

INFORME DE CALIDAD DE PRODUCTO

Autor	Ignacio Agüero Salcines
Número de Sprint	Sprint 003
Historia de Usuario	Filtrar Gasolinera
Fecha	21/11/2016

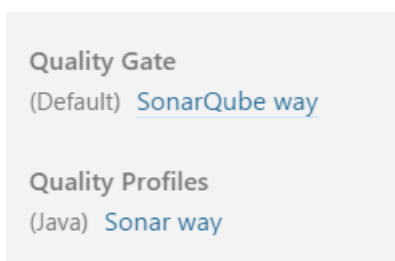
1. Introducción

El objetivo de la actividad es realizar un análisis de calidad interna de producto software mediante la herramienta SonarQube.

El análisis se realizará sobre el proyecto “AppGasolineras”, concretamente se analiza la historia de usuario “Filtrar gasolineras”, en la cual se realizarán medidas y/o correcciones para mejorar u optimizar la calidad del proyecto.

2. Análisis

Quality Gate y Quality Profile



En este proyecto se está utilizando una métrica de calidad (Quality Gate) por defecto que proporciona la propia plataforma de SonarQube.

Metric	Over Leak Period	Operator	Warning	Error
Coverage on New Code	Always	is less than		80.0%
New Bugs	Always	is greater than		0
New Vulnerabilities	Always	is greater than		0
Technical Debt Ratio on New Code	Always	is greater than		5.0%

Uno de los parámetros de calidad del software implementado es la calificación SQALE, En el análisis inicial sin refactorización la calificación SQALE es D. Por lo tanto, se mejorará el código para alcanzar una calificación mínima de A, atendiendo primordialmente a aquellos elementos que impiden una mayor calificación.

Bugs & Vulnerabilities		Leak Period: since 2.1 started hace 9 horas	
14 ^D	6 ^D	11	4
Bugs	Vulnerabilities	New Bugs	New Vulnerabilities

Además, utiliza un perfil orientado a la programación sobre el lenguaje Java, el cual establece reglas propias del lenguaje y define métricas sobre las declaraciones o implementación en si del proyecto.

Sonar way		
Rules	Active	Inactive
Total	268	111
Bugs	89	13
Vulnerabilities	18	13
Code Smells	161	85
Activate More		

Rules

☒ **Language**

Java	268
------	-----

☒ **Type**

Bug	89
Vulnerability	18
Code Smell	161

☐ Tag

☐ Repository

☐ Default Severity

☐ Status

☐ Available Since

☐ Template

☒ **Quality Profile**

Sonar way Java

active

inactive

☐ Inheritance

Evidencias

Observando las evidencias que presenta Sonarqube, podemos observar que existen un total de 353 issues o problemas en el código (referente a la calidad) que deben de ser tenidos en cuenta.

3

Ignacio Agüero

Type			
Bug	16		
Vulnerability	6		
Code Smell	331		
Resolution			
Unresolved	321	Fixed	32
False Positive	0	Won't fix	0
Removed	0		
Severity			
Blocker	0	Minor	202
Critical	16	Info	0
Major	135		

De todos los issues presentados, se resolverán en orden preferente los Críticos, Mayores y menores, de forma que se optimice la calidad del código.

Medidas

En el proyecto se pueden comprobar las medidas de reliability referentes a su calificación SQALE, los bugs detectados y la deuda técnica asociada:

14	11	D	Reliability Remediation Effort	1h 27min
Bugs	New Bugs	Reliability Rating	Reliability Remediation Effort on New Code	1h 2min

Se observa así, que existen 11 nuevos bugs a los 14 ya asociados a la anterior versión del proyecto y que el Reliability Rating tiene una calificación D, la cual deberá ser mejorada. Además, se estima que se tardaran en resolver dichos bugs nuevos alrededor de 1 hora.

