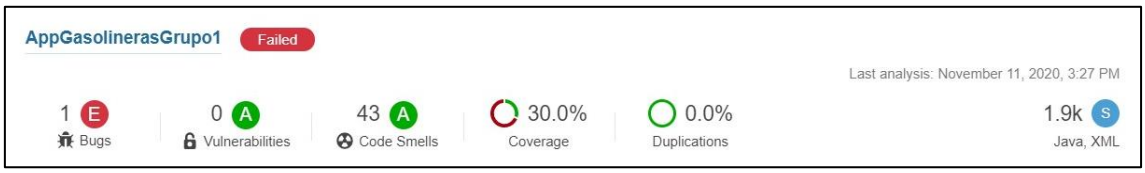
Análisis de Calidad del Producto – Sprint 2

Autores: Carolay B. Corales Acosta y Elena Romón

ANÁLISIS 11 NOVIEMBRE 2020 (Elena)

1. Resultados de Sonar



2. Análisis

El análisis no pasa los criterios de calidad de la organización, ya que la calificación en confiabilidad es E, cuando debería ser A. Esto se debe a un bug que se encuentra en la clase *AppDatabase*.

Por otro lado, en cuanto a mantenibilidad, hay una deuda técnica de 3 horas, debido a 43 *Code Smells*. Estos se encuentran es su mayoría en las clases *DetailActivity* y *Main Activity*, siendo 1 de ellos bloqueante, 2 críticos y 9 mayores, así como 8 menores y 24 de información.

Sin embargo, la deuda técnica no sobrepasa los criterios establecidos por la organización, pero se puede intentar reducir para dejar un mayor margen para las siguientes integraciones.

3. Plan de acción

1. Resolver el bug de la clase *AppDatabase* que ocurre por una doble comprobación del valor de la instancia. Una posible solución es cambiar instanciar el método *getInstance* como *synchronized* (20 *mins effort*).

2. Resolver el *issue* crítico de la clase *ParserJSONGasolineras*, que se encuentra en el método *parseaArrayGasolineras*. Una posible solución es crear el objeto de tipo *JsonReader* dentro del try (15 *mins effort*).

3. Resolver el otro *issue* crítico, que se encuentra en la clase *ThreadAnhadirGasolineras* anidada en *DetailActivity*. El *issue* se debe a que el método *ThreadAnhadirGasolineras* se encuentra vacío, por lo que una posible solución sería simplemente añadir un breve comentario que indique por qué este se encuentra vacío (5 *mins effort*).

4. Resolver el *issue major* de la clase *PresenterGasolinerasFavoritas*, que se encuentra en el setter de la lista de gasolineras. Aunque sonar nos invita a eliminar el parámetro que se le pasa al método, la forma de corregirlo es cambiar la asignación de dentro de este método (5 *mins effort*). Al corregir este *issue*, se corrige automáticamente otro *major*, que pedía eliminar la asignación de una lista a sí misma en el interior de este método (5 *mins effort*).

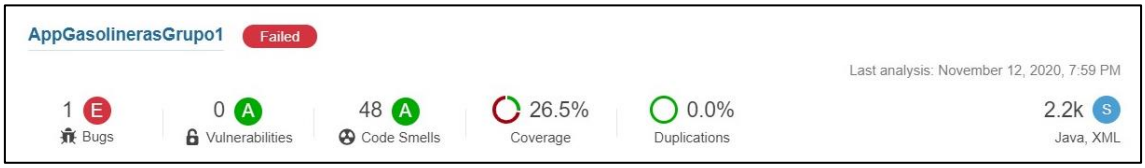
4. Comentarios

Sólo con arreglar el *bug* ya pasaríamos los criterios de calidad establecidos por la organización.

Aunque las horas de deuda técnica estaban dentro de los límites de la organización, se han añadido otros *issues* al plan de acción para ir bajando las horas de deuda técnica, de forma que haya un mayor margen de error en las próximas integraciones. Si se realizan todos los cambios enumerados en el plan de acción, la deuda técnica se reduciría en 50 minutos.

ANÁLISIS 13 NOVIEMBRE 2020 (Elena)

1. Resultados de Sonar



2. Análisis

Para este análisis, aún no se había aplicado el plan de acción del análisis anterior, por lo que todos los issues que se encuentran en ese plan de acción, se van a obviar en este análisis, ya que se entiende que se llevarán a cabo con la mayor prontitud posible.

