos analíticos y las acciones correctivas llevadas a cabo para optimizar la calidad del código asociado a la historia de usuario *"Visualizar logo en arranque"*.

Los resultados obtenidos tras la corrección presentan un balance positivo, habiéndose establecido la calificación máxima (A) en todos los elementos evaluables. El fallo al pasar el análisis se afrontará en posteriores sprints, tras consultar con el profesor responsable de la asignatura.