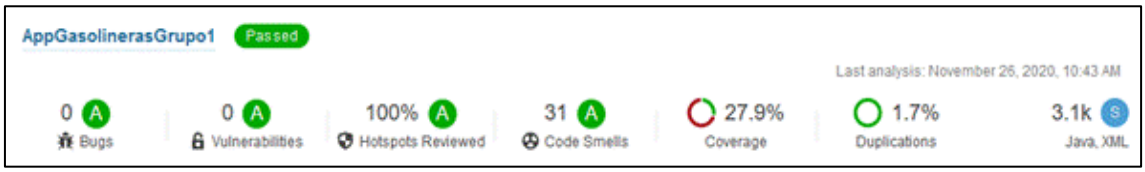
Análisis de Calidad del Producto - Sprint 3

Autores: Luis Cruz y Jaime López-Agudo Higuera

ANÁLISIS 26 NOVIEMBRE 2020 (Luis Cruz)

1. Resultados de Sonar



2. Análisis

Tras la primera integración de la rama “Aplicar tarjetas de descuento en favoritos” aparecieron 31 nuevos code smells y un bloque de código duplicado (38 líneas) que aumentaron la deuda técnica del proyecto en 2 horas y 34 minutos.

A pesar de lo descrito anteriormente, y como se puede ver en la imagen, el proyecto todavía cumple con los estándares de calidad definidos en sonar. Aunque es posible que, si no se solucionan estos problemas de calidad, esto pueda cambiar en la siguiente integración.

La mayor cantidad de deuda técnica se sitúa en las clases “FiltroFavoritosActivity” que contribuye 47 minutos y “DetailActivity” que añade unos 44 minutos. Tanto “FiltroFavoritosActivity” como “DetailActivity” aportan tanto tiempo debido al bloque de código duplicado. Asimismo, “PresenterTarjetasDescuento” añade 27 minutos, “MainActivity” 15 minutos e “InfoActivity” 10 minutos.

3. Plan de acción

1) Extraer el bloque de código duplicado situado en “FiltroFavoritosActivity” y “DetailActivity, si se pudiera, a una clase externa. De esta manera se reduciría el porcentaje de código duplicado a 0.5%.
2) Arreglar el error menor de “PresenterTarjetasDescuento”(línea 97) cambiando tanto el tipo de retorno del método como el tipo del parámetro que recibe por un List, ahorrándonos así 20 minutos de deuda técnica.
3) Añadir también comentarios anidados de los métodos no implementados en el bloque de código duplicado (BeforeTextChanged, AfterTextChanged). De esta manera se reduciría otros 20 minutos la deuda técnica.
4) Eliminar la línea 19 (System.out.println) de “InfoActivity” y la 471 de “MainActivity”, ahorrando así otros 20 minutos.
5) Eliminar las líneas de código comentadas en “MainActivity” (472,473) reduciría otros 5 minutos la deuda técnica.
6) Por último, eliminar un cast innecesario en “FiltroFavoritosActivity” en la línea 133 para así reducir la deuda otros 5 minutos.

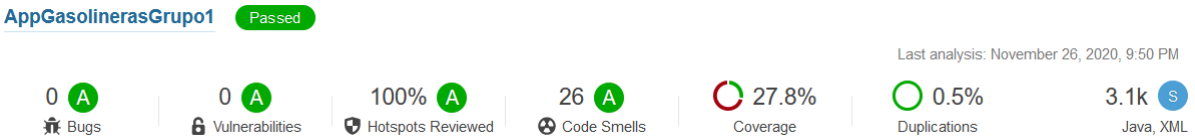
4. Comentarios

Si todas las medidas detalladas en la sección 3 se llevaran a cabo, se reduciría la deuda técnica en unos 70 minutos, además de eliminar un bloque de código duplicado. La prioridad a la hora de solucionar estos problemas de calidad deberían ser los puntos 1 y 3.

Además de los code smells detallados en el plan de acción, hay unos pocos más cuya aportación, relativamente pequeña, a la deuda técnica e importancia han contribuido a que no se incorporaran a este informe.

