INFORME DE CALIDAD DE PRODUCTO

Autor	Manuel Álvarez Lavín
Número de Sprint	Sprint 01
Historia de Usuario	Filtrar Gasolineras por CCAA
Fecha	26/10/2016

1. Introducción

El objetivo de este documento será recoger el análisis de calidad realizado sobre la del código de la App “No Dry” desarrollada por el equipo con la finalidad de recoger datos acerca de la buena práctica de nuestro desarrollo y de la calidad del producto que se presentará ante el product owner.

Este documento se descompondrá en dos apartados, los cuales tratarán los resultados arrojados por el análisis, y posibles mejoras para solventar las evidencias que se detectan tras el análisis.

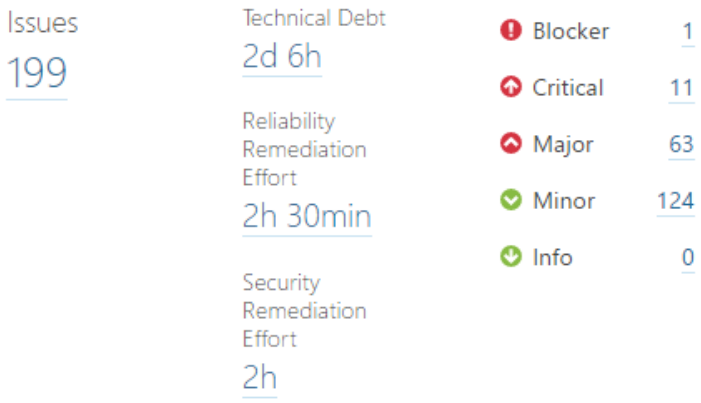
2. Resultados del análisis

En este apartado vamos a pasar a analizar los resultados arrojados por el análisis producido por la herramienta que nos proporciona sonar. Para este estudio vamos a diferenciar entre cuatro secciones, que serán: Evidencias, medida, estructura y evolución del proyecto.

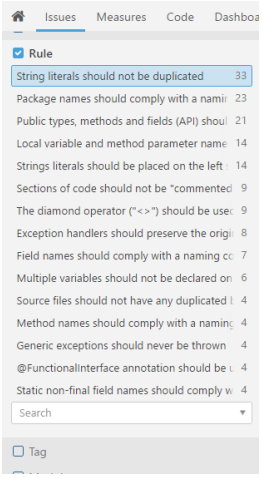
Evidencias

En este primer apartado podemos comprobar que se encuentran un total de 199 evidencias, las cuales nos dan lugar a una deuda técnica de 2 días y 6 horas.

Si miramos un poco más en profundidad encontramos distintos tipos de evidencias que sonar se encargará de clasificarlas según su severidad. En nuestro caso observamos que tenemos evidencias bloqueantes (1), críticas (11), mayor (63), menor (124), y de información (0).



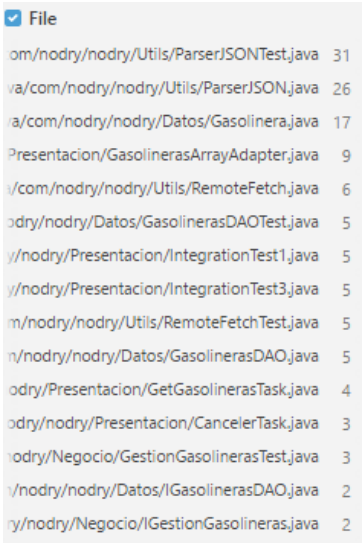
Además, si clicamos sobre cualquiera de estos tipos de evidencias sonar nos desglosará una por una todas las evidencias que hay en nuestro código de ese tipo.



Continuando con este análisis podemos comprobar si el código cumple determinadas reglas, para lo cual en el apartado Issues solo hemos de filtrar por determinada regla (“RULE”) y sonar volverá a seleccionarnos únicamente los posibles errores detectados en nuestro código. Como ejemplo en nuestro caso tomaremos “String literals should not be duplicated”

Un ejemplo de las vulnerabilidades mostradas por sonar al utilizar este método de filtrado es este:

