Documento entrega:

Sonar Qube: Es una plataforma abierta, que facilita la inspección continua del código para detectar problemas, vulnerabilidades y *code smells*. Es posible usarla en 20 lenguajes diferentes. En nuestro caso la usaremos para nuestra aplicación escrita en Java.

SonarQube corre en el puerto 9000. Para lograr el funcionamiento correcto de la aplicación, se debe primero configurar el servidor sobre el cual correrá la aplicación y los parámetros del ejecutable. Que son el nombre de la aplicación, la versión, la fuente, entre otras cosas.

Después de lo anterior, se despliega un *dashboard* informativo con diferentes métricas, las cuales se explican a continuación.

En la figura 1, observamos un *overview*  de lo que es la aplicación. Nos muestra la deuda técnica, así como la tasa de la misma y los problemas sobre los cuales se calcula tal deuda. Además divide los problemas en bloqueantes, críticos, importantes, no importantes e ínfimos.

El cálculo de la deuda técnica es basa en la metodología SQALE (*Software Quality Assessment based on LifeCycle Expectations).* Dicho cálculo tiene en cuenta bloques duplicados, pruebas unitarias fallidas, insuficiente cubrimiento de las pruebas unitarias, insuficiente densidad de comentarios del código, insuficiente cubrimiento de líneas por parte de las pruebas unitarias y pruebas no ejecutadas.

Teniendo en cuenta la deuda técnica de cada problema encontrado se realiza una sumatoria, teniendo así el valor final que es el mostrado en la pantalla. También es posible visualizar la deuda total por criterio en la interfaz de problemas.

*Technical Debt Ratio:* esta métrica nos da la taza entre la deuda técnica actual y el esfuerzo que tomaría reescribir el código por completo desde cero. Adicionalmente el valor de esta métrica depende del tamaño del proyecto. Dado un valor para esta tasa, el proyecto recibe una calificación que va desde A hasta la E, siendo la ultima la peor calificación.

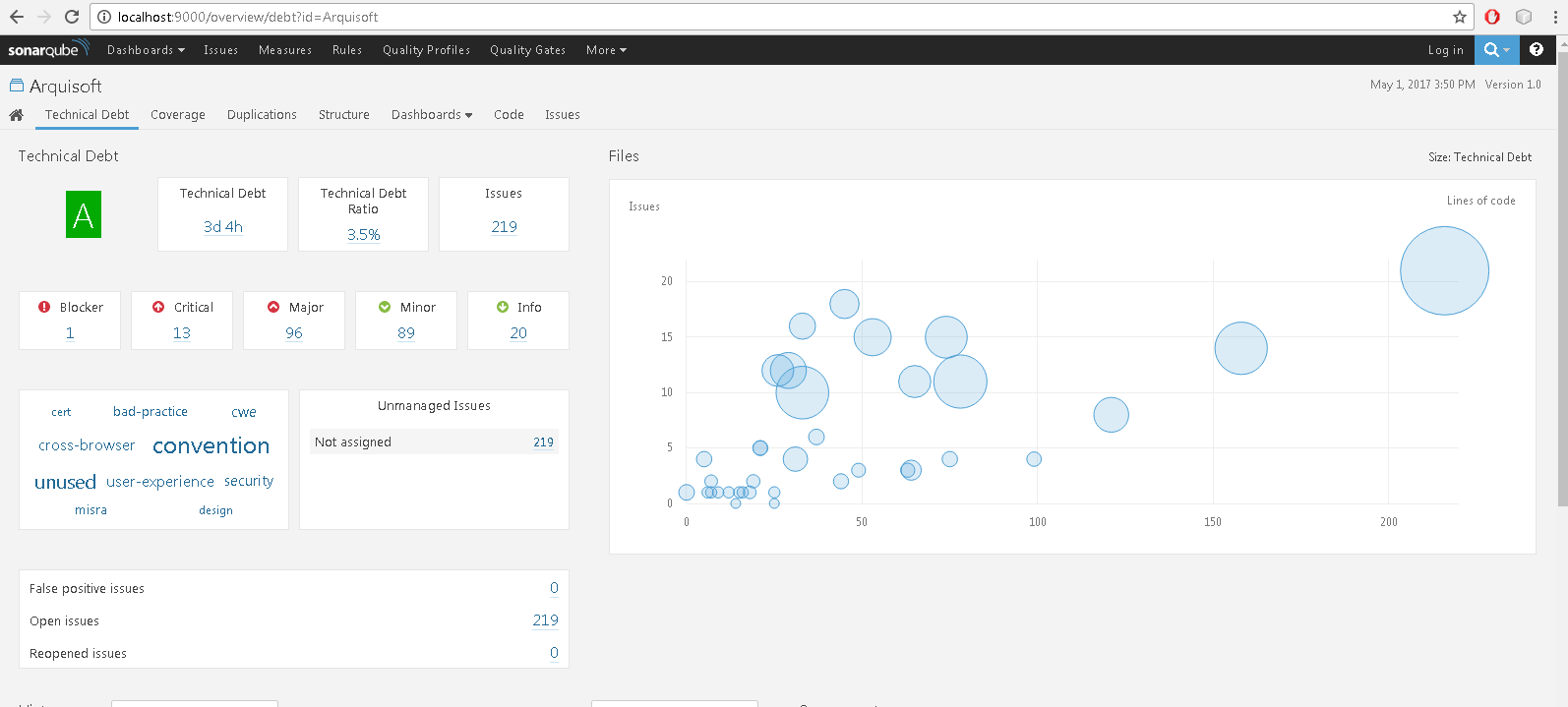
Compararemos dos proyectos con la misma deuda técnica:

* Arquisoft 1: 95 days y 80.000 lineas de codigo.
* Arquisoft 2: 95 days y 7.000 lineas de codigo.

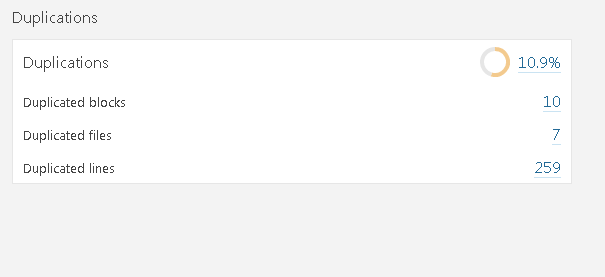
La fórmula de la tasa de deuda técnica es: Deuda técnica / tiempo estimado de desarrollo , que se calcula en LOC/30min.

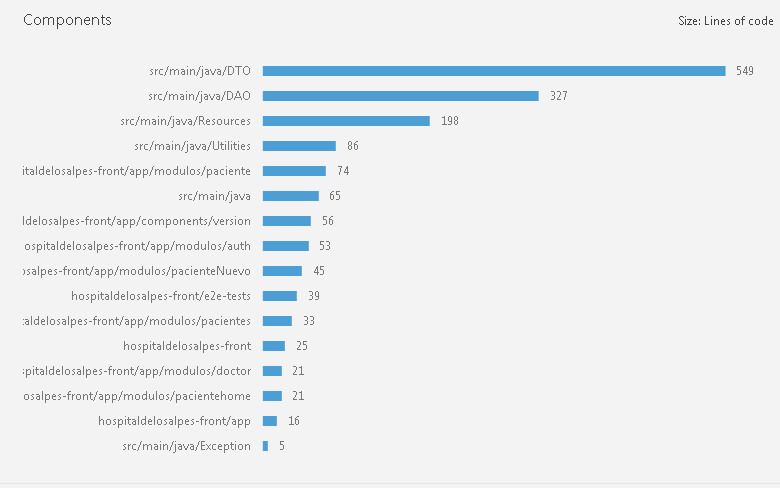
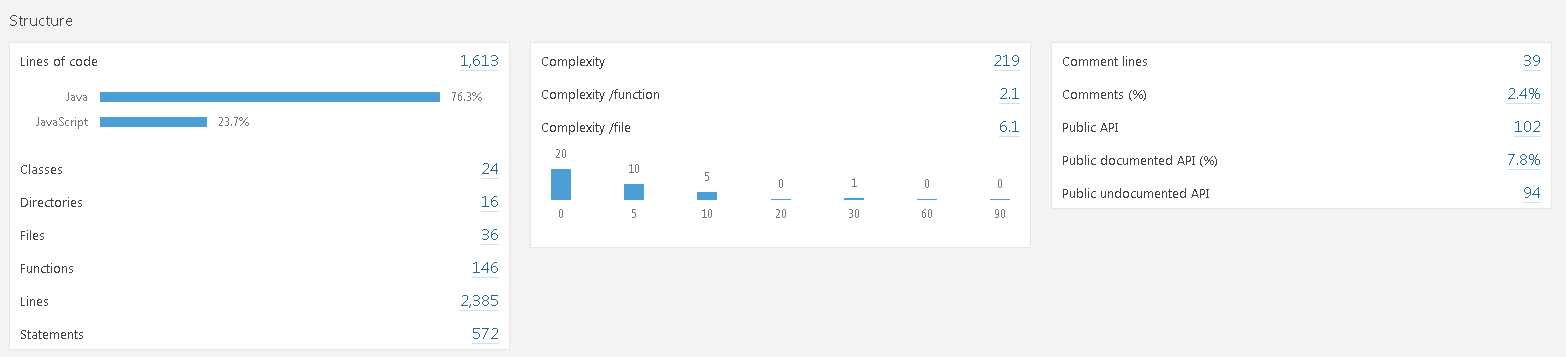
* Arquisoft 1: (95 x 8 x 60) / (80 000 x 30) = **1,9%**
* Arquisoft 2: (95 x 8 x 60) / (13 000 x 30) = **21,7%**

De lo anterior sabemos que el proyecto 1 está en mejor forma que el proyecto 2.



