Análisis de Calidad: Historia de usuario "Ver estimación de los próximos autobuses"

Resumen

A continuación, se realizará el análisis de calidad del código que implementa la historia de usuario "*US-241917-VerEstimaciones*".

1. Introducción

El documento contiene el análisis de calidad realizado a la implementación de la historia de usuario "*US-241917-VerEstimaciones*". En este informe se detallarán los análisis realizados de los resultados obtenidos al subir el código al servidor SonarCloud, las incidencias encontradas y sus respectivas soluciones para obtener la calidad exigida.

2. Análisis inicial

Al observar los resultados del análisis inicial se obtiene una clasificación A en los tres apartados, reliability (bugs), maintainability (code smells) y security (vulnerabilities).

A pesar de obtener la máxima calificación en los 3 apartados, aún se pueden hacer mejoras en el código para aumentar su calidad. Esto se debe al apartado de Maintainability donde se obtienen 87 nuevos code smells (118 en total) y se estima que se tardarían dos días en corregirlos.

En relación a los demás apartados, se considera que se ha alcanzado el quality gate establecido ya que en el apartado de Reliability no se encuentran bugs y en el apartado de Security no hay vulnerabilidades.

3. Solución a los errores encontrados

En esta sección se basa en resolver los errores relativos a los code smells obtenidos en el análisis inicial.

El orden de corrección establecido está ordenado según la gravedad de los errores para la calidad del código, por ello se comienza por errores críticos, después los errores mayores y finalmente los errores menores.

- Los errores críticos y sus respectivas correcciones, ordenados por clases:
    - **ParserJson**: se utilizan las constantes previamente establecidas para "UTF-8" y "resources" para evitar repeticiones en el código.
    - **ListLineasAdapter**: se detecta que se debe eliminar la duplicidad del "if infernal" generado a la hora de colorear las diferentes líneas de autobuses. Se decide dejar la corrección de este error para un futuro al no detectar una forma más óptima de escribir dicho bloque de código.
    - **ListEstimacionesAdapter**: se detecta el mismo error que en el caso anterior por lo que se decide no corregirlo.

Los errores críticos generados con esta historia de usuario ya integrada en el proyecto quedarían solucionados.

- A continuación, realizaremos el mismo proceso con los errores mayores:

  - **CommonUtilsTest:** los 15 primeros errores mayores que aparecen están relacionados con el método assertEquals(expected value, actual value) de la clase Assert. En el código algunos test tienen el orden de los parámetros cambiado y aparece como assertEquals(actual value, expected value). Por ello, se cambia el orden de estos en varios test de la clase, desde U8A hasta U8M.

  - **ListLineasPresenter:** se indica que el atributo de tipo Context nunca llega a ser utilizado más allá de su asignación. Se elimina el atributo context y se modifica el constructor de la clase ListLineasPresenter, retirando el parámetro context. Además se modifican las clases que creaban instancias de ListLineasPresenter, ListLineasPresenterTest y LineasTodasFragment, quitándoles también el parámetro de tipo Context.

  - **ListEstimacionesAdapter**: se eliminan varios imports de la clase ya que no son utilizados.

  Los errores mayores generados con esta historia de usuario ya integrada en el proyecto quedarían solucionados.

- A continuación, realizaremos el mismo proceso con los errores menores:

  - **Estimacion**: el campo "MENSAJE_LLEGADA" no corresponde con la nomenclatura correspondiente, por lo que se añade el modificador final ya que debería serlo y se corrige el error.
    Otro error menor de esta clase es que no está sobrescrito el método equals(Object obj) para cumplir con el convenio del método "compareTo(T o)". Para corregirlo se añade el método equals(Object obj) a la clase Estimacion en el que se comparan los atributos correspondientes de la clase.

  - **Linea**: En esta clase aparece el mismo error con el método equals(Object obj) que en la clase Estimacion, así que se toman las mismas medidas de corrección.

  - **DetallesParadaPresenter**: se elimina el import 'android.widget.BaseAdapter' ya que no era utilizado.

