Autor: Adrián Lucas Degaudinne
Informe de Calidad: ImplementarBotonActualizar.

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 nos encontramos con una vulnerabilidad y bastantes code smells.

En nuestro caso obtenemos:

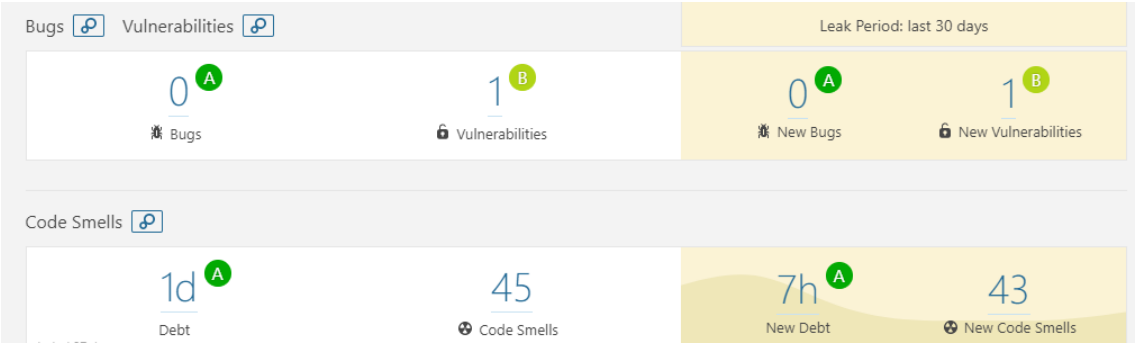

Figura 1. Análisis inicial.

Podemos apreciar que el principal problema de calidad le tenemos en las vulnerabilidades. Su resultado es “B”.

Figura 2. Deuda técnica inicial.

Si analizamos la deuda técnica del proyecto, podemos ver que las clases que más deuda técnica tienen son LineasFragment, ParadasFragment y ListLineasPresenter.

Aun así, la deuda técnica de todas las clases excepto ListLineasPresenter (quien tiene la vulnerabilidad), es debido a los code smells existentes en cada una de estas clases.

Para mejorar la calidad de nuestro código, en primer lugar, hemos decidido quitar la vulnerabilidad que nos provoca la “B” en el análisis en el apartado de seguridad del proyecto.

Para ello hemos hecho la siguiente modificación en la clase ListLineasPresenter:

Hacer privada la variable context.

Aunque con este cambio la calificación ya sea de “A” en todos los apartados, se ha decidido tratar de reducir un poco el número de code smells, para mejorar más si cabe la mantenibilidad del proyecto.

-ListLineasPresenter:

Se ha cambiado el mensaje “Error” que se mostraba varias veces en métodos de esta clase por una variable global ERROR que guarde este String.

Se ha cambiado el mensaje “DBTUS” que se mostraba varias veces en métodos de esta clase por una variable global DBTUS que guarde este String.

Se han borrado las líneas comentadas de código irrelevantes.

Se ha borrado el import innecesario java.util.Collection.

Se ha cambiado “” + position a la hora de pintar por pantalla por Integer.toString(position)

Se ha borrado el import no utilizado Java.io.InputStream

Se ha borrado el import no utilizado Android.view.Menu

-ParadasFragment

Se ha cambiado “” + position a la hora de pintar por pantalla por Integer.toString(position)

-ListParadasPresenter

Se ha cambiado el mensaje “Error” que se mostraba varias veces en métodos de esta clase por una variable global ERROR que guarde este String.

Se han borrado los imports innecesarios java.util.Collections y java.io.InputStream.

Se han borrado las líneas comentadas de código irrelevantes.

-MisFuncionesBBDD

Se ha cambiado el mensaje “idLinea” que se mostraba varias veces en métodos de esta clase por una variable global IDLINEA que guarde este String.

Se ha cambiado el mensaje “ParadaLinea” que se mostraba varias veces en métodos de esta clase por una variable global PARADALINEA que guarde este String.

-ParserJSON

Se ha eliminado el atributo no utilizado LINEAS_DE_LA_PARADA

-MainActivity

Se ha eliminado el import innecesario java.util.Log

Tras realizar la refactorización, volvemos a subir el proyecto a sonar para efectuar el análisis. Hemos conseguido quitar la vulnerabilidad que teníamos y así conseguir una “A” en ese apartado. También hemos reducido el número de code smells de 45 a 36, y hemos conseguido bajar la deuda técnica de un día a 7 horas.