ANÁLISIS 27 NOVIEMBRE 2020 (Luis Cruz)

1. Resultados de Sonar



2. Análisis

Tras la primera integración de la rama “Mostrar aviso lista favoritos vacía” aparecieron 20 nuevos code smell, que aportaron 1 hora y 40 minutos de deuda técnica. La mayor parte de esta deuda vino de un bloque de código duplicado, situado en “FiltrosFavoritosActivity” y “DetailActivity”. Estas dos clases son, de nuevo, las que más deuda técnica aportan (1 hora). A ellas les sigue la clase “InfoActivity”.

A pesar de estos problemas de calidad, el proyecto sigue cumpliendo los estándares de calidad de sonar. De hecho, la calidad del proyecto ha mejorado, reduciéndose los code smells en 5 desde el último informe de calidad.

Esta trayectoria es esperanzadora, ya que hace más difícil que la calidad decaiga drásticamente entre integraciones, aunque hay problemas en el código que se vienen arrastrando de sprints anteriores y que el equipo ha considerado muy costosos de arreglar durante un sprint normal.

3. Plan de acción

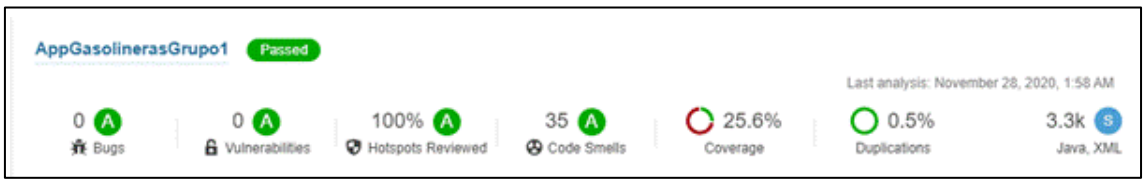
1) Quitar los castings innecesarios en “FiltrosFavoritosActivity” (líneas 297-307). Esto reduciría la deuda técnica en 20 minutos (4 castings de 5 mins)
2) Eliminar la línea 19 (System.out.println) de “InfoActivity”, reduciendo así otros 10 minutos de deuda técnica.
3) Crear un constructor vacío privado en “DialogoComentarioGasolinerasFavoritas” ahorraría otros 5 minutos de esfuerzo.
4) Quitar el cast innecesario a ArrayList en línea 489 de “MainActivity” reduciendo otros 5 minutos la deuda.

4. Comentarios

Con las medidas detalladas en el plan de acción se reduciría la deuda técnica en 40 minutos. Cabe destacar que no se ha detallado en el plan de acción una medida para el bloque de código duplicado. Esto se debe a que dicho bloque de código se ha estimado muy difícil y costoso de extraer, ya que partes de ese bloque dependen de parámetros como el contexto de la interfaz en la que se encuentra.

ANÁLISIS 28 NOVIEMBRE 2020 (Jaime López-Agudo Higuera)

1. Resultados de Sonar



2. Análisis

Tras la primera integración de la rama filtrarGasolineraPorPrecioMaximo aparecieron 14 nuevos code smells, así como un aumento de la deuda técnica de 56 minutos desde la última integración de calidad sobre el proyecto.

La mayor parte de la deuda técnica la volvemos a encontrar en la clase MainActivity.java, con una deuda de 2h 1min, seguida de lejos por las clases PresenterGasolineras.java (22 min), DetailActivity.java (20 min) y FiltroFavoritosActivity.java (20 min).

A pesar de estos problemas de calidad, el proyecto sigue cumpliendo con los estándares de calidad de sonar fijados al comienzo del proyecto integrado, aunque la progresión en cuanto a code smells no es buena, ya que estos han aumentado en 9 desde la última revisión de calidad.