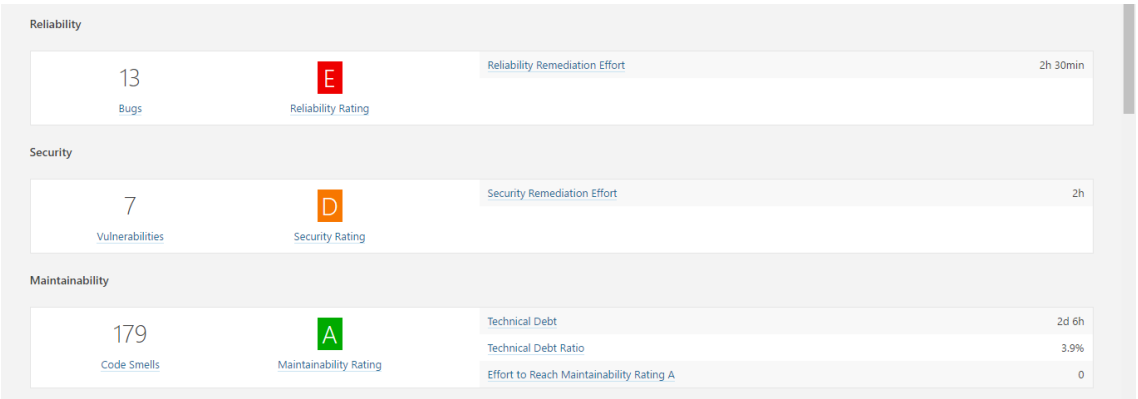
Otra posibilidad de filtrado es comprobar el número de vulnerabilidades por archivo, lo cual en este ejemplo nos dejará ver que el código con más vulnerabilidades en nuestro caso se encontrará en la clase “ParserJSONTest.java” con 31.



Medidas

Para este apartado Sonar dispone de una escala (SQALE) basada en las letras desde la A a la E, la cual nos indicará la calidad de nuestro código.

En este caso solo hemos de desplazarnos hasta la sección medidas y nos saldrá un resumen completo de todo nuestro proyecto, indicándonos la confiabilidad, el nivel de seguridad y la mantenibilidad.



Como se puede apreciar en la captura de pantalla tenemos distintas calificaciones para cada una de nuestras categorías, obteniendo una E en confiabilidad, lo cual nos indica que tenemos trabajo en este apartado.

Si profundizamos más vemos que calificación reciben cada una de nuestros ficheros, lo cual nos puede servir para identificar los cuales deberíamos mejorar. En este caso GasolinerasDAO.java es el único fichero con calificación E y tenemos otros 3 con calificación D, por lo que si centrásemos nuestros esfuerzos de mejora en estos ficheros la nota de este apartado mejoraría notablemente.

En primer lugar, la medida de confiabilidad referente a la calificación SQALE es de una E, la más baja dentro de la calificación SQALE. Esto puede provocar que el usuario quiera realizar una tarea, y el sistema falle. Al ver los diferentes archivos, la clase GasolineraDAO es la única que tiene una calificación de E, por lo que, si se mejorara esta clase, seguramente se obtendría una diferente calificación.

Además de lo ya comentado podemos observar como en materia de seguridad Sonar nos otorga una calificación D, la cual también ha de mejorarse, para lo cual tendremos que centrarnos en clases encargadas de parsear los datos obtenidos mediante JSON y también revisar los test. Hechos estos cambios nuestra calificación mejorará ostensiblemente.

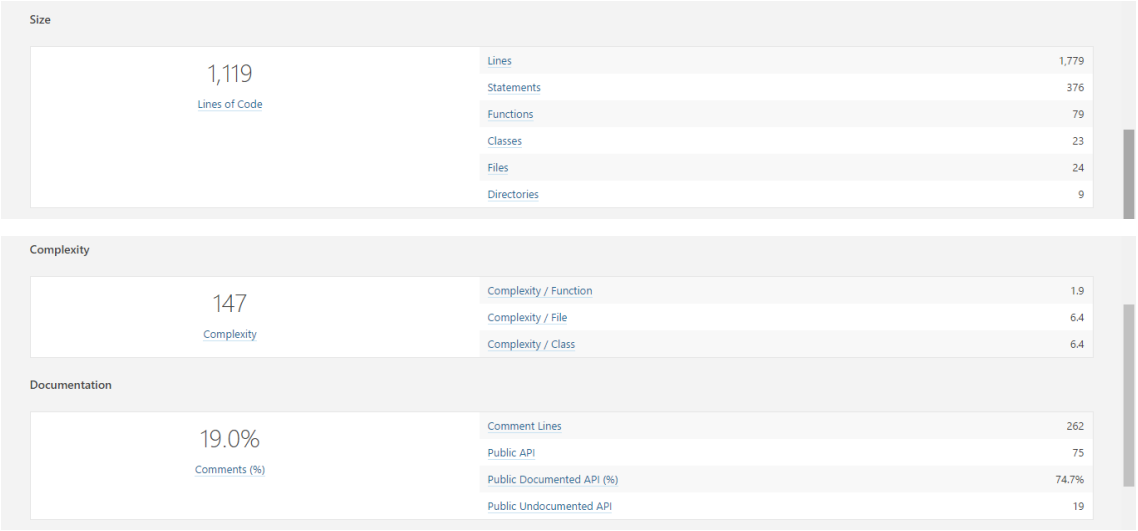
En lo referente a la mantenibilidad Sonar nos otorga la calificación de A, lo cual indica que el trabajo es satisfactorio, y a pesar de ser mejorable y tener más de dos horas de deuda técnica, podemos centrar nuestros esfuerzos en los otros

