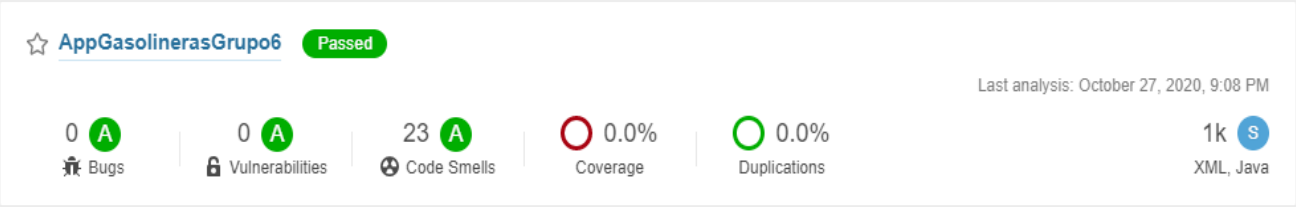
Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

ANÁLISIS 28 OCTUBRE 2020

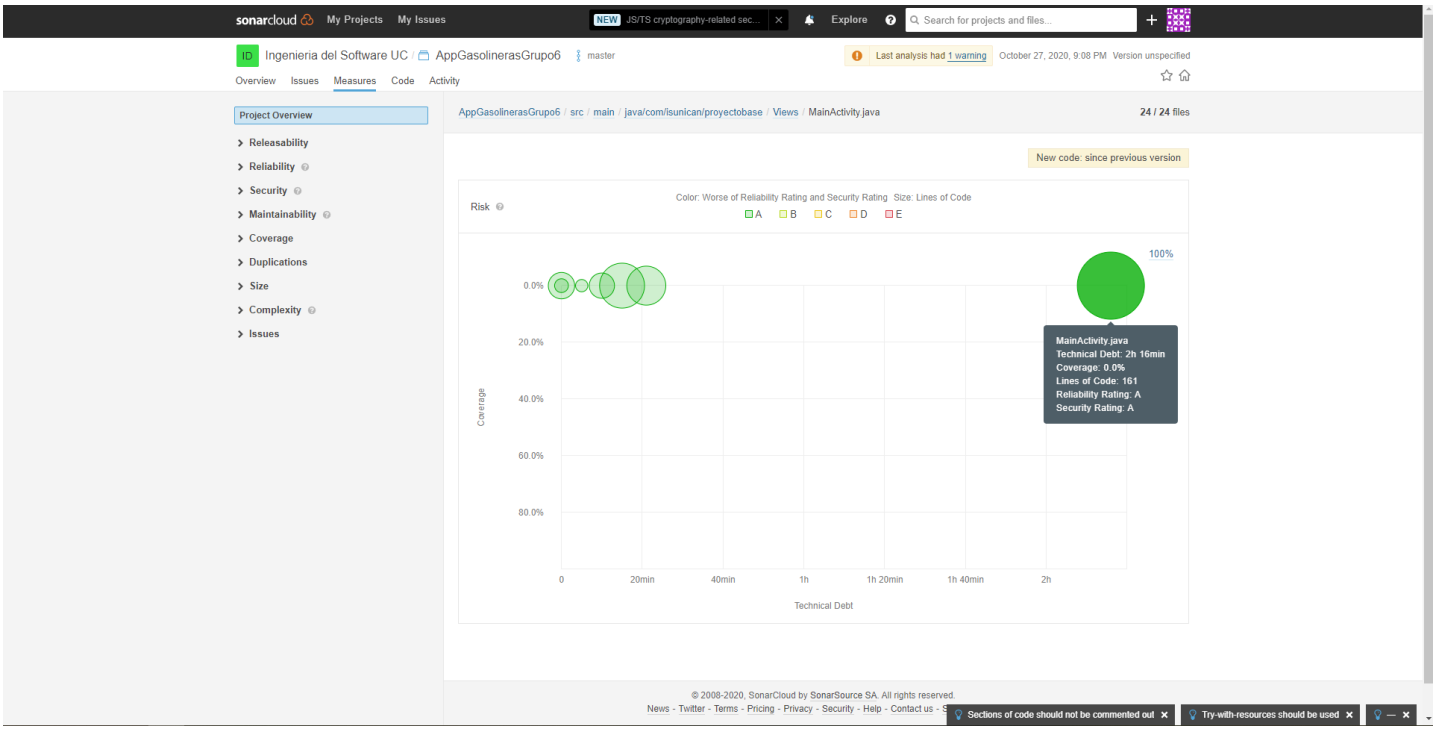
CAPTURA



INCIDENCIAS

El análisis pasa todos los criterios de calidad, pero nos encontramos con una deuda técnica de 3h por un total de 23 issues de mantenibilidad (code smells).

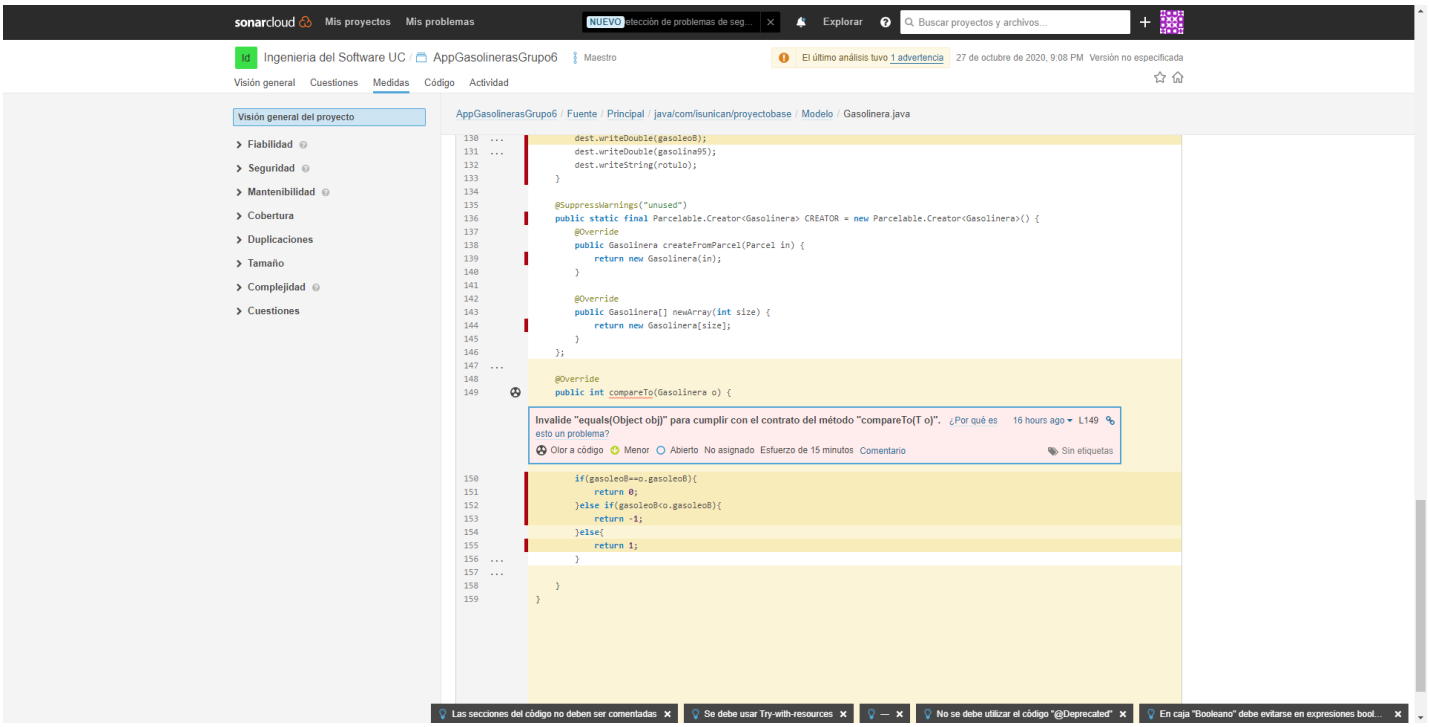
Está la mayor parte de la deuda técnica en la Activity MainActivity con un total de 2h 16min de la deuda técnica total:



Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

En la clase Gasolinera hay un único issue, que es un code smell, que genera una deuda técnica de 15 min:



PLAN DE ACCIÓN

1) Eliminar todas las issues de la Actividad MainActivity que no estén relacionadas con un uso de código que es deprecated, que se quedará obsoleto. Entre ellas corregir las issues relacionadas con casteo innecesario, código comentado, acotar bloques de código innecesariamente con llaves, eliminar la importación de librerías innecesarias, entre otros.
2) Dentro de la clase Gasolineras solo hay una issue, es un code smell que genera un deuda técnica de 15 minutos, si lo eliminamos tendremos la clase sin ninguna issue. Se trata de una issue que indica que junto al método compareTo(), que había implementado, había que implementar también el método equals() o "invalidarlo". Se optará por implementar dicho método.

Comentarios:

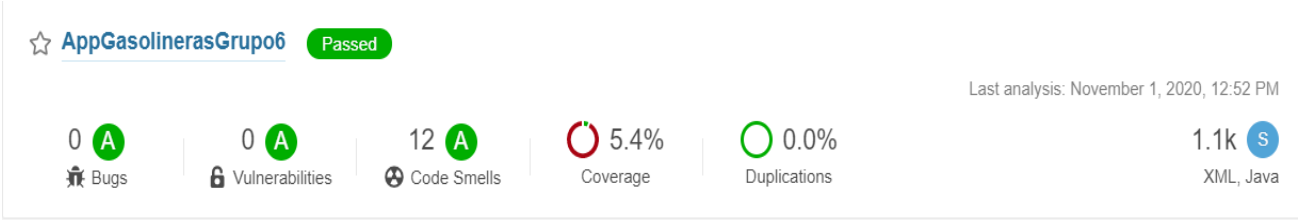
- Sin aplicar ninguna medida pasamos los criterios de calidad, pero el objetivo es mantener el código con la mayor calidad posible, por lo que seguiremos revisando en busca de issues.

Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

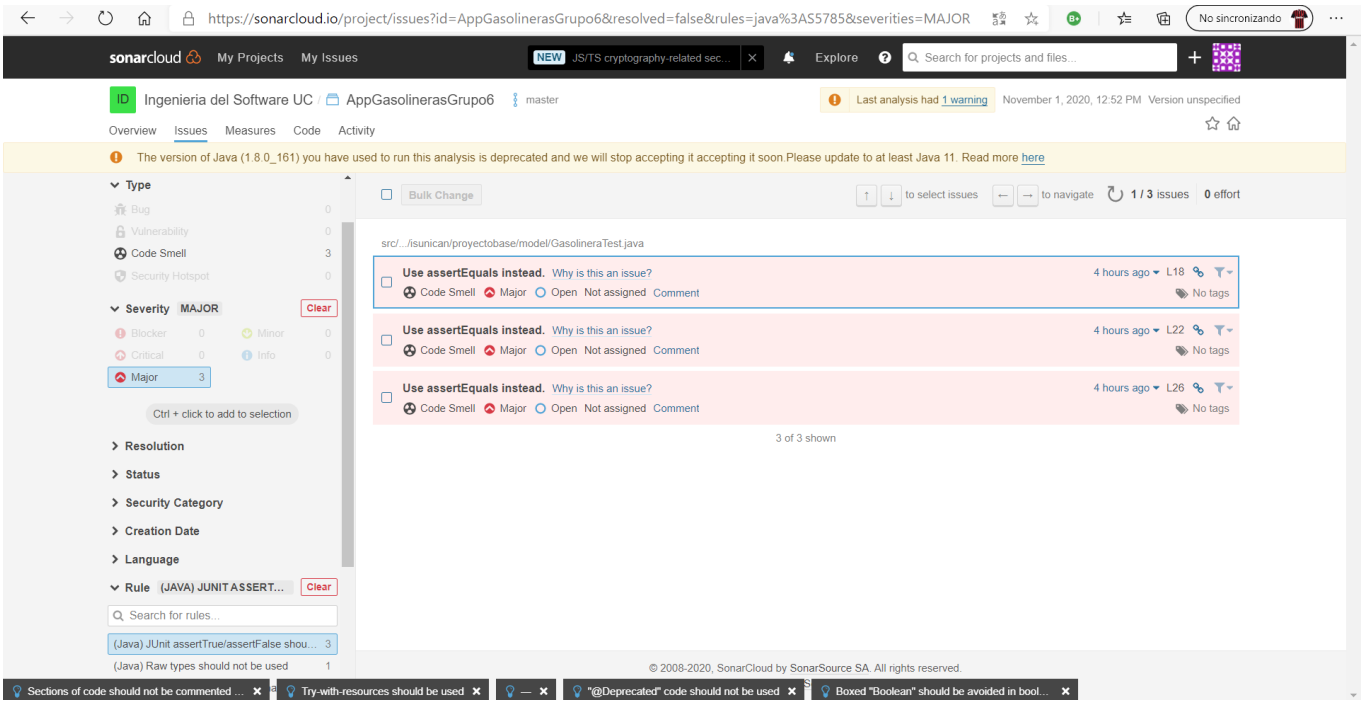
ANÁLISIS 1 NOVIEMBRE 2020

CAPTURA



INCIDENCIAS

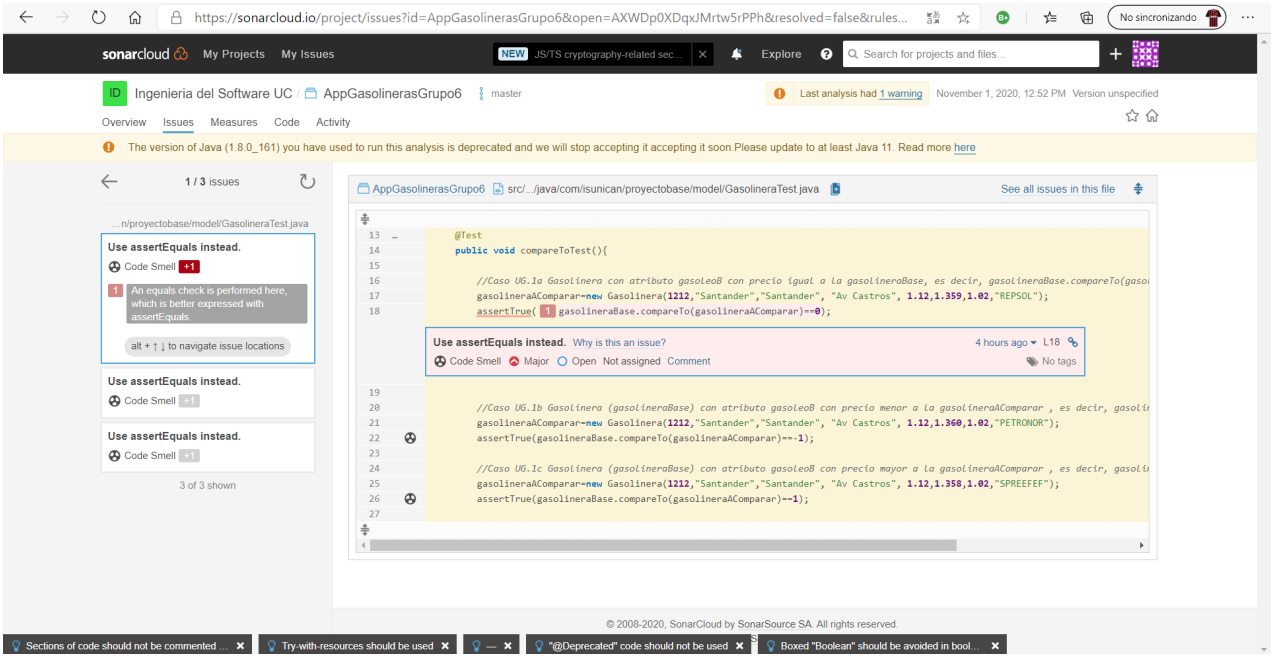
Como se puede observar en la siguiente imagen de las issues que hay (12), las que más severidad tienen son 5, de las cuales son 3 las que incumplen más veces la misma regla, "JUnit assertTrue/assertFalse should be simplified to the corresponding dedicated assertion", lo que quiere decir que los asserts que se usan deberían de ser otros para el tipo de comprobación que hago, los que se usan valen pero no son los más idóneos.

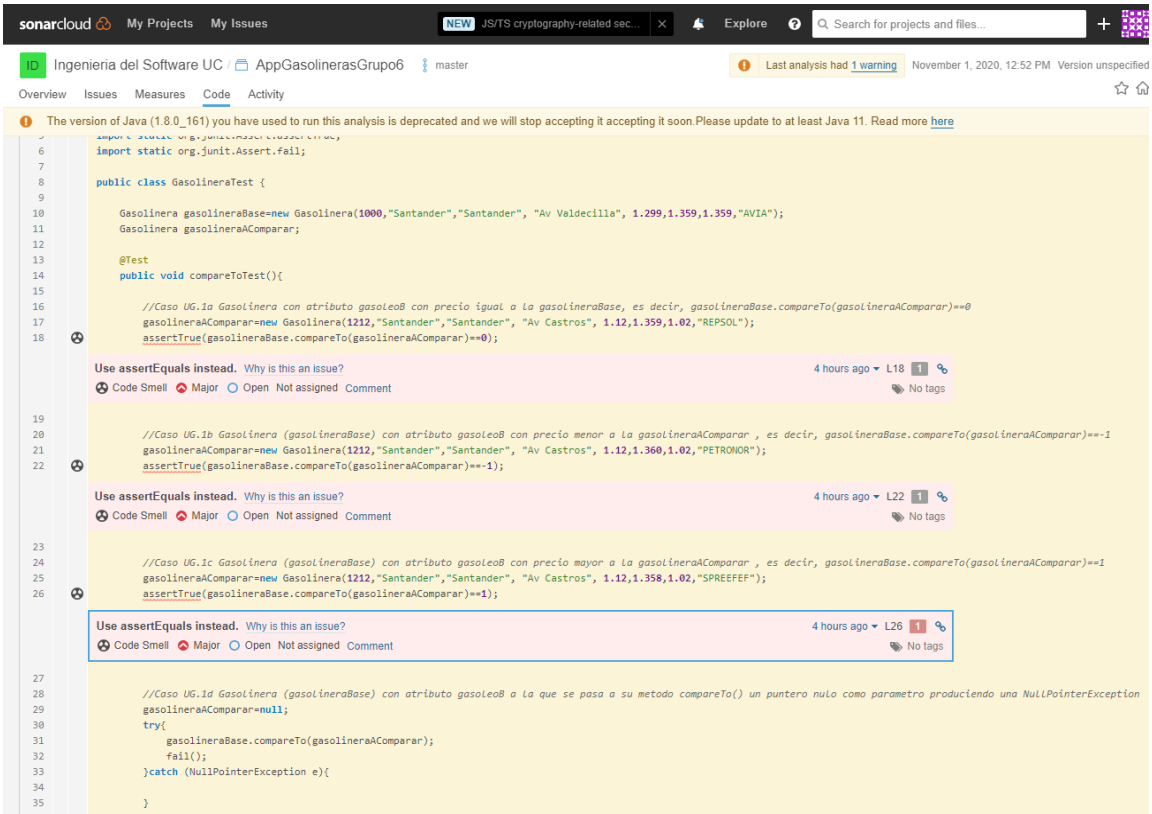


Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

En la siguientes imágenes se ve más claramente donde se producen estos code smells, evidentemente se van a producir en clases de prueba. Se me aconseja convertir mis assertTrue, en assertEquals que son más específicos. Todos se encuentran en la clase GasolineraTest.





Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

PLAN DE ACCIÓN

1) Eliminar todos los issues de la clase de prueba GasolineraTest dejándola limpia de code smells. Siendo además estos de severidad “Major”.

Comentarios:

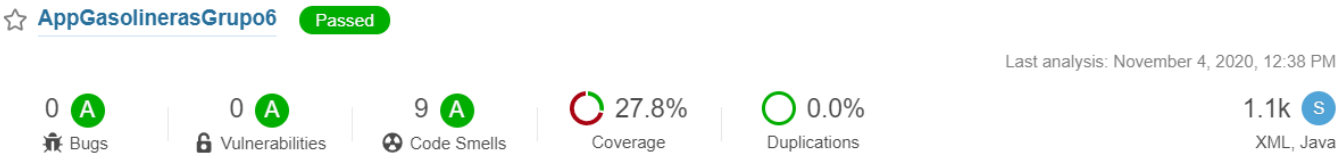
- Los issues restantes después de las correcciones serán 9, de los cuales 6 no se deben de corregir ya que son por código obsoleto, si lo hacemos puede que la aplicación deje de funcionar por desconocimiento de la tecnología.

Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

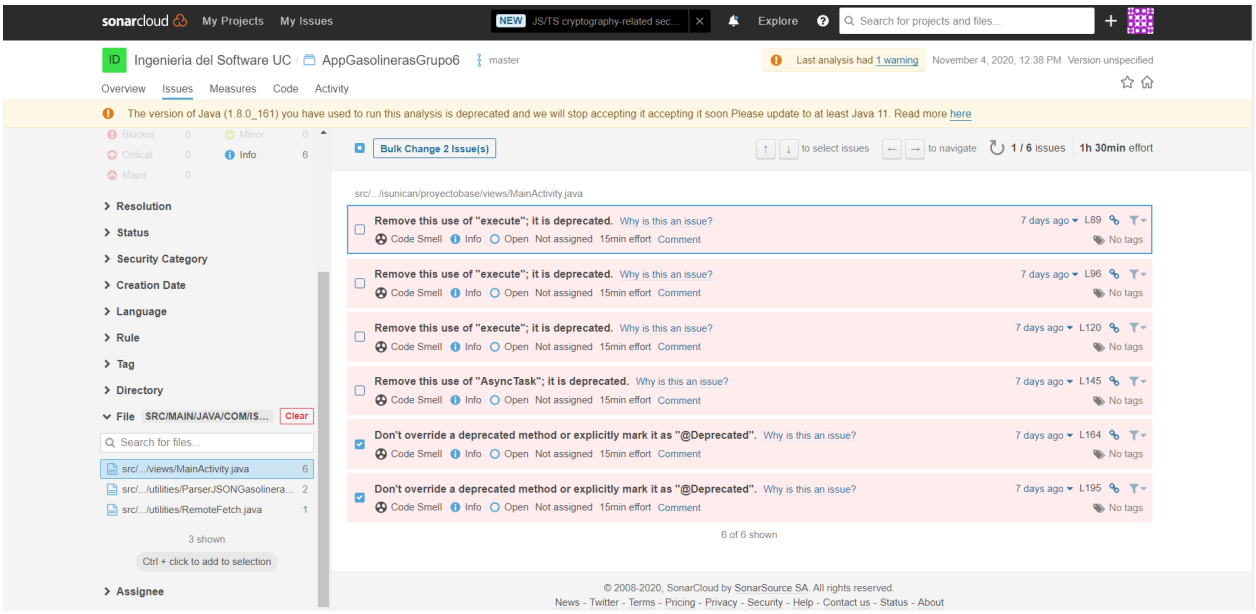
ANÁLISIS 4 NOVIEMBRE 2020

CAPTURA



INCIDENCIAS

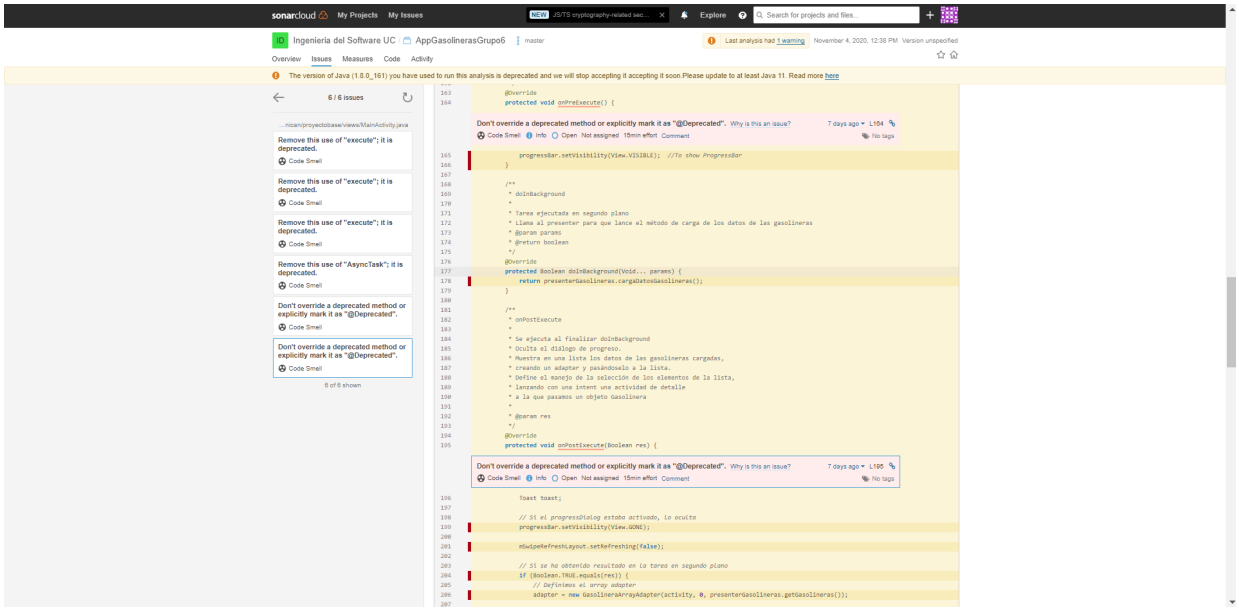
Como se puede observar en la siguiente imagen de las issues que hay (9), el fichero que más deuda técnica genera es MainActivity, el cual tiene 6 code smells. De ellos solo se pueden corregir dos (los que están marcadas con un tick en la imagen), con la seguridad de que la App siga funcionando,siendo una corrección incompleta. Los dos code smells generan una deuda técnica conjunta de 30 min.



Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

Son específicamente las issues que aparecen en la siguiente imagen:



PLAN DE ACCIÓN

1) Corregir los únicos code smells de mayor deuda técnica que se pueden corregir dentro de la Activity MainActivity. Hay que indicar que el método onPreExecute() está obsoleto, para ello hay que colocar sobre este la etiqueta "@Deprecated" y un comentario sobre el método que dé información y que empiece por @deprecated. Ejemplo:

```
/*
*@deprecated este método esta obsoleto y pronto no podrá ser usado
*/
```

2) Hay que indicar que el método onPostExecute() está obsoleto, para ello hay que colocar sobre este la etiqueta "@Deprecated" y un comentario sobre el método que de información y que empiece por @deprecated.

Comentarios:

- Hay que recordar que estas correcciones solo hacen más visible al resto de desarrolladores que se esta usando un código obsoleto, todo con el fin de que estos lo actualicen por otro vigente que realice la misma funcionalidad.

Informe de Calidad (Sprint 1)

