**Autor:** Pablo Martínez Arana
**Informe de Calidad:** VerEstimacionesParada

**Análisis del proyecto**

Tras realizar la subida del código y el análisis al servidor SonarCloud, observamos los resultados del análisis para comprobar si ha pasado o no el Quality Gate establecido en el servidor, en nuestro caso es "failed", debido a que no hemos conseguido una clasificación de "A" en dos de los tres apartados analizados: reliability(bugs), security(vulnerabilities) y maintainability (code smells).

En nuestro caso obtenemos:

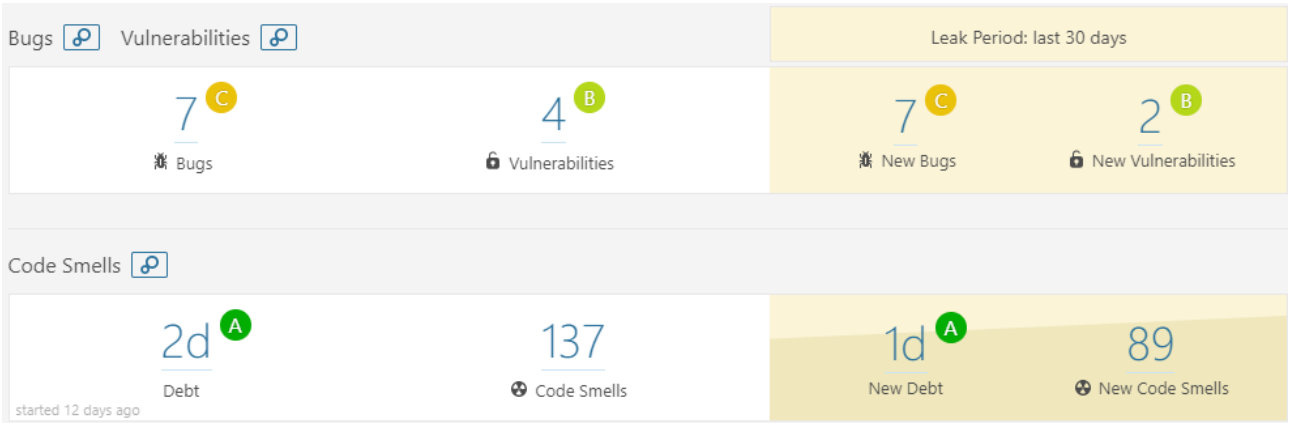


Figura 1. Análisis inicial

Como se puede observar los principales problemas de calidad los encontramos en los apartados de reliability(bugs) y security(vulnerabilities), siendos los resultados de estos, "C" y "B" respectivamente.

En cuanto a la deuda técnica, podemos observar que se ha visto incrementada debido al aumento de code smells, que podemos ver de forma detallada en la siguiente imagen:


Figura 2. Deuda técnica inicial

Para mejorar la calidad de nuestro código se ha decido inicialmente mejorar la clasificación obtenida tanto en reliability como en security, ya que no cumplían con los criterios de calidad exigidos.

Para mejorar la clasificación obtenida en reliability, se han solucionado problemas relativos a estas tres clases:

-Estimacion.java: casteo de variables, se ha pasado de utilizar variables de tipo int a variables de tipo double ya que al realizar divisiones se perdían decimales.
-ListLineasPresenter.java: tratamiento de la excepción NullPointerException, ahora se comprueba si el valor devuelto por la variable "db" es null o no y se informa de ello.
-ListParadasPresenter.java: tratamiento de la excepción NullPointerException, ahora se comprueba si el valor devuelto por la variable "db" es null o no y se informa de ello.

Para mejorar la clasificación obtenida en security, se han solucionado problemas relativos a estas tres clases:

-ListEstimacionesPresenter.java: depuración usando Log.e(), se ha modificado la forma de depurar mediante el cambio de la instrucción e.printStackTrace() por la instrucción Log.e().
-ListLineasPresenter.java: depuración usando Log.e(), se ha modificado la forma de depurar mediante el cambio de la instrucción e.printStackTrace() por la instrucción Log.e().
-ListParadasPresenter.java: depuración usando Log.e(), se ha modificado la forma de depurar mediante el cambio de la instrucción e.printStackTrace() por la instrucción Log.e().

Aunque la clasificación obtenida en  maintainability cumple con el requisito pedido, se ha conseguido mejorar aún más, esto es, se ha disminuido el número de code smells de 137 a 52, lo que se traduce en una disminución de una deuda técnica inicial de 2 días a 1 día.

Estas mejoras se pueden observar en las dos siguientes imágenes:

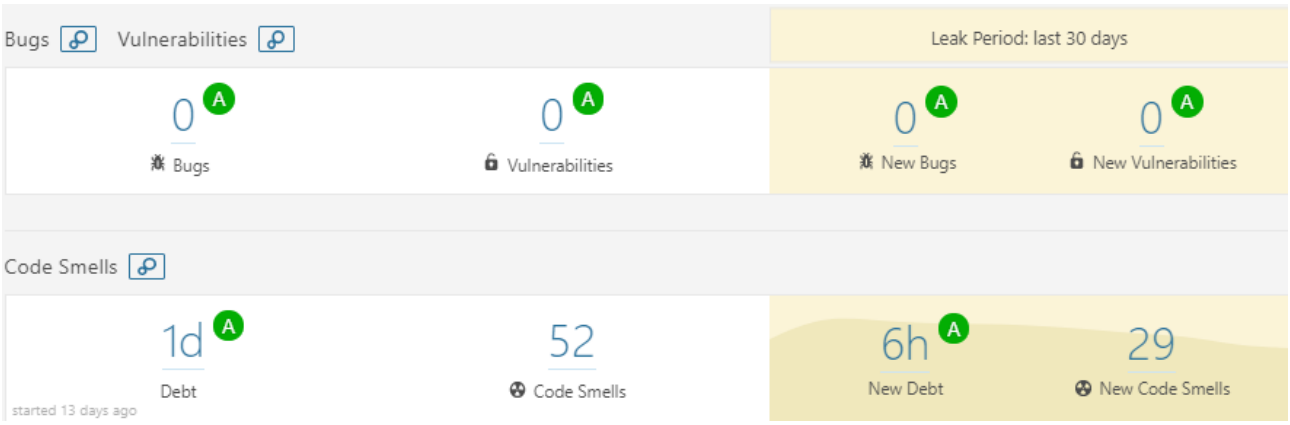


Figura 3. Análisis final


Figura 4. Deuda técnica final

Como se ha citado anteriorme, tras realizar los cambios necesarios, observamos que nuestro código cumple con los requisitos pedidos. Cabe destacar que, si bien el número de code smells se ha reducido considerablemente, éstos se podrían haber reducido aún, sin embargo se ha optado por solucionarlos en un próximo análisis puesto que son fallos menores.

Para finalizar, decir que, aunque el porcentaje de código duplicado se ha visto incrementado de un 0.0% a un 1.4%, no se ha considerado necesario realizar ninguna acción, si este porcentaje se ve incrementado notablemente se procederá a tomar medidas en un próximo análisis.