Dicho esto, el análisis no pasa los criterios de calidad de la organización, por el bug encontrado en la clase *AppDatabase,* ya comentado en el análisis anterior. Sin embargo, tras ejecutar el plan de acción del análisis anterior, el análisis de calidad pasaría con 44 issues (los 48 detectados en este análisis menos los incluidos en el plan de acción anterior), y sin una deuda técnica demasiado preocupante (4 horas, 3 horas tras aplicar el plan de acción anterior).

En cuanto a mantenibilidad, hay una deuda técnica de 4 horas, debido a los 48 *Code Smells*. Estos se encuentran es su mayoría en las clases *DetailActivity* y *Main Activity*, siendo 1 de ellos bloqueante, 2 críticos y 12 mayores, así como 8 menores y 26 de información.

Aunque la deuda técnica no sobrepasa los criterios establecidos por la organización, se empieza a acercar preocupantemente a este límite, así que se debería ejecutar cuanto antes el plan de acción aquí presente, para tener un cierto margen de error de cara a la segunda semana del sprint.

3. Plan de acción

1. Aplicar el plan de acción del análisis anterior, de forma que se elimine el bug y se reduzca la deuda técnica (55 *mins effort*).

2. Reducir los *issues* de la clase con mayor deuda técnica, es decir, *Main Activity*. Al no poder solventar todos los relacionados con el código *deprecated*, no se reducirá demasiado.

   2.1. Eliminar 2 *imports* no necesarios (4 *mins effort*).

   2.2. Cambiar las especificaciones al crear los *ArrayAdapter*, dejando los <> vacíos (2 mins *effort*).

   2.3. Eliminar los castings a *ArrayList* innecesarios (10 mins *effort*).

3. Reducir los *issues* de la siguiente clase con mayor deuda técnica, es decir, *DetailActivity*.

3.1. Eliminar los *imports* no necesarios (2 mins *effort*).

3.2. Eliminar los bloques de código comentados (20 mins *effort*).

4. Comentarios

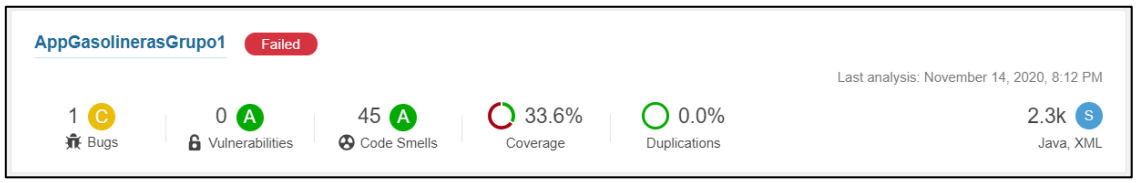
El plan de acción, a excepción del primer punto, se va a centrar en intentar reducir los *issues* de la clase *MainActivity*, ya que es la que mayor deuda técnica tiene (1 hora y 46 mins). Todos los *issues* derivados del uso de código *deprecated* no se han tenido en cuenta en el plan de acción.

Tras analizar el código de la clase *MainActivity* se ha visto que, a no ser que se dejen de usar las llamadas a los métodos *deprecated*, la deuda técnica de esta siempre va a ser muy alta. De esta forma, al aplicar el plan de acción apenas se podría reducir la deuda técnica en 20 minutos. Si la deuda técnica sigue en aumento, habrá que plantearse modificar esta parte de la clase, pero de momento se mantiene.

Para intentar compensar la deuda técnica que no se ha podido reducir en la clase ya mencionada, se ha analizado la siguiente clase con más deuda técnica (*DetailActivity*). Si se aplica el punto 3 del plan de acción se reduciría la deuda en 22 minutos. Cabe destacar que esta clase sigue en desarrollo, y que por tanto no se has tenido en cuenta para el análisis los numerosos comentarios "TODO" del código, ya que se entiende que para futuras integraciones se habrán retirado.

ANÁLISIS 15 NOVIEMBRE 2020 (Carol)

1. Resultados de Sonar



2. Análisis

El análisis no pasa los criterios de calidad de la organización, la calificación en confiabilidad es C y debería ser A. Esto se debe a un *bug* que se produce en la clase *ParsenJSONGasolineras*.

Por otro lado, en cuanto a mantenibilidad, hay una deuda técnica de 4 horas, debido a los 45 *Code Smells*. Estos se encuentran, en su mayoría, en las clases *MainActivity* y *PresenterGasolinerasFavoritas*. Por un lado, los *issues* que nos encontramos en la primera clase son 2 menores y 10 de información, por otro lado, para la segunda clase 2 mayores, 3 menores y 2 de información.