campos para mejorar la nota general de nuestro proyecto y así con menos esfuerzo conseguir mayor rendimiento.

Además de estas informaciones Sonar nos indica la cantidad de duplicaciones que tenemos en nuestro código, que en nuestro caso se elevará al 3,9%, lo cual es una cantidad muy aceptable, aunque siempre mejorable.

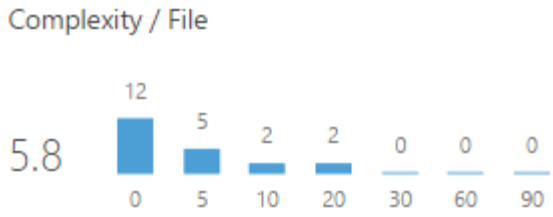
Estructura

Este apartado nos servirá para estudiar la información relativa a la estructura de nuestro proyecto, y nos proporcionará por ejemplo datos como el tamaño, la complejidad o la documentación.



En nuestro caso el proyecto consta de 1779 líneas, de las que 1119 serán de código, 79 de funciones, 23 de clases, etc.´

Además, si hablamos de la complejidad de nuestro proyecto Sonar nos indica que es de 147, los cuales podemos ver desgranados a su izquierda por función, fichero, o clase.



Tras ver estos gráficos, considero que los ficheros tienen una complejidad media alta pero dentro de un intervalo razonable.

En la imagen anterior, se ven los ficheros que presentan una mayor complejidad.

En lo relativo a la documentación, vemos que el código está comentado en un 19 %, es un porcentaje bastante pequeño y podría provocar pérdidas de tiempo para posteriores implementaciones, y se considerará un punto importante de mejora.

Evolución del proyecto

En este caso dado que el sondeo se ha realizado en el primer Sprint no hay una versión estable de prueba con la que comparar resultados previos. Si comparamos con la historia de usuario previa si podemos concluir que la calidad del proyecto se ha mantenido prácticamente inalterada, mejorando en algunos aspectos, como la documentación o la reducción en la duplicidad del código, pero también empeorando en el número de vulnerabilidades por poner un ejemplo, aunque como se ha indicado previamente, las oscilaciones en mejoras o deterioros son tan pequeños que no se notan en el resultado final.

3. Acciones correctivas

Vamos a tomar distintas medidas correctivas de cara a la implementación de la segunda versión de nuestro proyecto, las cuales vendrán a ser:

- Dado que antes hablamos de las graves deficiencias encontradas en confiabilidad y seguridad y las bajas notas obtenidas (E y D respectivamente), tendremos que detenernos en la mejora de las clases antes propuestas, como podían ser GasolinerasDAO, los tests, o el parseo de JSON. Otra de las posibles mejoras se centrará en la seguridad, la cual también se centra en las clases Test, y necesitarán mayor cantidad de horas de trabajo, pero nos mejorarán sustancialmente la nota de Sonar.

- Para mejorar el código y conseguir así que fuese más claro, deberíamos aumentar el porcentaje de documentación ya que el valor 19% es un valor extremadamente pequeño.

- Por último, debemos de solucionar de manera apremiante la evidencia bloqueante que tenemos, y acto seguido las .11 críticas para centrar nuestros esfuerzos en las acciones que son más severas y nos proporcionarán una mejora más sustancial.