3. Plan de acción
    1) Sustituir en la clase InfoActivity.java, en la línea 49, el System.out por un logger o un system.err, lo que reduciría le deuda en 10 minutos.
    2) Quitar el parámetro no usado en el método caracterValidos, en la clase MainActivity.java, línea 333, lo cual reduciría la deuda técnica en 5 minutos.
    3) Cambiar el uso del switch en la clase PresenterGasolineras, en la línea 164 por un bloque if, esto reduciría la deuda técnica en 10 minutos, ya que se solucionarían los code smells de sustituir el switch por un if y de añadir un default case al switch.
    4) Marcar con la anotación @Deprecated los métodos onPreExecute() y onPostExecute(), dentro de la clase MainActivity.java, lo cual reduciría le deuda técnica en 30 minutos (15 cada uno).
    5) Eliminar imports innecesarios en las siguientes clases:
    PresenterGasolinerasFavoritas.java (línea 5), 2minutos.
    PresenterTarjetaDescuento.java (línea 9), 2 minutos.
    MainActivity.java (líneas 34 y 35), 4 minutos.

6) Eliminar de la variable local gasolinerasFiltradas en la línea 278 de la clase MainActivity.java, lo que reducirá la deuda técnica en 5 minutos.
7) Solucionar 3 code smells críticos en la clase MainActivity.java en las líneas 281, 287 y 291, cambiando el método replaceAll por replace en los 3 casos, reduciendo la deuda técnica en 6 minutos.
8) Cambiar el tipo de retorno del método fitrarGasolinerasPorPrecioMaximo dentro de la clase PresenterGasolineras.java para que pase de ser un ArrayList a una List. Esto reducirá la deuda técnica en 10 minutos.

4. Comentarios

Con el cumplimiento de las medidas detalladas del plan de acción, se reducirá la deuda técnica de la aplicación en 1h y 24 minutos, lo cual mejoraría enormemente la calidad del código y evitaría futuros problemas durante el desarrollo de la aplicación.

ANÁLISIS 29 NOVIEMBRE 2020 (Jaime López-Agudo Higuera)

1. Resultados de Sonar

2. Análisis

Tras la última integración de la rama ScrollCampoComentario, el sistema ha acabado con una deuda técnica de 2h 26 minutos y un total de 25 code smells, estas métricas siguen estando dentro de los estándares de calidad definidos al comienzo del proyecto integrado, pero se puede mejorar un poco más la calidad de la aplicación aplicando un nuevo análisis junto a sus medidas correspondientes.

Desde el último análisis de calidad observamos que los code smells se han reducido en 10, lo cual nos indica que continuamos con una progresión óptima en este apartado en el proyecto.

En cuanto a la organización de los code smells y la deuda técnica, seguimos viendo como la mayoría de estos se encuentran sobre la clase MainActivity, la cual contiene una deuda técnica de 1h51 minutos y 12 code smells, los cuales son debidos principalmente a la necesidad de refactorizar la clase encargada de la AsyncTask (LavaFlow), los demás code smells se encuentran en las clases DetailActivity (7 minutos y 2 code smells) y PresenterTarjetaDescuento (7 minutos y 4 code smells).

3. Plan de acción
    1) Renombrar el campo INSTANCE dentro de la clase AppDataBase, AppDataBaseTest y PresenterTarjetaDescuento para que siga el formato de java (líneas 16, 16 y 22 respectivamente), esto tendría una mejora de 6 minutos de deuda técnica.
    2) Hacer uso del método isEmpty() proporcionado por java, en vez de usar lista.size()==0 para determinar el estado de la lista en la clase PresenterGasolineras y MainActivity (líneas 52 y 291 respectivamente), esto mejoraría en 4 minutos la deuda técnica.