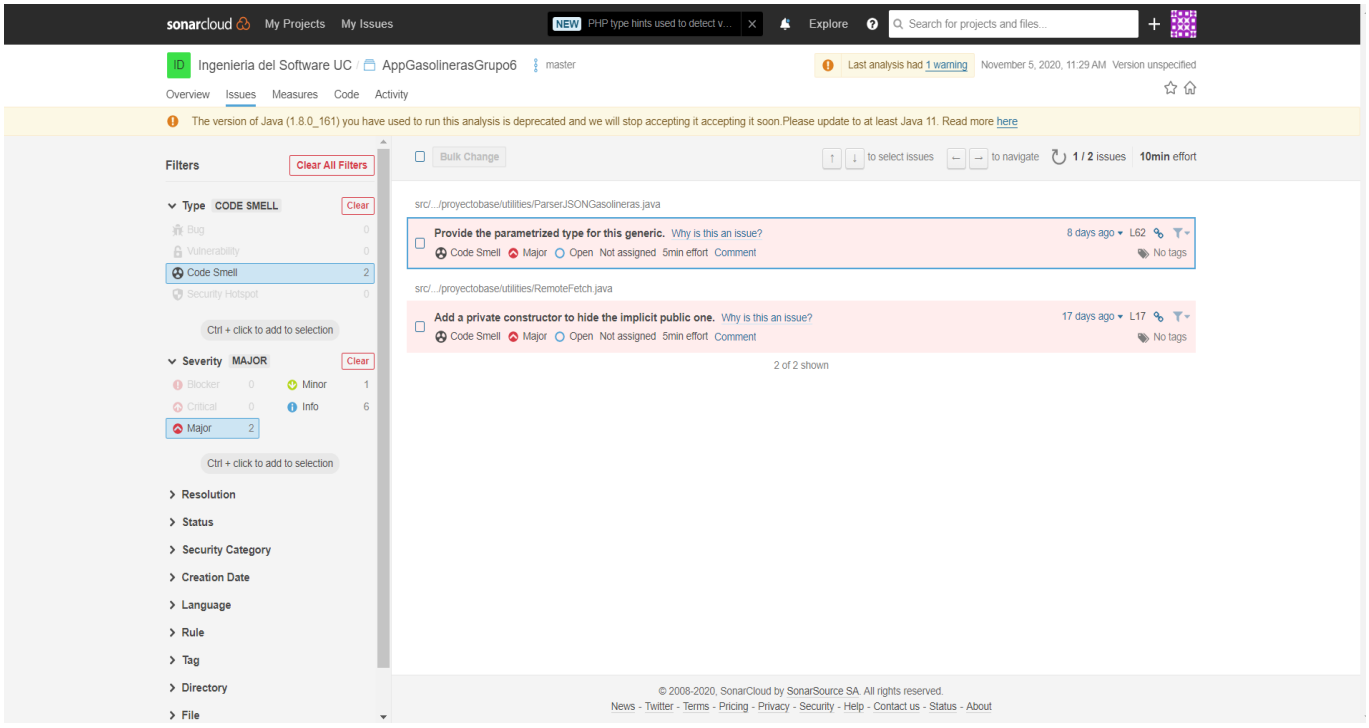
Autores: Laureano Ateca González

ANÁLISIS 5 NOVIEMBRE 2020

CAPTURA

INCIDENCIAS

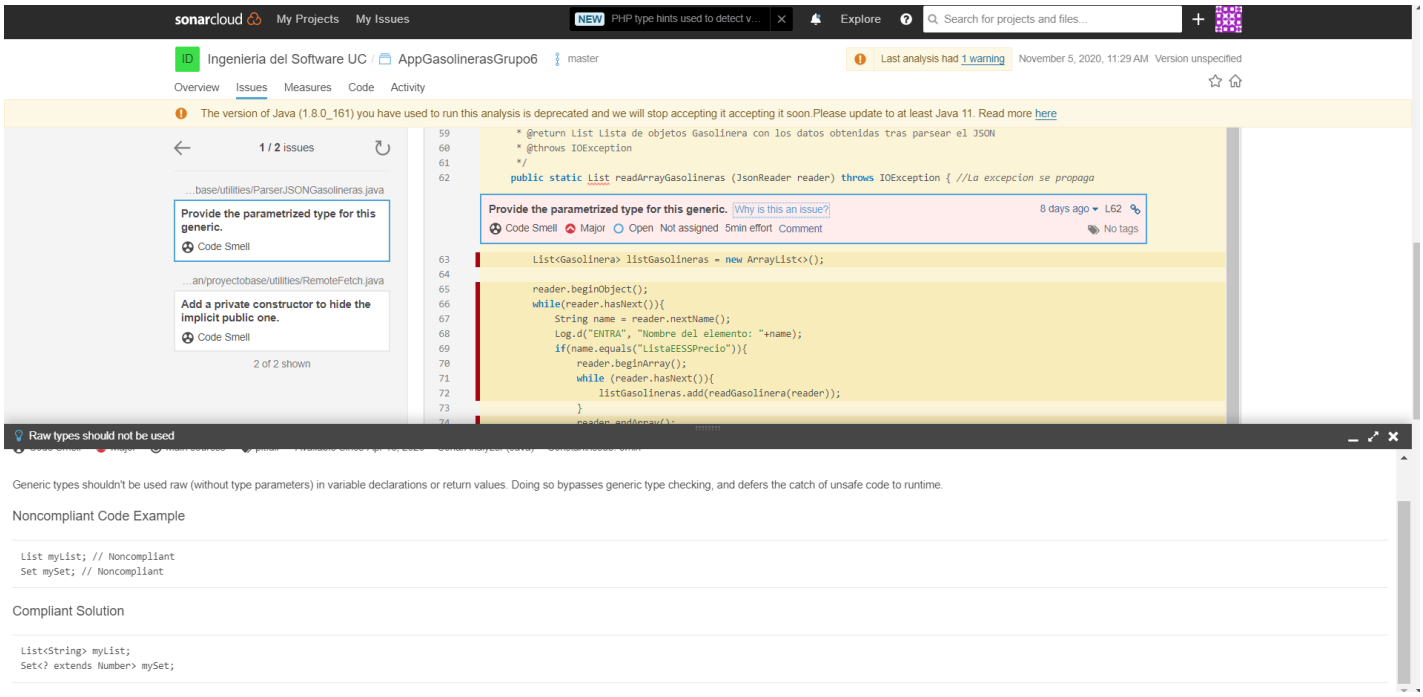
Hay un total de 9 issues, las dos corregidas en la anterior entrada de este informe se han convertido en otras que dan el aviso de que en algún momento el código etiquetado como obsoleto tiene que sustituirse por otro que no lo sea. De las issues que realmente podemos solucionar, las cuales son 3, dos de ellas tienen una severidad destacada (Major) con una deuda técnica conjunta de 10 minutos. Estas issues son code smells como se puede observar:



Informe de Calidad (Sprint 1)

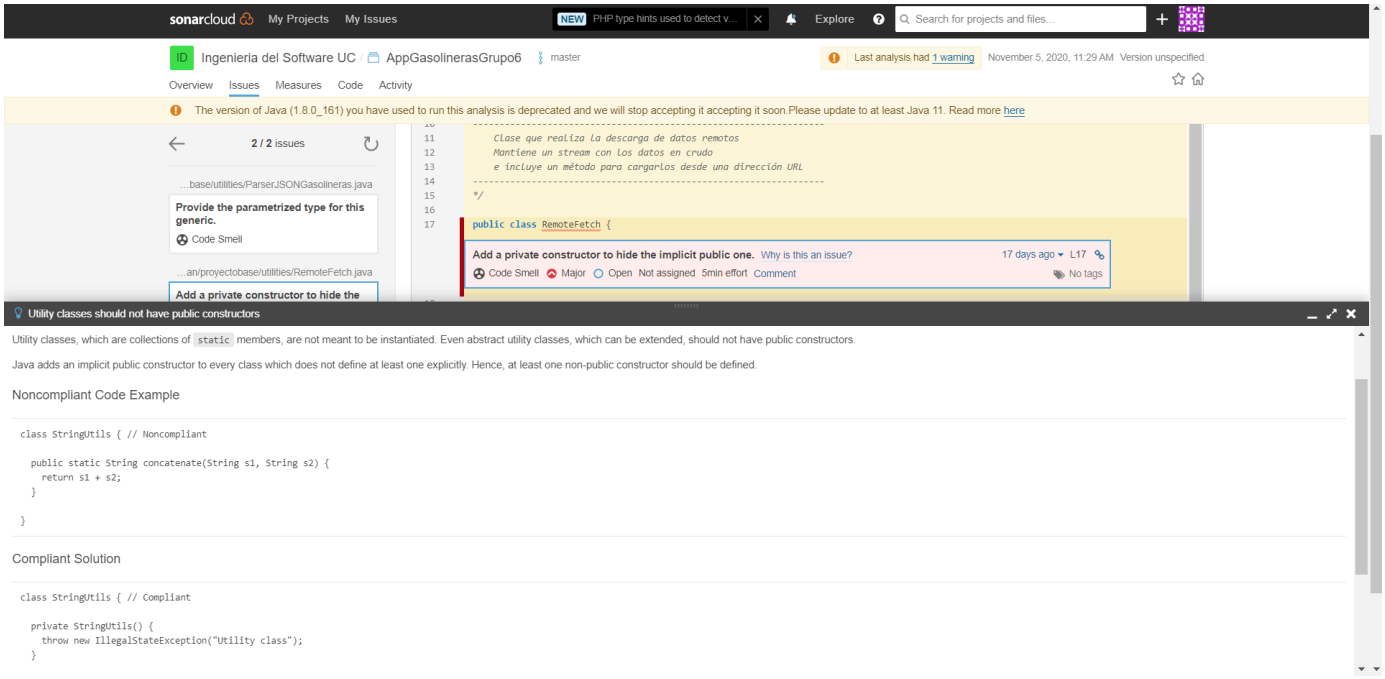
Autores: Laureano Ateca González

El primer code smell corresponde a la clase ParserJSONGasolineras, y se puede ver en detalle qué parte del código provoca el problema, así como la forma de solucionarlo:



El problema de este primer code smell es que solo se indica List como tipo para la salidad del método, siendo List una interfaz genérica.