Sin embargo, la deuda técnica no sobrepasa los criterios establecidos por la organización, pero se puede intentar reducir para dejar un mayor margen para las siguientes integraciones.

3. Plan de acción

1. Resolver el bug de la clase *ParsenJSONGasolineras* en la clase *parseaArrayGasolineras.* Este *issue* podría lanzar un *NullPointerException.* (10 min effort).

2. Resolver el *issue* crítico, aunque no esté presente en las clases con más code smells optamos por resolverlo ya que la deuda técnica se reduciría 15 min y es el único *issue* crítico que nos encontramos en el código. Este *issue* se produce en la clase *ParserJSONGasolineras* en la clase *parseaArrayGasolineras.* Una posible solución es crear el objeto de tipo *JsonReader* dentro del try (*15 min effort)*.
3. Resolver 3 *issues* mayores en las clases *DateConverter, CommonUtils* y *ExtractorLocalidadUtil.* El *issue* es el mismo en las tres clases. El esfuerzo es de 5 min cada uno. Se puede resolver añadiendo un constructor privado. (*15 min effort).*
4. Resolver 12 *issues* menores de distintas clases. Estos *issue* conllevan un esfuerzo de 2 minutos cada uno. Para solucionarlo se tienen que eliminar los *imports* que no están siendo utilizados. Se ha optado por resolver estos *issues* ya que el esfuerzo que conlleva quitar los *imports* suponemos que es menor que el que indica sonar y además quitaría una gran cantidad de deuda técnica. Clases que contienen *imports* sin utilizar: 2 en *MainActivity,* 2 en *DetailActivity, 3 en PresenterGasolinerasFavoritas*, 1 en *PresenterGasolineras,* 3 en *GasolineraFavorita* y 1 en *GasolineraDAO.* (*24 min effort)*.

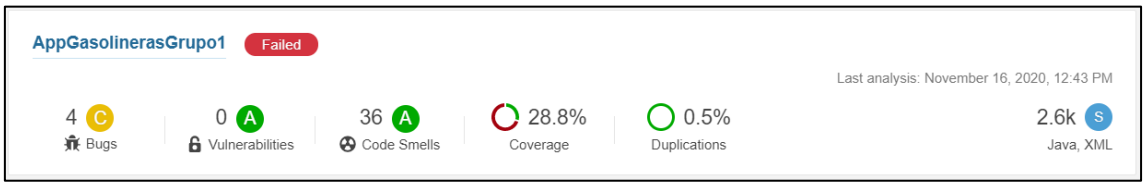
4. Comentarios

Sólo con arreglar el *bug* ya pasaríamos los criterios de calidad establecidos por la organización.

Aunque las horas de deuda técnica estaban dentro de los límites de la organización, se han añadido otros *issues* al plan de acción para ir bajando las horas de deuda técnica, de forma que haya un mayor margen de error en las próximas integraciones. Si se realizan todos los cambios enumerados en el plan de acción, la deuda técnica se reduciría en 1 hora y 4 minutos.

ANALISIS 16 DE NOVIEMBRE 2020 (Carol)

1. Resultados de Sonar



2. Análisis

El análisis no pasa los criterios de calidad de la organización, la calificación en confiabilidad es C y debería ser A. Los mayores problemas son:

1. 4 bugs presentes en diferentes clases: *TarjetaDescuento, TarjetaDescuentoPorLitro, TarjetaDescuentoPorcentaje* y *FiltroFavoritosActivity*.
2. *Code Smell:* el código presenta 36 *issues* de mantenibilidad (la mayoría de estos *issues* se encuentran en el *MainActivity* con 10 *issues* y en el *FiltroActivityUITest* con 6 *issues*.
3. Deuda técnica: la mayor parte de la deuda técnica se encuentra en el *MainActivity* con un total de 2h, en *ParsenJSONGasolineras* con 36 min y en el *DetailActivity* con 22 min.

3. Plan de acción

En este apartado se definen un plan de mejora de la calidad del código en el que se propongan una lista de acciones correctivas a realizar y se razone por qué se han optado por ellas.

