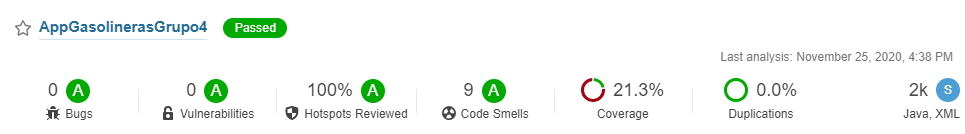
**Informe de Calidad (Sprint 3)**

**Autores: Daniel Llovio del Río y Víctor Pérez Aberasturi**

**ANÁLISIS 26 NOVIEMBRE 2020**

**CAPTURA**



**INCIDENCIAS**

El análisis de calidad supera los criterios definidos por la organización, con una calificación de A. No se han encontrado nuevos bugs ni vulnerabilidades en el código, por lo que no se observan cambios en la deuda técnica restante, que permanece en 1 hora y 37 minutos.

Con la nueva implementación del código, se han solucionado dos de los *code smells* detectados en el último análisis. Sin embargo, se han generado otros 2 nuevos *issues* de mantenibilidad, siendo ambos de *minor severity* y estando los dos alojados en la clase MainActivity.java

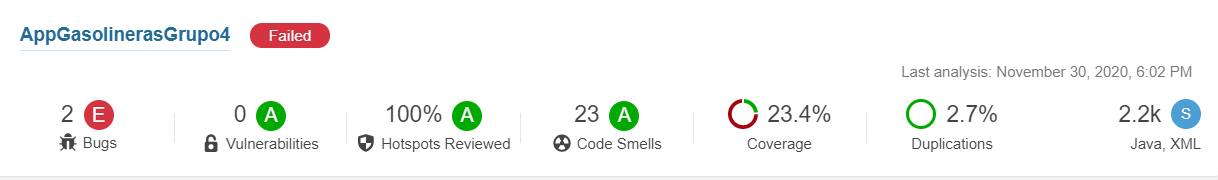
**PLAN DE ACCIÓN**

1. Arreglar el *code smell Minor* localizado en la clase MainActivity, que agrega hasta 5 minutos de deuda técnica, eliminando la variable local no usada gasolinerasFiltradas del método myClickHandler().
2. Arreglar el segundo *code smell Minor* de MainActivity, que añade hasta 2 minutos de deuda técnica, borrando el import java.util.ArrayList que se encuentra al comienzo de la clase.

**COMENTARIOS**

Los nuevos *code smells* que se han generado no tienen un gran impacto y son de rápida solución. Los restantes, dada su naturaleza, se han descartado del plan de acción ya que al ser del tipo *deprecated*, supondrían reemplazar una gran parte del código, por lo tanto, serán atajados más adelante, ya que actualmente la calidad del proyecto es sobresaliente.

**ANÁLISIS 30 NOVIEMBRE 2020**



**INCIDENCIAS**

El análisis de calidad no supera los criterios definidos por la organización, con una calificación de E. Como podemos ver en la captura, han aparecido dos nuevos *Bugs* y 14 *code smells* más respecto al anterior informe. Estos nuevos *bugs* y los *code smells*, han provocado un aumento sustancial en la deuda técnica, en este caso la deuda pasa a ser de 3 horas 54 minutos.

En este caso, se van a intentar atajar los *code smells* de tipo *Blocker* (1 en este caso), *Critical* (3) y *Major* (2). Los de tipo *Minor* (6) e *Info* (8) se dejarán debido a su poca gravedad y el resto de *code smells* de tipo *Critical(4) y* Major (1) se dejarán para resolver más adelante. Por tanto, el plan de acción será el siguiente:

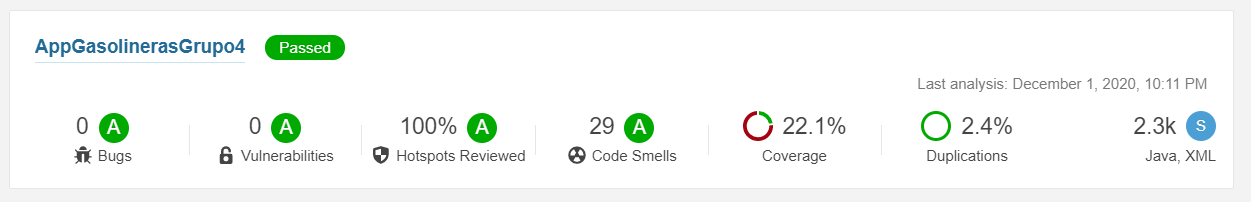
**PLAN DE ACCIÓN**

1. Arreglar el *bug* de tipo *Blocker* localizado en la clase PresenterVehiculos, que agrega hasta 5 minutos de deuda técnica, cerrando el “*FileWriter”* en la cláusula *Try/Catch* en el *Finally* del método borraSeleccionados().
2. Arreglar el *bug* de tipo *Major* localizado en la clase VehiclesActivity, que añade una deuda técnica de 15 minutos, eliminando la llamada al método equals() en la comparación “if( seleccionado && vehículo.**equals**(v.findViewById(…)))” ya que siempre devuelve falso en el método seleccionarVehiculo().
3. Arreglar *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos, que añade una deuda técnica de 10 minutos, definiendo una constante para el literal repetido “login activity” en el tratamiento de excepciones en la parte que ejecuta “*catch (FileNotFoundException e) {Log.e(“****login activity****”, “…”)}”* y en “*catch (IOException e) {Log.e(“****login activity****”, “…”)}”* en los métodos consultaDB() y consultaDBSeleccionado
4. Arreglar *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos, que añade una deuda técnica de 8 minutos, definiendo una constante para el literal “Fallo al escribir en la base de datos” que se repite 3 veces y se encuentra en el tratamiento de excepciones “*catch (IOException e) {Log.e(“Excepcion”, “***Fallo al escribir en la base de datos***”)}”* en el método borraSeleccionados(), escribeVehiculoSeleccionado() y consultaDBSeleccionado().
5. Arreglar *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos, que añade una deuda técnica de 8 minutos, definiendo una constante para el literal “seleccionado.txt” que se repite 3 veces y se usa en la creación del FileWriter outputStreamWriter en el método borraSeleccionados(), escribeVehiculoSeleccionado() y consultaDBSeleccionado().
6. Arreglar *code smell* del tipo *Major* localizado en la clase PresenterVehiculos, que añade 15 minutos de deuda técnica, eliminando la variable local “outputStreamWriter” debido a que no se usa en el método borraSeleccionados.

**COMENTARIOS**

Al realizar este plan de acción la deuda técnica se rebajaría en 1 hora y 1 minuto. Una vez solucionados los *bugs* *sonar* debería de pasar ya que no ha fallado por la deuda técnica, si no que ha fallado por estos bugs. Igualmente, arreglando 4 *code smells* nos alejaríamos de las 4 horas de limite permitiendo ser mas laxos en futuros análisis.

**ANÁLISIS 2 DICIEMBRE 2020**



**INCIDENCIAS**

El análisis de calidad supera los criterios definidos por la organización, con una calificación de A. Como podemos ver en la captura, han desaparecido los *Bugs* que se notificaron en el anterior análisis, pero han aparecido otros tantos *code smells*. Como podemos ver, tenemos 29 *code smells* que han provocado una deuda técnica de 3 horas y 44 minutos y que se acerca peligrosamente al límite de 4 horas.

Dado que nos hemos acercado de forma peligrosa a la deuda técnica, vamos a tratar de resolver los *code smell* que más deuda nos este provocando. En este análisis hemos encontrado 9 *code smells* de tipo *Critical*, 3 *Major*, 10 *Minor* y 7 *Info. Vamos a resolver 5 code smells* de tipo *Critical* y un *code smell* de tipo *Major*. Por tanto, el plan de acción será el siguiente:

1. Arreglar el *code smell* de tipo *Critical* localizado en la clase VehiclesActivity y que nos provoca una deuda técnica de 8 minutos. Para ello, definiremos una variable de tipo *String* que contendrá el literal “VALOR” que aparece tres veces en el método seleccionarVehiculo().
2. Arreglar el *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos y que nos provoca una deuda técnica de 8 minutos. Para ello, definiremos una variable de tipo *String* que contendrá el literal “Excepcion” que aparece tres veces en el *catch* de las excepciones en el método borraSeleccionados ().
3. Arreglar el *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos y que nos provoca una deuda técnica de 8 minutos. Para ello, definiremos una variable de tipo *String* que contendrá el literal “GasoleoA” que aparece tres veces en distintos condicionales de tipo *if* de las excepciones en el método anhadirVehiculo ().
4. Arreglar el *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos y que nos provoca una deuda técnica de 8 minutos. Para ello, definiremos una variable de tipo *String* que contendrá el literal “Gasolina95” que aparece tres veces en distintos condicionales de tipo *if* de las excepciones en el método anhadirVehiculo ().
5. Arreglar el *code smell* de tipo *Critical* localizado en la clase PresenterVehiculos y que nos provoca una deuda técnica de 6 minutos. Para ello, deberemos rebajar la complejidad cognitiva máxima permitida de 31 a 30 refactorizando condicionales *if* o algún bucle *for* en el método anhadirVehiculo ().
6. Arreglar el *code smell* de tipo *Major* localizado en la clase PresenterVehiculos y que provoca una deuda técnica de 50 minutos. Para ello, deberemos eliminar 4 bloques de código duplicados en el método anhadirVehiculo() y que corresponden a los *ifs* para controlar la aparición de vocales en las matrículas de los vehículos.

**COMENTARIOS**

Al realizar este plan de acción la deuda técnica se rebajaría en 1 hora y 28 minutos. Como podemos ver, los 4 primeros *code smells* son lo mismo y teniendo en cuenta esto, sabemos que tardaremos bastante menos de 8 minutos en resolverlo. Debido a esto, hemos decidido añadir un *code smell Major* con 50 minutos de deuda técnica porque tenemos una idea de como resolverlo y el *code smell* 5 se resolvería al mismo tiempo que resolvemos el *code smell* 6.

Una vez completado este plan de acción, la deuda técnica pasaría a ser de 2 horas y 16 minutos que dará un margen para próximas integraciones.