El segundo code smell corresponde a la clase Remote Fetch, en la que se indica claramente el problema, y es que no se especifica de forma explicita un constructor privado para esta clase con métodos estáticos, la cual nunca se va a instanciar. Se observa en la siguiente imagen:



Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

PLAN DE ACCIÓN

2) Indicar para la clase ParserJSONGasolineras, en el método readArrayGasolineras(), que la salida en lugar de List sea List<Gasolinera> .
3) Añadir para la clase RemoteFetch un constructor privado vacío:

```
private RemoteFetch(){

}
```

Comentarios:

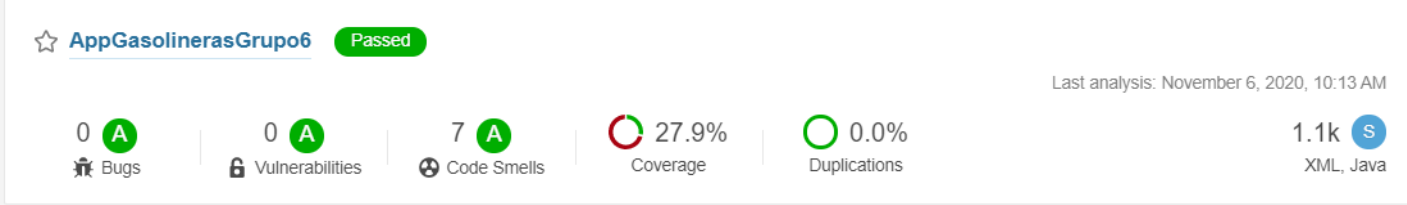
- Aunque se sigan pasando los estándares de calidad, siempre hay que buscar mejorar la calidad del producto.

Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

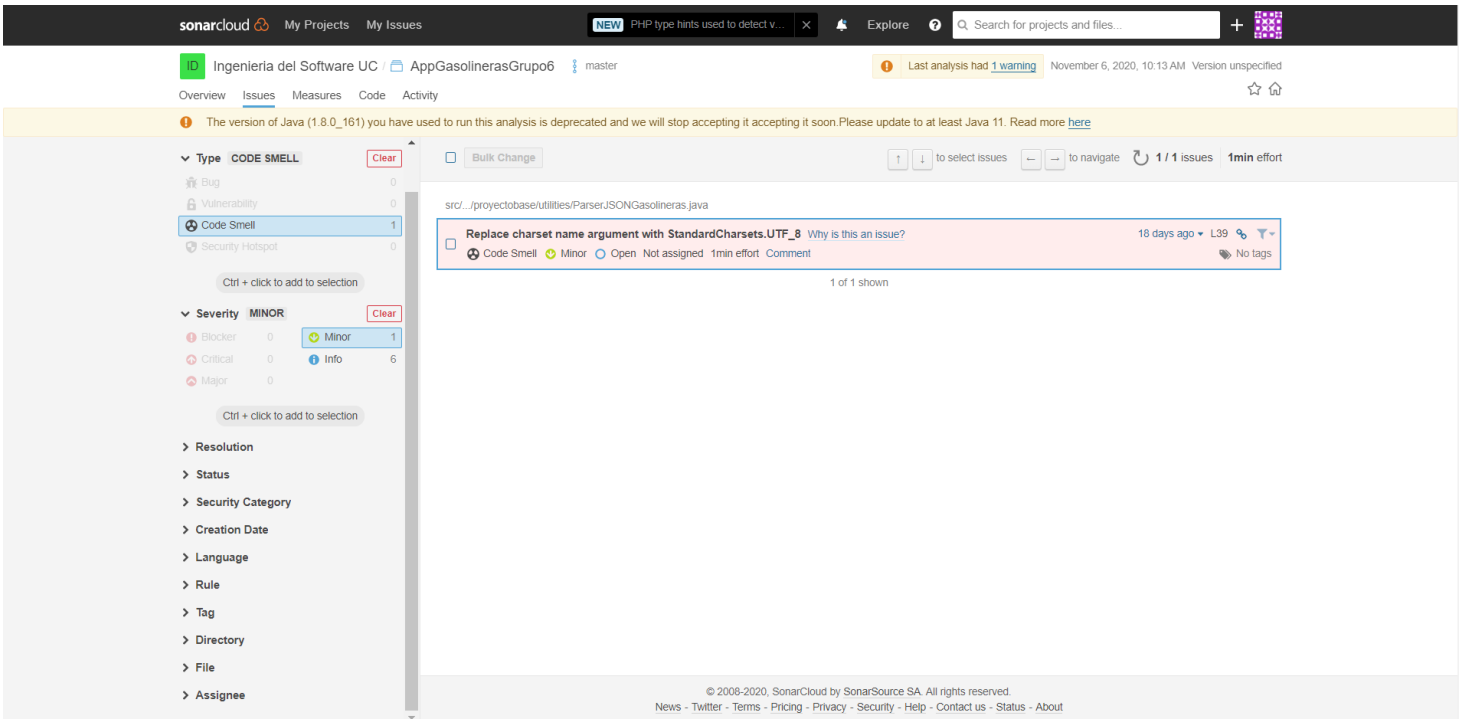
ANÁLISIS 6 NOVIEMBRE 2020

CAPTURA

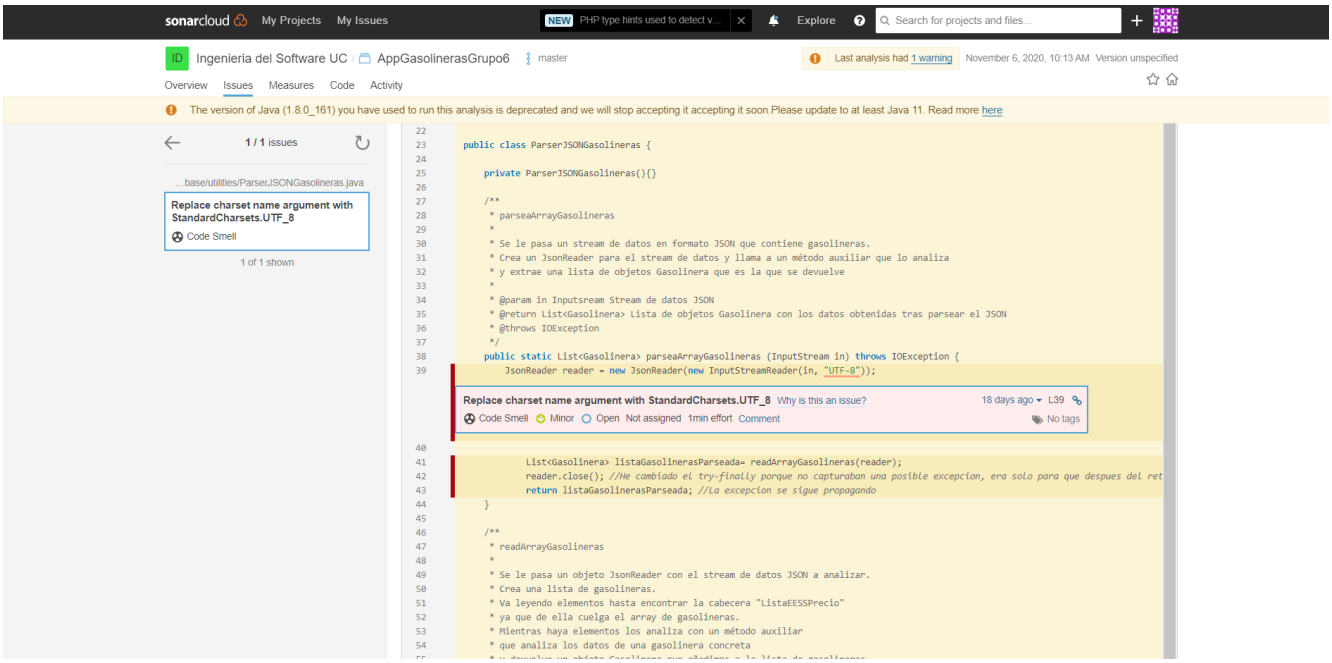


INCIDENCIAS

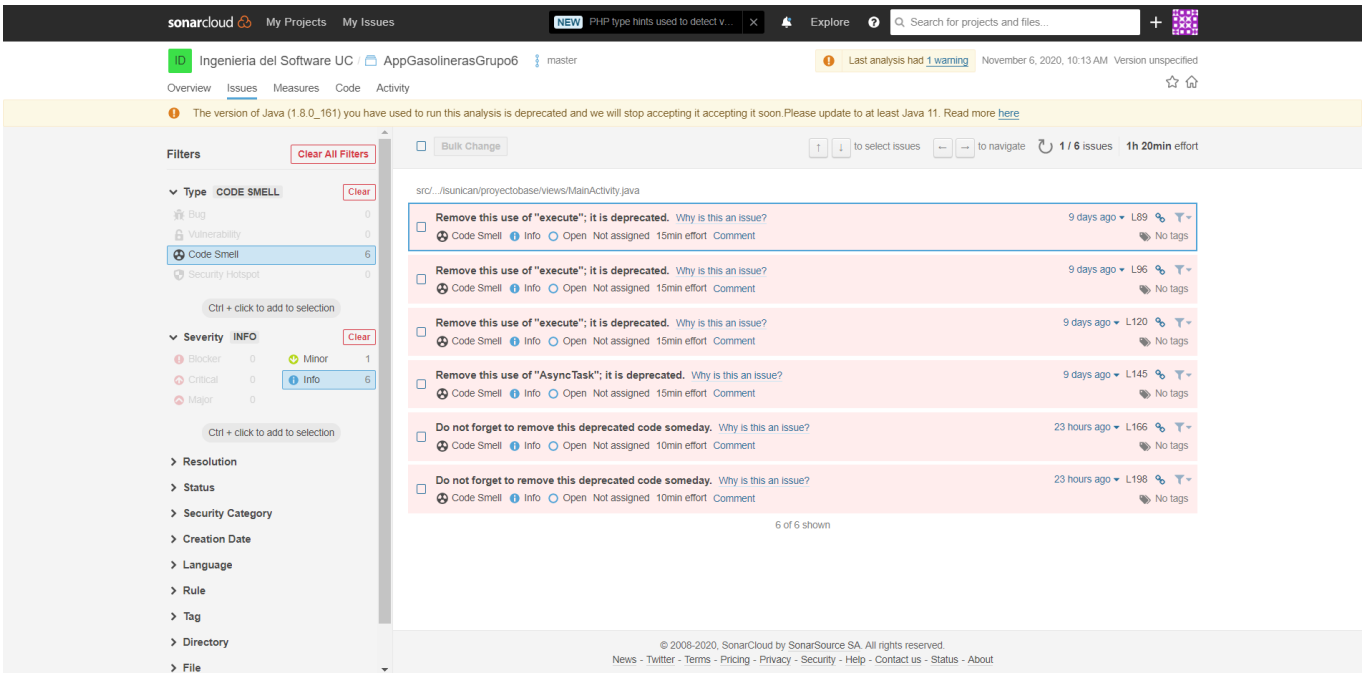
De las 7 issues que quedan en el código solo queda una que tenga una solución óptima. Se trata de un code smell que se encuentra en la clase ParserJSONGasolineras, más específicamente en la instanciación del JsonReader del método parseaArrayGasolineras(). El problema reside en que en lugar de usar un atributo estático definido para dar el valor UTF-8, se usa un String que contiene el mismo valor, quedando la clase que proporciona dicho atributo estático (constante) desaprovechada. Se puede observar en la siguiente imagen:



Se puede analizar aun mejor en su contexto, viendo el código fuente que provoca el code smell:



El resto de issues es desaconsejable abordarlas por mantener la funcionalidad de la App asegurada. Se observa que dichas issues son de severidad informativa, pero que introducen mucha deuda técnica debido a que su corrección implica reconstruir una gran cantidad de código:



Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

PLAN DE ACCIÓN

1) Sustituir el String “UTF-8” por el atributo estático (la constante), el cual es más adecuado. De forma que:

   JsonReader reader = new JsonReader(new InputStreamReader(in, **"UTF-8"**));

   Se sustituiría por:

   JsonReader reader = new JsonReader(new InputStreamReader(in, **StandardCharsets.UTF_8**));

Comentarios:

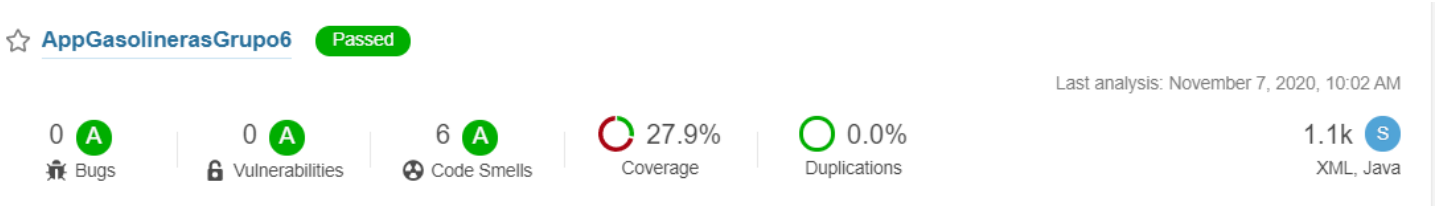
- Ya no hay más issues que se puedan corregir con la seguridad de obtener el resultado deseado.

Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

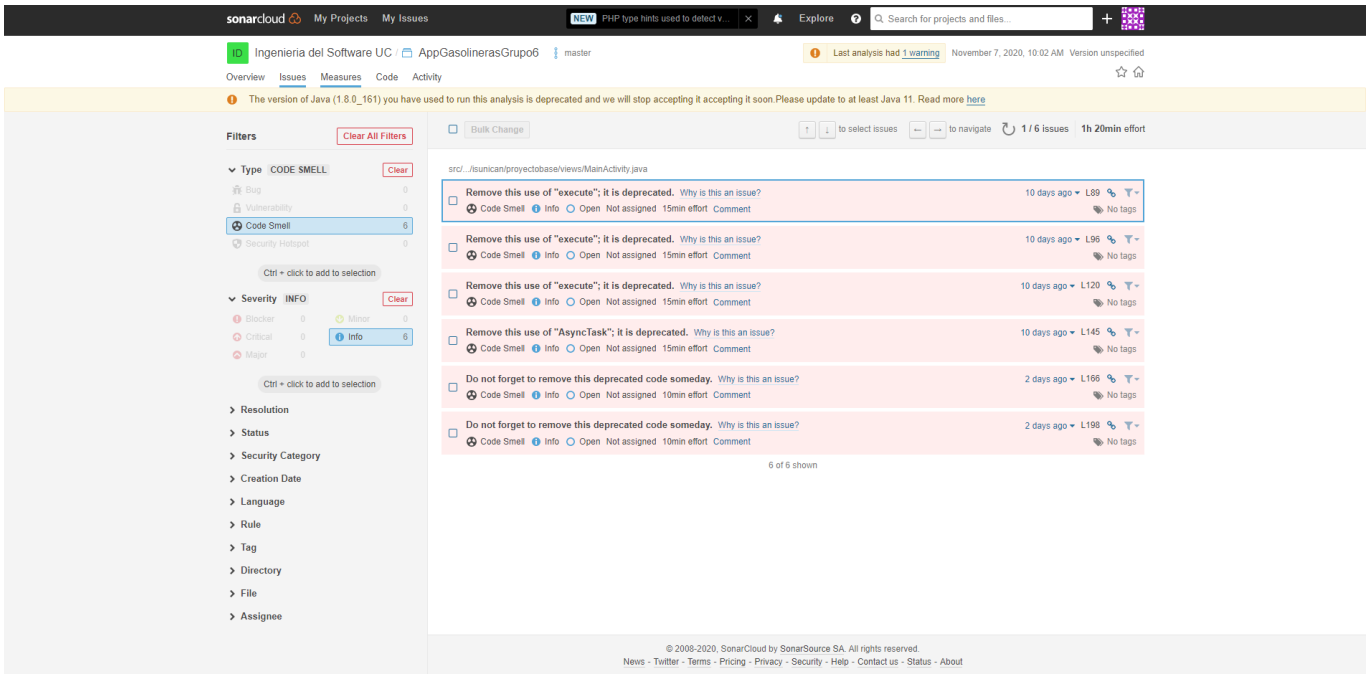
ANÁLISIS 7 NOVIEMBRE 2020

CAPTURA



INCIDENCIAS

Quedan 6 issues que son code smells de severidad informativa, pero que introducen una deuda técnica de 1h 20 min. Dichos code smells no son triviales de resolver, y por asegurar que el funcionamiento de la App quede intacto, de momento, no se van a abordar. Debido a esta situación nos dispondremós a realizar un análisis de la situación actual de la calidad de nuestro producto.

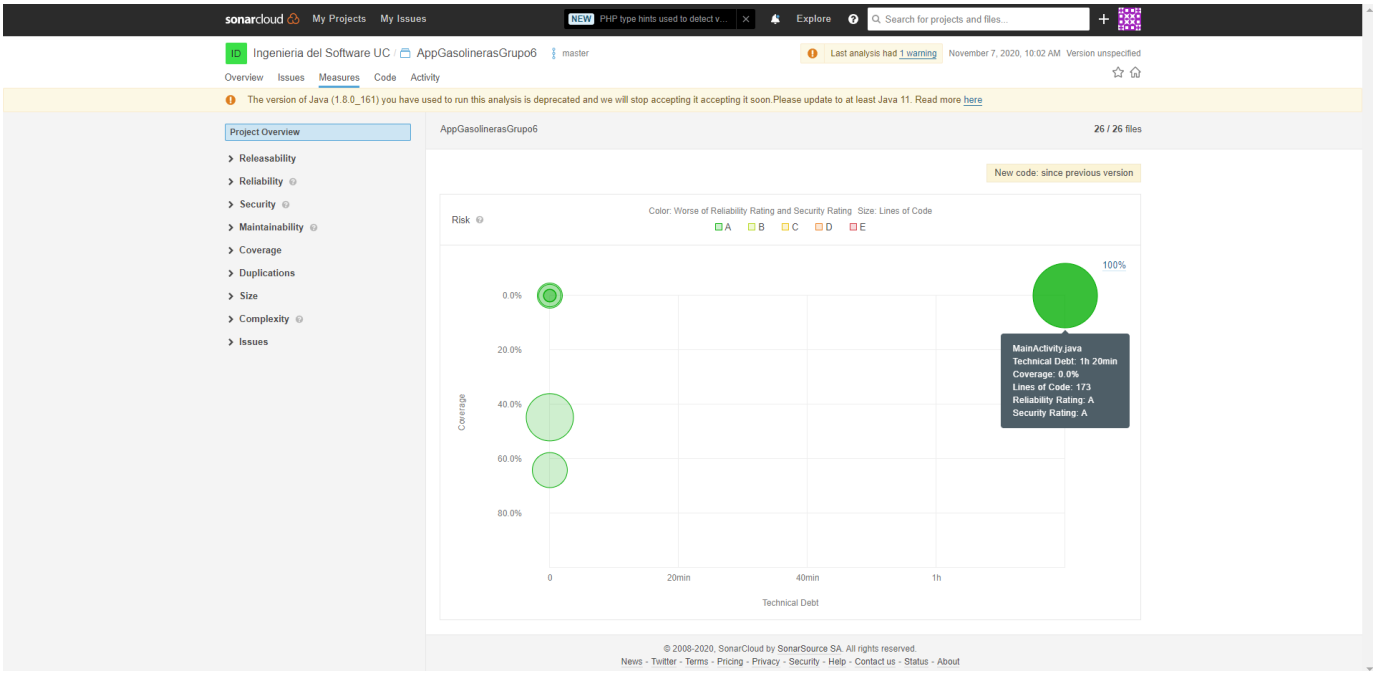


Estos son los code smells mencionados, todos resultado del uso de código que es obsoleto y cuyo uso desaparecerá próximamente. Se pueden observar todos los aspectos mencionados previamente.

Informe de Calidad (Sprint 1)

Autores: Laureano Ateca González

Ahora vamos a analizar en la siguiente gráfica la situación del producto:



En esta se pueden analizar tanto la cobertura de las diferentes clases y activities como la deuda técnica. La deuda técnica del resto de clases tiene un valor de 0 min, que se representa según la posición del eje X, siendo el elemento situado más a la derecha el que mayor deuda técnica posee; este es el caso de la MainActivity con sus 6 issues por código “deprecated” y con una deuda técnica de 1 hora y 20 minutos, siendo cada una de las issues de entre 10 y 15 minutos. La cobertura, podemos observar, que se representa en el eje Y, siendo el elemento situado más abajo en este eje el que mayor cobertura de código tiene en cuanto a pruebas; el elemento que más cobertura tiene es la clase ParserJSONGasolineras, con una cobertura del 64,1%, justo después se encuentra la clase Gasolinera con un 44,7% de cobertura. Respecto al conjunto de líneas de código se obtiene una cobertura del 27,9%.

PLAN DE ACCIÓN

1) De momento no se debe hacer otra cosa que obtener información para sustituir el código obsoleto por otro que realice la misma funcionalidad siendo más actual.

Comentarios:

- Asegurar la calidad del producto es una tarea continua cuyo afán es ofrecer la mejor versión posible del trabajo. De ello depende distinguirse de la competencia entre otras cosas.