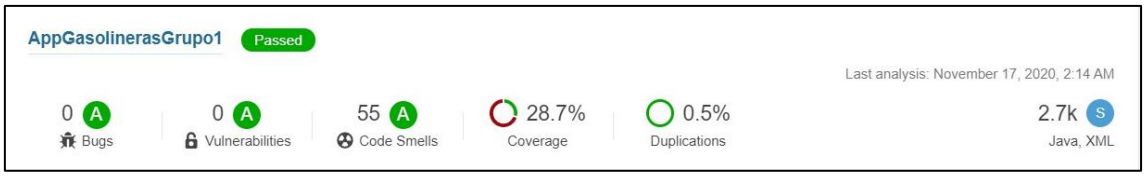
1. Corregir los 4 bug. Nos encontramos con 3 *bugs* que son del mismo tipo*: This class overrides "equals()" and should therefore also override "hashCode()"*, en las clases *TarjetaDescuento, TarjetaDescuentoPorLitro y TarjetaDescuentoPorcentaje*. El cuarto *bug* se encuentra en la clase *FiltroFavoritosActivity*. Cada bug supone un esfuerzo de 15 mins cada una. Si se solucionan todos lo *bugs* la deuda disminuye 1h.
2. Para seguir reduciendo la deuda optamos por resolver tres *issues* del tipo *major*. El primero se encuentra en la clase *ParsenJSONGasolineras*. Que disminuye 30 min la deuda técnica. La solución que nos sugiere sonar esta la siguiente: *Return an empty collection instead of null. Return Collections.emptyList()*. El segundo se encuentra en la clase *MainActivity* en el método *modificaComentario*. Esto supone un esfuerzo de 5 minutos. La solución que nos sugiere sonar es la siguiente: *Remove this unused method parameter "position"*. El tercero se encuentra en la clase *PresenterGasolinerasFavoritas* y sonar nos sugiere corregirlo de la siguiente forma: *Remove this unused "gasolineras" private field.* El esfuerzo que supone es de 5 min.
3. Por último, para reducir un poco más la deuda, resolveremos 2 *issues* del tipo *minor*. Eliminaremos dos import que no son utilizados. Estos *imports* se encuentran en las clases: *PresenterGasolinerasFavoritas* y *DetailActivity*. Supone un esfuerzo de 2 min cada uno.

4. Comentarios

Si se lleva a cabo el plan de acción, el sistema se quedaría sólo con algunos *issues* menores y de *info,* y reduciría la deuda técnica 1h y 44 min, quedando una mayor holgura para las siguientes integraciones del sprint.

<u>ANÁLISIS 17 NOVIEMBRE 2020</u> (Elena)

1. Resultados de Sonar



2. Análisis

Como podemos ver, el análisis pasa los criterios de calidad de la organización. Sin embargo, este pasaría con 55 issues, y con una deuda técnica rozando el límite establecido (4 horas).
En cuanto a mantenibilidad, como ya hemos dicho, hay una deuda técnica de 4 horas, que se concentra mayoritariamente en las clases Main Activity (2 horas) y PresenterGasolinerasFavoritas (1 hora y 20 mins).

Por otro lado, podemos ver que los issues se dividen en 28 mayores, 16 menores, y 11 de información, no habiendo ninguno bloqueante ni crítico.

Aunque la deuda técnica no sobrepasa los criterios establecidos por la organización, se encuentra rozando ese límite, así que se debería ejecutar cuanto antes el plan de acción aquí presente, para tener un cierto margen de error de cara al final del sprint.

3. Plan de acción

1. En primer lugar, para reducir el número de issues críticos se deben cambiar los assertTrue por assertEquals en el test FiltroFavoritosActivityUITest. Aunque SonarCloud no indica el esfuerzo estimado, este no debería ser mayor a 15 minutos.

2. Por otro lado, vamos a reducir la deuda técnica de la clase PresenterGasolinerasFavoritas, empezando por los code smells mayores, como son los siguientes:

    - Eliminar el ArrayList privado de gasolineras, ya que no se utiliza (5 mins effort).

    - Eliminar el System.out del método getGasolineraFavoritaPorId, ya que es de un mensaje de depuración que ya no es necesario (10 mins effort).

    - En el método eliminaGasolineraFavorita, eliminar tanto las asignaciones a la variable local gasolinera, como las comprobaciones del valor de esta, ya que son restos de una implementación anterior, y por tanto ya no son necesarios (25 mins effort). Así mismo, se debe eliminar de este mismo método otro System.out de depuración que ya no es necesario (10 mins effort).

    - En el método getGasolinerasFavoritas, se debe cambiar el System.out por un logger (10 mins effort).

3. Para seguir reduciendo el número de issues mayores, se deben cambiar los assertTrue de la clase PresenterGasolinerasFavoritasTest por assertEquals. Al igual que en el primer punto, SonarCloud no calcula el tiempo estimado, pero este no debería ser mayor de 20 minutos.

4. Para acabar con los issues mayores, se pueden cambiar también los assertTrue de la clase PresenterTarjetaDescuentoTest por assertEquals. Al igual que en el punto anterior, esto no debería llevar más de 10 minutos.