Comprobando las medidas de security, calificación SQALE, vulnerabilidades y deuda asociada, se obtienen las siguientes métricas:

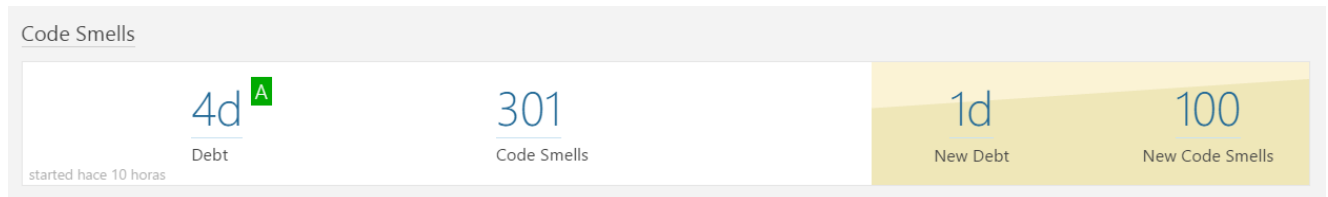
6	4	D	Security Remediation Effort	1h 40min
Vulnerabilities	New Vulnerabilities	Security Rating	Security Remediation Effort on New Code	1h

En base a estas métricas, se puede interpretar que existen 4 nuevas vulnerabilidades con un Security Rating de D, el cual deberá de ser mejorado. Para resolver estas nuevas vulnerabilidades, se estima que tiene una deuda de 1 hora.

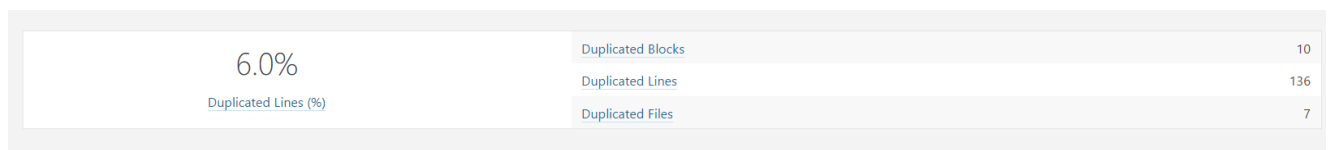
Si comprobamos las medidas de mantenibilidad, calificación SQALE, code smells, deuda técnica, ratio y el esfuerzo necesario para subir de nivel, nos percatamos de las siguientes métricas:



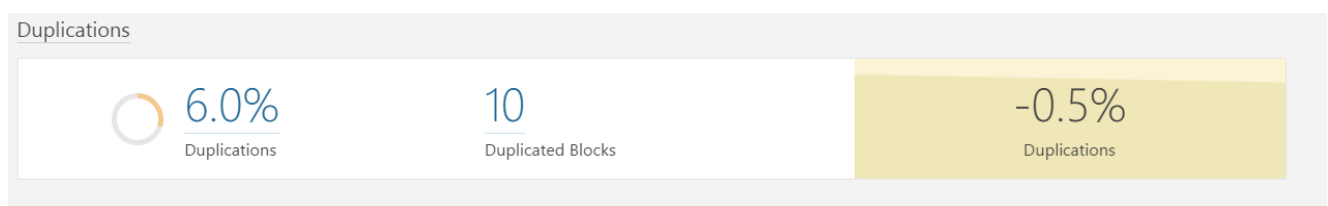
En este caso se han creado 100 nuevos Code Smells, sin embargo, la calificación Maintainability es de A, por lo tanto, en un principio no se tratará de mejorar dichos problemas ya que la deuda que implica es muy alta (4 días y 5 horas).



Obteniendo la información de duplicados (porcentaje, líneas, bloques y archivos):



Se puede comprobar que existe un 6% de código duplicado en el proyecto, el cual se tendrá en cuenta para mejorar la calificación global de la calidad del proyecto.

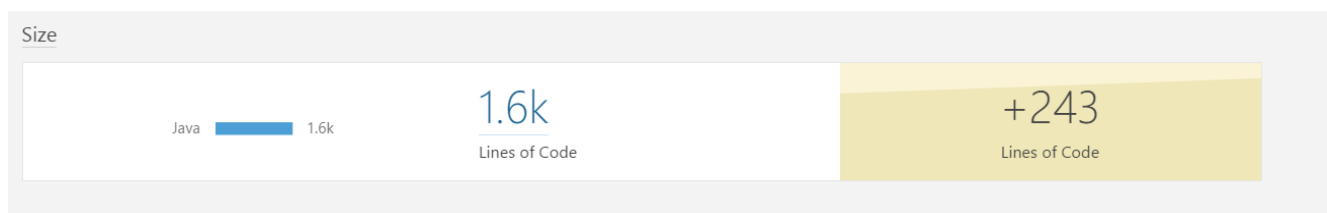


Comparando con la versión anterior, se puede ver que se ha mejorado esta métrica ya que se ha reducido la cantidad de duplicaciones.