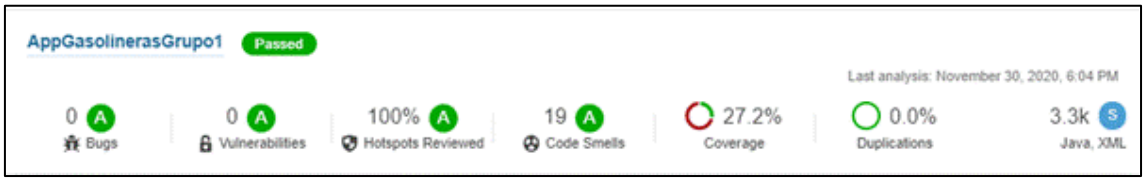
3) Eliminar imports innecesarios en las clases DetailActivity y PresenterTarjetaDescuento (líneas 23 y 9), lo que reducirá la deuda técnica en 4 minutos.
4) Reordenar los modificadores para que sigan la especificación del lenguaje de java (de private synchronized static a private static synchronized y de final static int a static final int) en las clases PresenterTarjetasDescuento y DialogoComentarioGasolineraFavorita (líneas 34 y 13 respectivamente).
Esto reduciría la deuda técnica en 4 minutos.
5) Eliminar el bloque comentado de código en la línea 99 de la clase DetailActivity, lo que tendría una reducción de deuda técnica de 5 minutos.
6) Eliminar casteos innecesarios a ArrayList en las líneas 508 y 706 dentro de la clase MainActivity, lo que reducirá la deuda en 10 minutos.

4. Comentarios

Con el cumplimiento de las medidas presentadas en el plan de acción, la deuda técnica del proyecto se reducirá en 43 minutos, colocando la deuda técnica total del proyecto en una buena posición, por debajo de las 2 horas de deuda, lo cual se alejaría mucho del límite de deuda técnica de 4h y 10 minutos, fijado al comienzo del proyecto.

ANÁLISIS 30 NOVIEMBRE 2020 (Jaime López-Agudo Higuera)

1. Resultados de Sonar



2. Análisis

Tras la última integración de la rama AplicarTarjetasDeDescuentoAFavoritosAFavoritos, el proyecto acaba con un total de 19 code smells, lo que es una mejora significativa frente a la última integración de calidad que tuvo 25 code smells, en cuanto a la deuda técnica, se encuentra actualmente en 2h 3 minutos, por lo que, aunque estas métricas siguen estando dentro de los estándares fijados para el proyecto, se pueden mejorar aun la calidad del software.

En referencia a la organización de los code smells y la deuda técnica, seguimos viendo como ambos se encuentran en su gran mayoría sobre la clase MainActivity (12 code smells y 1h51min de deuda), seguida muy por detrás por las clases PresenterTarjetaDescuento (4 code smells y 7min de deuda técnica), DialogoComentarioGasolineraFavorita (1 code smell y 2 minutos de deuda) y ParserJSONGasolineras (1 code smell y 1 minuto de deuda).

Como podemos observar, la progresión del proyecto en cuanto a calidad ha sido óptima, pero aún se puede mejorar un poco más, sin embargo, dado que la mayor parte de la deuda técnica se concentra en la clase interna CargaDatosGasolineraTask dentro de MainActivity y para solucionar sus conflictos se necesitaría realizar una refactorización completa de la misma, basándose en una investigación

previa de la librería concurrent de Android, se ha decidido no optar por esa solución y tratar de minimizar la deuda técnica externa a la clase CargaDatosGasolineraTask.

3. Plan de acción
    1) Marcar con la anotación @Deprecated los métodos onPreExecute() y onPostExecute(), dentro de la clase MainActivity.java, lo cual reduciría le deuda técnica en 30 minutos (15 cada uno) *.
    2) Eliminar casteos innecesarios a ArrayList en las líneas 512, 637 y 710 dentro de la clase MainActivity, lo que reducirá la deuda en 15 minutos*.
    3) Quitar import no usado de java.util.Collections de la línea 9 de PresenterTarjetaDescuento, lo que implica una mejora de 2 minutos de deuda técnica.
    4) Renombrar la variable INSTANCE de la clase PresenterTarjetaDescuento a instance, para seguir el formato de los argumentos de java (línea 22, 2 minutos de deuda).
    5) Reordenar los modificadores del método createInstance() (línea 34 de la clase PresenterTarjetaDescuento, 2 minutos de deuda) *.