5. Para seguir reduciendo la deuda técnica de la clase PresenterFavoritas, se deben eliminar los imports que no se utilizan, y algunos que se encuentran repetidos (20 mins effort).

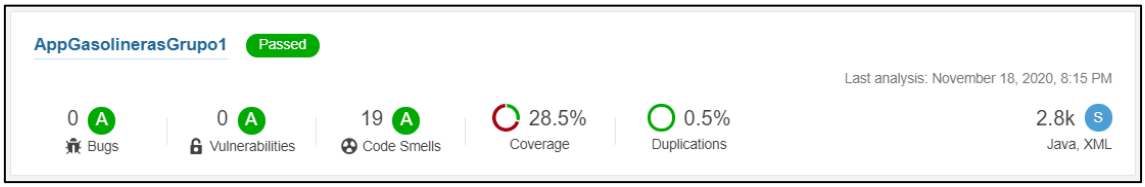
6. Por último, en la clase DetailActivity se pueden corregir 2 issues menores: eliminar los imports no utilizados (2 mins effort), y cambiar el nombre de la variable local txt_confirmacion para evitar utilizar caracteres como "_". Sería recomendable cambiarlo por un nombre parecido, como puede ser txtConfirmacion (2 mins effort).

4. Comentarios

Si se lleva a cabo el plan de acción, el sistema se quedaría sólo con algunos issues menores y de info, y reduciría considerablemente la deuda técnica, quedando una mayor holgura para las siguientes integraciones del sprint.

ANÁLISIS 19 NOVIEMBRE 2020 (Carol)

1. Resultados de sonar



2. Análisis

El análisis pasa los criterios de calidad de la organización. La calificación en confiabilidad (*reliability*) es A. Aunque se pasen los criterios de calidad, se va a realizar un plan de acción para reducir la deuda técnica y así poder contar con un margen mayor para el siguiente sprint.
Vemos que el código presenta 19 *issues* de mantenibilidad (*code smells*) de los cuales 4 *issues* son del tipo *major*, 6 *issues* son del tipo *minor* y 9 son del tipo *info.*

3. Plan de acción

En este apartado se define un plan de mejora de la calidad del código en el que se propone una lista de acciones correctivas a realizar y se razona por qué se ha optado por ellas.

1. En esta ocasión el plan de acción se centrará en la severidad de los *issues* y se elegirá filtrar por el tipo *major.* Estos *issues* se encuentran distribuidos en 4 clases distintas: *RemoteFetch, DetailActivity, FiltroFavoritosActivity* y *PresenterFiltroMarcasITest*. Corregir estos errores disminuye la deuda técnica 45 min.
2. En este apartado se aplica un filtro de *rule,* la segunda regla con más *issues* es la de *The diamond operator("<>") should be used*. Son *issues* del tipo info. Dos de ellos se encuentran en la clase *PresenterTarjetaDescuento* y el otro en *PresenterGasolineras*. Reduce la deuda 3 min.
3. En este apartado filtramos por S*everity* y por *File.* Encontramos dos *issues* del tipo *minor* en la clase *PresenterGasolineras*. Reduce la deuda 7 min.

El objetivo es reducir lo que se pueda dependiendo de nuestras necesidades o situación. No es necesario resolver todas.

Resolución:

1. En la clase *RemoteFetch* se produce un *issue* del tipo *major*. Para resolver este problema, sonar nos recomienda añadir un constructor privado. En la clase *DetailActivity* y *FiltroFavoritosActivity* se produce una duplicación del código que debe ser eliminada. Y por último en la clase *PresenterFiltroMarcasITest* sonar nos recomienda usar *assertEquals* en lugar de *assertTrue.*
2. En este caso sonar nos recomienda lo siguiente: *Replace the type specification in this constructor call with the diamond operator ("<>").* Encontramos estos *issues* del tipo info en las siguientes clases: dos en *PresenterTarjetaDescuento* y uno en *PresenterGasolineras*.
3. Estos *issues* se encuentran en la clase *PresenterGasolineras*. En primer lugar, en el método *getGasolinerasPorIdess,* sonar nos recomienda sustituir: lista.size == 0 por *isEmpty()*. En segundo lugar, sonar nos dice que debemos eliminar la declaración del *thown NullPointerException* de la clase *filtrarGasolinerasTipoCombustible*.

4. Comentarios

Si se lleva a cabo el plan de acción, el sistema se quedaría sólo con algunos *issues* menores y de *info*, y reduciría la deuda técnica 55min.