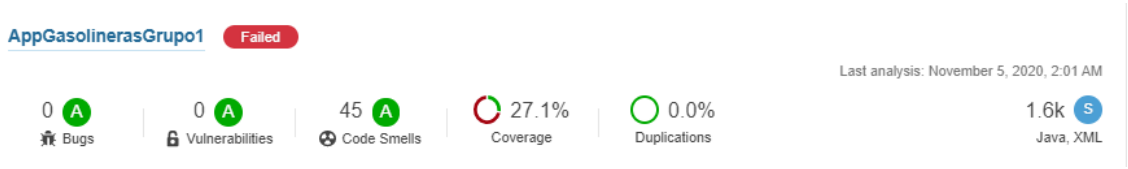
ANÁLISIS 4 NOVIEMBRE 2020:

- Adrián Celis Fernández

CAPTURA:



ANÁLISIS:

Tras estudiar el análisis que realiza SonarCloud, podemos observar que el código contiene 45 code smells agrupados en: 5 bloqueantes, 5 críticos, 7 mayores, 10 menores y 18 de info.

Cabe destacar que en la clase SideBarUITest, el método sidebarHasAllItems se encuentra completamente comentado en su interior debido a que el test no pasa con éxito, y no se ha conseguido encontrar solución. Por lo tanto, el bug smell bloqueante se ignorará.

Esto último ya se comentó en la anterior revisión, por lo que sigue siendo necesario para poder analizar el código.

PLAN DE ACCIÓN: más grave a menores.

1. Sobre la clase BrandExtractorUtil nos encontramos con 5 code smells que nos restarían 32 minutos de deuda técnica.
   El primero consistiría en la eliminación de imports que no se utilizan
   El segundo sería añadir un constructor privado a la clase, ya que las clases de Utilities son colecciones de métodos estáticos por lo tanto no están diseñadas para ser inicializadas. Java implementa un constructor implícito a no ser que se le especifique uno, que debería de ser privado.
   El tercero es que el parámetro gasolineras del método extractBands debería de realizarse con una implementación de List en vez de ArrayList ya que a la hora de generar Utilities se deben de ayudar a definir una herencia de interfaces, de manera que se oculten los detalles de implementación.
   El cuarto consiste en lo mismo que el tercero, pero en el método applyFilter.
   El quinto indica que el toLowerCase y el equals se pueden sustituir por el método equalsIgnoreCase que realiza la misma funcionalidad.
2. Sobre la clase MainActivity se pueden realizar ciertos cambios que consisten en:
   -Renombrar la variable btn_positivo para que cumpla los estándares de estilo.

-Eliminar código comentado (línea 152).
-Cambiar el método onOptionsItemSelected para que no devuelva siempre verdadero.
-Renombrar o reutilizar la variable adapter situada en la línea 239 que hace sombra a la misma variable situada en la línea 75.
-Eliminar la especificación del tipo a la hora de crear el nuevo ArrayList en la línea 239.
-Exactamente lo mismo que el anterior, pero en la línea 247.
-Lo mismo, pero en la línea 331.
-Eliminar código comentado situado entre las líneas 304 y 307.
-Como ya se pidió en el anterior informe, usar un acceso estático con android.content.DialogInterface para BUTTON_POSITIVE ya que los miembros estáticos de una clase base no deberían de ser accedidos haciendo uso de una clase hija.
-Como se comentó en el informe previo, realizar comentarios dentro de los métodos beforeTextChanged y onTextChanged si estos métodos no se van a implementar en ningún momento y se necesitan de esta forma.
-Renombrar o reutilizar la variable adapter situada en la línea 423 que hace sombra a la misma variable situada en la línea 75.
-Eliminar la especificación del tipo a la hora de crear el nuevo ArrayList en la línea 423.
-En la línea 546, en vez de usar el if con la expresión básica de indicar el parámetro res pasado al método, usar Boolean.TRUE.equals(res).
Eliminar casteos innecesarios a ArrayList situados en las líneas 548 y 550.
-Con todos estos cambios reduciríamos 1 hora en total de deuda técnica lo que nos permitiría tener un margen de error para posibles implementaciones como ha ocurrido la última vez, ya que la gran mayoría de los errores indicados se corrigieron, permitiendo que esta vez el número de fallos no haya añadido una deuda técnica tan alta

COMENTARIOS:

1. Con todas las correcciones descritas obtendríamos una reducción de 1h 32 minutos, y nos quedaríamos con un tiempo de deuda por debajo de la implementación previa con los cambios realizados.
2. Fijándonos en los code smells bloqueantes nos encontramos con que tenemos una deuda técnica de 30 minutos en la clase PresenterFiltrosMarcasTest. Estos code smells se intentaron corregir, pero tras el análisis se decidió que la clase iba a ser eliminada o implementada en algún punto posterior, por lo que de momento no se tiene en cuenta sobre el análisis de calidad.