Estructura

Tamaño

Se exponen a continuación las medidas referentes al tamaño:



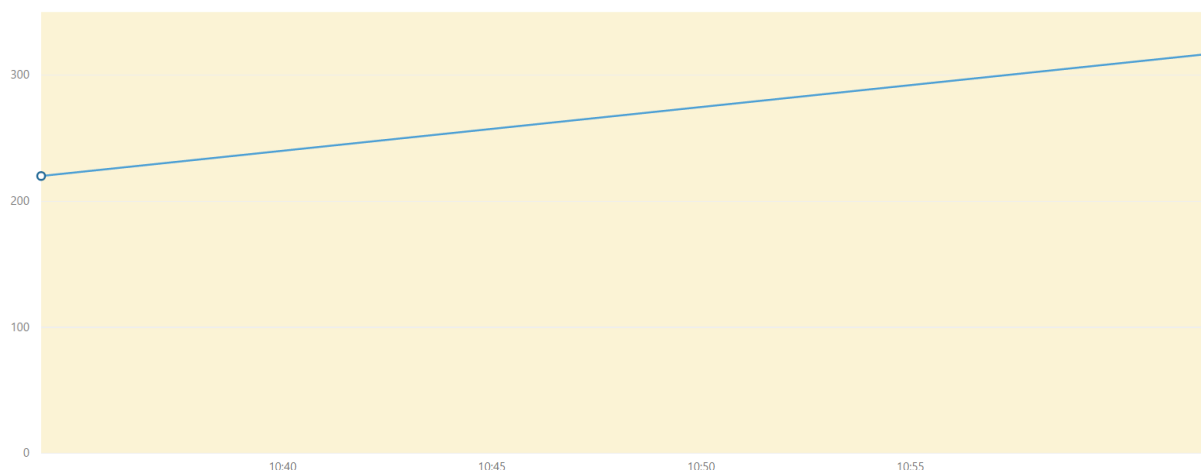
Gracias a estas métricas se puede comprobar que el volumen de código originado en este caso es bastante copioso, ya que se han añadido un total de 243 líneas nuevas de código.

Complejidad

Valores de complejidad total, por fichero, clase y función:

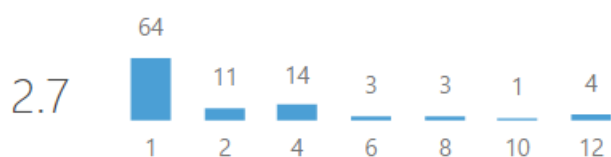


Comparando la complejidad con la anterior versión, se observa que se ha aumentado la complejidad de la misma. En este caso se han aumentado de 220 a de 317:



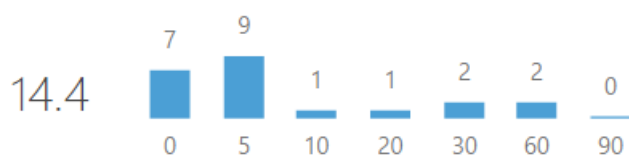
Analizando individualmente, se observa lo siguiente:

Complexity / Function



La complejidad de las funciones ha aumentado de 2.3 a 2.7. Además, se ve que la mayor parte de ellas tienen complejidad 1, seguidas de complejidad 4 y 2.

Complexity / File



Respecto a la complejidad de los ficheros, se puede observar que la mayor parte de los ficheros tienen complejidad 9, seguidos de complejidad 0, por lo tanto, no es una mala estadística.

La complejidad ha aumentado de 10 a 14,4 respecto a la anterior versión. Además, se puede ver que la mayor parte de complejidad reside en las clases `ListaGasolinerasActivity.java`, `GestionGasolinera.java` y `Gasolinera.java`.