En cuanto a las duplicaciones, encontramos diferentes tipos, la primera en cuanto a los bloques repetidos, debe haber al menos 10 declaraciones sucesivas y duplicadas, independientemente del número de fichas y líneas, esto lo toma como bloques repetidos, líneas duplicadas, archivos duplicados y el porcentaje de líneas duplicadas.



La complejidad es calculada con base en el número de caminos existentes en el código. En donde sea que el flujo de una función se divida, la complejidad se aumenta en uno. Cada función tiene una complejidad mínima de 1. Este cálculo varia depende el lenguaje, ya que algunas palabras claves y funcionalidades también lo hacen. 

Ahora bien, para el segundo análisis, se implementó un wrapper (encapsulamiento) y una fachada, con el fin de probar una nueva distribución para saber cuál de las dos si la nueva o a anterior es mejor.



Uso de una interfaz como wrapper

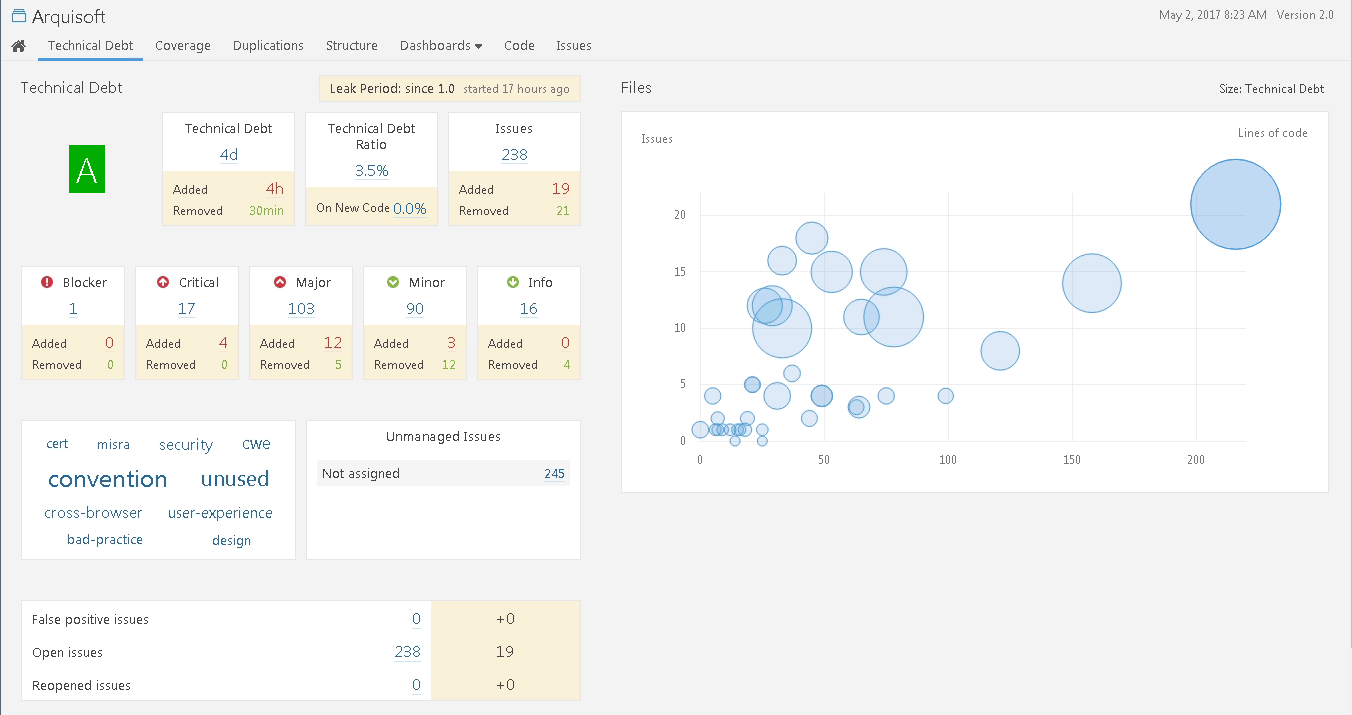


Uso de clases de recursos como mediadores



Se tiene un UML general en donde se tiene un mundo llamado HospitalLosAlpes que actúa como fachada de todo

En cuanto a la deuda técnica se encuentra un incremento, pero la tasa de la misma se mantiene por lo que este criterio no cambia. Los *issues*, asi como se reducen algunos se incrementan otros cuantos, pero es por comentarios, o elementos similares:



La tasa de duplicaciones aumenta debido a las nuevas clases introducidas y a que las líneas de las nuevas clases son similares en cuanto a estructura.