  - **ListEstimacionesAdapter**: los 14 próximos errores menores se encuentran en esta clase y son debidos a varios imports no utilizados. Se borran los imports no utilizados corrigiendo los errores.

  - **MainActivity**: se requiere eliminar el campo "mBottomBar" y declararlo como variable local en los métodos respectivos pero se decide no quitarlo ya que corresponde a la barra de debajo de navegación y si se establece como variable local no se vería.

  Los errores menores generados con esta historia de usuario ya integrada en el proyecto quedarían solucionados.

- En relación con los code smells marcados como “info”, se decide resolver uno ya que el resto se consideran poco relativos.

    - ListParadasLineaPresenter: se elimina un TODO “equivocado” en el comentario de un método.

4. Segundo análisis

Después de tratar de corregir los errores citados en el anterior apartado. El resultado del segundo análisis es B en el apartado de reliability ya que se encuentran 2 bugs, y dos A en los apartados de maintainability (code smells) y security (vulnerabilities).

A pesar de seguir obteniendo la máxima calificación en el apartado de maintainability y de haber reducido el número de code smells de 118 a 83, se ha detectado que aún existen code smells que pueden resolverse. Esto se debe a que al corregir errores en el anterior apartado, han surgido otros que se deben solucionar.

5. Solución a los errores encontrados

- Primero se corrigen los dos **bugs** que se han obtenido debido a la corrección de dos de los errores menores en los que era necesario sobrescribir el método equals(Object obj). Ya que si sobrescribe el método equals también debería sobrescribir el método hashCode(), por lo que añadimos el método hashCode() en las clases **Estimacion** y **Linea**.

- Los 2 errores de carácter crítico que se obtienen son los mismos que se decidió no corregir en el primer análisis, relacionados con el color de las líneas en **ListLineasAdapter** y **ListEstimacionesAdapter**.

  Los errores críticos generados con esta historia de usuario ya integrada en el proyecto quedarían solucionados.

- Los 3 errores mayores que quedan sin resolver también se obtienen en el primer análisis y se decide no corregirlos. Esto se debe a que dos de ellos son por duplicaciones de código también relativas a colorear las líneas de autobuses, clases **ListLineasAdapter** y **ListEstimacionesAdapter**. El otro error se debe a una duplicación de código en la clase **ParserJson** que se decide no corregir por motivos de claridad del código y modularidad.

- A continuación, realizaremos el mismo proceso con los errores menores:

    - **Estimacion:** se obtiene que la clase sobrescribe el método equals() y por tanto debe sobrescribir también hashCode(), no se modifica nada ya que se ha corregido anteriormente en el apartado de bugs.

    - **Linea:** se obtiene el mismo error que en la clase Estimacion, también corregido en el apartado de bugs.

- **ListLineasPresenter**: al modificar esta clase en los errores mayores del apartado 3 quitando el atributo context, se olvida quitar el import 'android.content.Context'. Se elimina el import corrigiendo el error.

- **MainActivity:** seguimos obteniendo el error en el campo “mBottomBar” ya que se decidió no corregirlo.

Los errores menores generados con esta historia de usuario ya integrada en el proyecto quedarían solucionados.

6. Análisis final

Después de tratar de corregir los errores citados en los anteriores apartados. Nos encontramos que la calificación del código es A en las medidas Reliability, Security, Maintainability, respectivamente. Se ha reducido el número de code smells de 118 a 81, quedando algunos errores que se decide no eliminar por las razones explicadas anteriormente. La deuda técnica del código aun habiendo disminuido significativamente los code smells sigue siendo de dos días.

7. Conclusión

En este entregable, se ha expuesto el análisis de calidad de la historia de usuario “*US-241917-VerEstimaciones*” detallando los cambios que se han ido realizando en la implementación para mejorar la calidad del código.

Finalmente, se llega a la conclusión de que se ha mejorado la calidad del código lo máximo posible, ya que todavía existen errores que por una u otra razón no se pueden resolver o son poco relevantes.