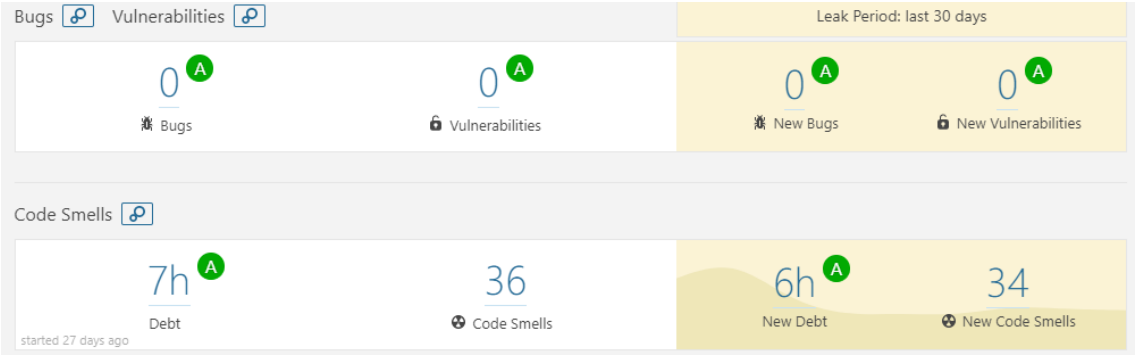

Figura 3. Análisis final.

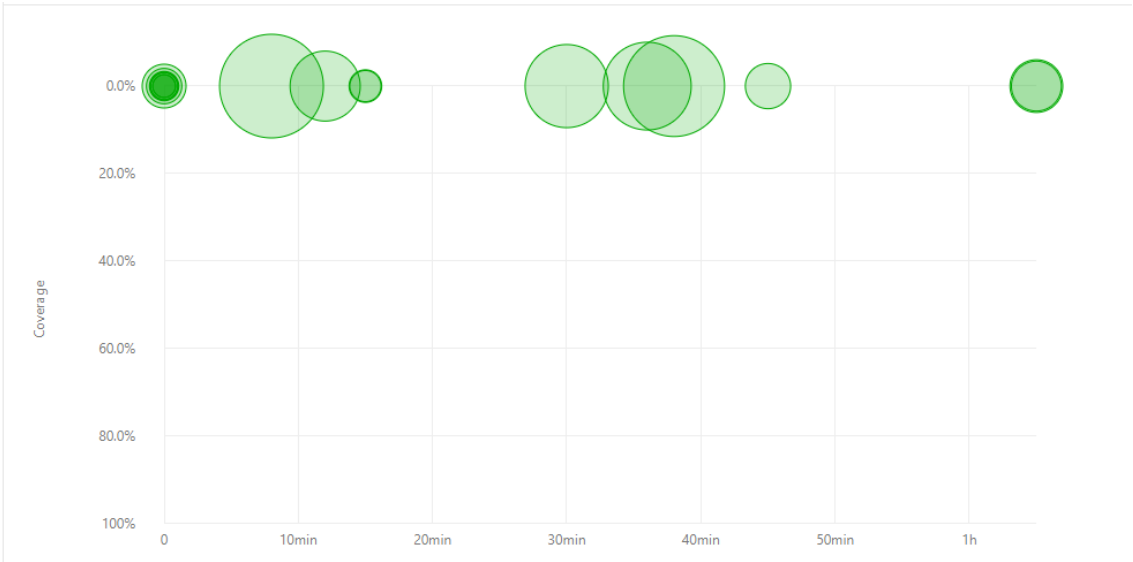

Figura 4. Deuda técnica final.

Tras realizar los cambios oportunos, podemos observar que nuestro código cumple con los requisitos acordados para pasar el análisis de calidad. Los code smells restantes se han decidido eliminar posteriormente, ya que son fallos menores y aún no se han investigado alternativas para refactorizar correctamente el código.

Por último, mencionar que el código duplicado es del 1’3%. En este aspecto no se ha considerado necesario realizar ninguna acción de refactorización, aunque si este porcentaje se viese incrementado, habría que tratarlo.