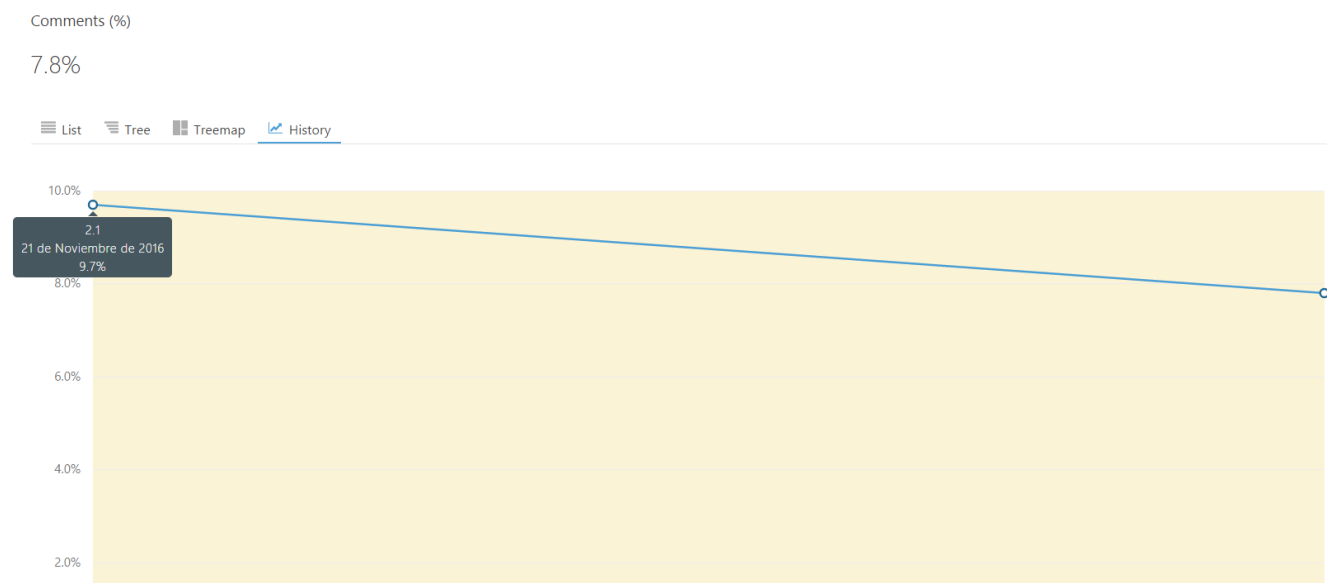
Documentación

El porcentaje de documentación y de comentarios del código del proyecto es el siguiente:

Documentation	
7.8%	Comment Lines 139
Comments (%)	Public API 125
	Public Documented API (%) 15.2%
	Public Undocumented API 106

Analizando cada apartado se observa lo siguiente:

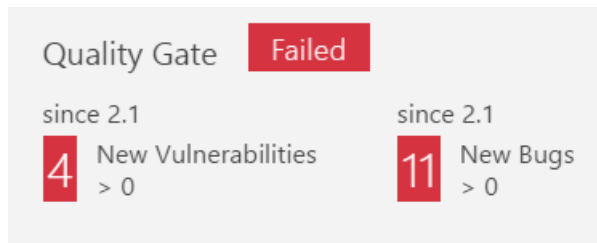
El código está comentado únicamente al 7.8%, por lo tanto, es un dato a tener en cuenta que mejorar para ayudar a la comprensión del código y respetar las métricas de calidad.



De este porcentaje comentado o documentado, se comprueba que se han reducido los comentarios en promedio sobre el código escrito de un 9,7 a un 7,8.

3. Resultados del análisis

Para mostrar los resultados del análisis, se tomará el Quality Gate y Quality profile definidos y mostrados anteriormente.



A continuación, se mostrará el conjunto de medidas que se deberían de tener en cuenta para mejorar la calidad del producto y conseguir una mejor calificación SQALE.

En esta ocasión, se tratarán únicamente aquellos issues que tengan una severidad bloqueante, crítica o mayor:

Bugs

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Datos/Gasolinera.java

Override "equals(Object obj)" to comply with the contract of the "compareTo(T o)" method. ...

