Informe de calidad de la aplicación TUS Santander (Recarga automática)

Calidad y Auditoría

Guillermo Argumosa Arroyo

Análisis Inicial

Se realiza el primer análisis de Sonar sobre el código implementado para la ejecución de la recarga automática de la aplicación.

Se obtiene una puntuación AAA (Fiabilidad, Seguridad, Mantenibilidad) con 0 bugs, pero se tiene también una deuda técnica de 3 días por "code smells" y hay un 18,7% de código duplicado.

Se procede entonces a reducir la tasa de duplicaciones al 0% por considerarse lo más prioritario.

Cambios Iniciales

La primera duplicación que se detecta es el método "showProgress" presente en las clases "MainActivity" y "LineasActivity".

Se ha tratado de llevar este método a una clase común de funciones para ser llamada desde cualquiera de las actividades, pero dada la gran cantidad de problemas relacionados con la ejecución en hilos separados que ha generado esta solución, se ha optado por una alternativa.

Como es necesario que ambos métodos se llamen de la misma forma pues en ocasiones se invoca sin conocer el tipo de actividad a la que se llama, el nombre se conserva. Para que haya una diferenciación, se procede a renombrar variables.

Estos cambios introducidos reducen la tasa de duplicación a 17,4% y las dos clases analizadas dejan de tener código duplicado entre sí.

Análisis 2

Haciendo un nuevo análisis con Sonar se detectan duplicaciones entre los métodos "getParada", "getAllParada", "getAllParadasFavoritos" y "getParadasByLinea" de la clase "DatabaseHelper".

Cambios 2

Se añade el método "setDatosParada" para reducir los bloques repetidos.

Esto reduce la tasa de duplicación a 16,1%.

Sigue habiendo bloques duplicados en la clase "DataBaseHelper" entre "getAllParada", "getAllParadasFavoritos" y "getParadasByLinea", por lo que añadiendo un método "setListaParadas" que llama al previamente diseñado se reduce la tasa de duplicación a 14,6% y la clase analizada deja de tener código duplicado.

Análisis 3

En el nuevo análisis ejecutado se detectan duplicaciones entre las clases "GetDataServicio" de los paquetes "Main" y "Lineas" del "Presenter" en todos sus métodos.

Cambios 3

Para solucionar estas últimas duplicaciones, se renombran los atributos "refresh" de las dos clases por "refreshLineas" y "refreshMenu", lo cual reduce la tasa de duplicación a 13,7% y elimina código repetido entre las dos clases analizadas.

Análisis 4

El último gran fallo de duplicación de código que se detecta es entre las clases "RecargaBaseDatosMenu", "RecargaBaseDatosParadas" y "RecargaBaseDatosLineas", cuyo código es en casi totalidad el mismo.

Cambios 4

Como ya se han realizado correctamente renombrados de variables y ha servido para mejorar la calidad del código (se distingue mejor qué variables pertenecen a qué clase y quita código duplicado) se opta de nuevo por esta misma opción.

Se cambian los nombres de variables como "activity" o "presenter" a "activityLineas", "activityMenu", "presenterParadas" … Según la clase en la que se encuentren.

Estos últimos cambios reducen la tasa de código duplicado a 6,5%.

No se ha podido reducir a 0% la duplicación de momento ya que en las clases de recarga de base de datos hay varios "switch" con muchos casos que contienen una sola línea muy semejante en todos ellos (detectada como duplicada por Sonar) a las que sustituir por un método no resolvería nada (una línea por otra) ni se pueden eliminar porque cada línea hace una acción diferente.

Análisis Final (Code Smells)

Una vez solventados los problemas de duplicación de código, se pasa a resolver "code smells", pequeños detalles mejorables en el código que al acumularse pueden generar una gran deuda técnica.

En el estado actual, Sonar detecta 166 "code smells" que implican una deuda técnica de 2 días.

Cambios Finales

En la clase "DatabaseHelper" se detecta el uso de literales en el código que podrían haber sido asignados a una variable. Para evitarlo, se crean nuevas variables como INTEGER1 en lugar de "INTEGER," y se introducen en el código.

Este borrado de literales en el código reduce los "code smells" a 160.

En esta misma clase, hay varias variables que se declaran, se asignan valor y justo después se retornan. Se puede juntar todo en una sola línea de código para que no sea una deuda técnica extra (ejemplo: "long línea_id = db.insert…"; siguiente línea: "return línea_id;" se sustituye por "return db.insert…").

Esta reducción de líneas de código reduce los "code smells" a 154.

Con estas modificaciones, el nivel de calidad sería adecuado para proseguir con el desarrollo de los tickets de la aplicación en el sprint 2.