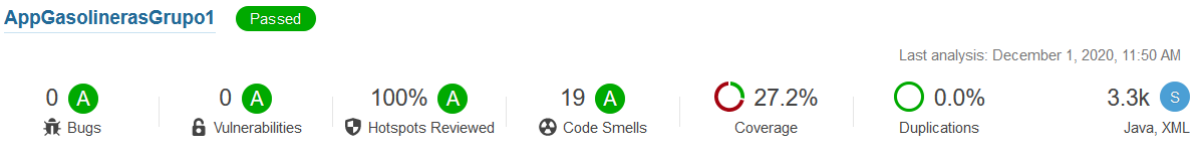
*Algunas medidas están repetidas debido a que no se aplicaron en resoluciones de calidad anteriores, por lo que se han vuelto a presentar en este plan de acción.

4. Comentarios

Con la aplicación del plan de acción presentado, la deuda de la aplicación se reducirá en 51 minutos, lo que dejaría al proyecto con una deuda técnica más cercana a la hora que a las dos horas, siendo la principal culpable de la deuda restante la clase CargaDatosGasolineraTask, cuya refactorización se dejará para la última verificación de calidad del tercer sprint.

ANÁLISIS 1 DICIEMBRE 2020 (Luis Cruz)

1. Resultados de Sonar



2. Análisis

Tras las integraciones de las ramas “Anhadir aviso de favoritos favoritos” y “Aplicar tarjetas de descuento a favoritos” la calidad del proyecto no varió, no se detectaron nuevos code smells. Esto es un buen indicador de que el grupo está poniendo especial atención a la calidad del código desarrollado.

Debido a la falta de nuevos code smells, el análisis de calidad se realizará sobre todo el proyecto y no (como se venía haciendo en integraciones anteriores) sobre la última integración.

El proyecto sufre en este momento de 19 code smells, estando la mayoría situados en “MainActivity” (y aportando el 90% de la deuda técnica). Después de ella, “PresenterTarjetasDescuento” aporta la mayor cantidad de deuda (a pesar de ser tan solo 7 minutos).

Los code smells de “Main Activity” (relacionados con el uso de métodos y clases *deprecated*) se vienen arrastrando desde el inicio del proyecto, y hasta ahora se habían ignorado al haberlos considerado difíciles de solucionar y de no mucha prioridad.

3. Plan de acción

    a. Hacer que la clase “CargaGasolinerasTask” herede de thread en vez de “AsyncTask”, conservando su método “onPostExecute” y trasladando el método “doInBackground” al método “run”. Esto reduciría la deuda técnica en 1 hora y 30 minutos.
    b. Cambiar la comparación de la línea 292 de “MainActivity” por currentList.isEmpty. Esto ahorraría otros 2 minutos de deuda técnica.
    c. Eliminar el if de la línea 342 de “MainActivity” y trasladar su condición al return, de este modo: return exp1||exp2||exp3. De esta manera, se bajará la deuda técnica en otros 2 minutos.
    d. Mover la declaración de la variable “contador” de la línea 356 de “MainActivity” a una nueva línea, ahorrando así otros 2 minutos.
    e. Eliminar los cast innecesarios a ArrayList de las líneas 513, 638 y 711 de “MainActivity” ayudaría a bajar la deuda técnica 15 minutos más.
4. Comentarios

Si todas las medidas descritas en este plan de acción se llevaran a cabo, la deuda técnica del proyecto se reduciría al mínimo histórico de 12 minutos. Sin embargo, puede que la medida descrita en “1)” sea demasiado difícil de aplicar. En ese caso, la deuda se reduciría a (todavía impresionante) 1 hora y 42 minutos.

Cabe destacar que es posible, debido al problema ya explicado de falta de aparición de code smells, que haya medidas que aparezcan en informes de calidad anteriores. Se ha intentado evitar, pero esto unido a la existencia de code smells que no se han solucionado entre integraciones puede haber contribuido a que existan medidas del plan de acción similares a planes de acción anteriores.