Bug 🔴 Critical 🔵 Open Not assigned 15min effort

Este Bug es producido debido a que el compareTo de la clase Gasolinera tiene implementada una comparación mediante un switch que realiza diferentes acciones.

La solución reside en realizar la comparación mediante la implementación de un método equals que realice la funcionalidad del compareTo en función del tipo de gasolina.

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Negocio/GestionGasolinera.java

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Presentacion/ListaGasolinerasActivity.java

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Equality tests should not be made with floating point values. ...

Bug Critical Open Not assigned 5min effort

Este grupo de bugs (Equality test should not be made with floating point values) son producidos al comparar un float con un double. La causa es el número de decimales que tiene un tipo y el otro.

La solución reside en realizar la comparación con la librería matemática, la cual está preparada para hacer ese tipo de comparaciones.

Identical sub-expressions on both sides of operator "&&" ...



Bug Critical Open Not assigned 2min effort

Este tipo de bug (Identical sub-expressions on both sides of operator "&&") es debido a que hay comparaciones seguidas de otras, de modo que, si una comparación que va antes que otra se cumple, la siguiente no tendrá la posibilidad de satisfacerse o no.



La solución es hacer las comparaciones de forma no anidada de forma que se pase por todas ellas.

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Utilities/FilesOperations.java

Either log or rethrow this exception. ...

Bug  Major  Open Not assigned 10min effort

Either log or rethrow this exception. ...

Bug  Major  Open Not assigned 10min effort



El último grupo de bugs reclama un log a la hora de retornar una excepción.

La solución sería crear un log de errores e ir actualizándolo con cada excepción.



Vulnerabilities

App Gasolineras app/src/androidTest/java/com/example/nachoaguero/appgasolineras/TestLecturaJson.java

Define and throw a dedicated exception instead of using a generic one. ...



Vulnerability  Critical  Open Not assigned 20min effort

Define and throw a dedicated exception instead of using a generic one. ...

Vulnerability  Critical  Open Not assigned 20min effort

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Datos/GasolineraDAO.java

Define and throw a dedicated exception instead of using a generic one. ...



Vulnerability  Critical  Open Not assigned 20min effort

Este grupo de vulnerabilidades tratan de que las excepciones no deberían de ser genéricas sino personalizadas o centradas en el problema concreto.



La solución sería utilizar excepciones dedicadas al problema.

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Utilities/FilesOperations.java

Use a logger to log this exception. ...

Vulnerability  Critical  Open Not assigned 10min effort



Use a logger to log this exception. ...

Vulnerability  Critical  Open Not assigned 10min effort

Al igual que un bug, esta vulnerabilidad requiere de un log a la hora de lanzar una excepción.

App Gasolineras app/src/main/java/com/example/nachoaguero/appgasolineras/Utilities/ParserJSON.java

Make this "public static estadoLectura" field final ...

Vulnerability  Critical  Open Not assigned 20min effort

Esta vulnerabilidad pide que el atributo estático estadoLectura sea también final.

Evolución del proyecto:

Observando los aspectos que han cambiado comparando a la anterior versión:

- Se han creado 4 nuevas vulnerabilidades.
- Hay 11 nuevos bugs.
- Existe más Code Smell.
- Se han reducido los bloques duplicados en un 5%.
- Se han generado 243 nuevas líneas de código.

En líneas generales se ha reducido la calificación SQALE de seguridad y confiabilidad y además el Quality Gate no se ha superado.

4. Plan de mejora

Con el objetivo de mejorar la calidad del proyecto y pasar la Quality Gate y conseguir una calificación SQALE superior, se realizarán las siguientes medidas:

- Corregir los bugs y vulnerabilidades críticos.
- Mantener el estado de la deuda técnica en calificación A.
- Reducir la complejidad de las clases Gasolinera, GestionGasolinera y ListaGasolinerActivity.
- Mejorar el porcentaje de código documentado, ya que aportara entendimiento al código.
- Reducir la cantidad de código duplicado localizado principalmente en la clase GestionGasolinera.

5. Conclusión

Este documento ha hecho un análisis de la calidad del proyecto analizándolo mediante la herramienta SONARQUBE. Ha descrito los principales problemas o las carencias más necesarias para mejorar la calidad del software, y ha descrito una plan de mejora a utilizar como guía de obtención de calidad.