



PS17 – Informe de calidad asociado a la historia de usuario 242062 – Ordenar paradas alfabéticamente.

Resumen

El presente documento recoge el análisis de calidad asociado a la historia de usuario 242062 (Ordenar paradas alfabéticamente), así como una breve descripción del proceso llevado a cabo para generar el reporte final.

Document	PS17/00/2017-QR002-US242062-
ID:	OrdenarParadasAlfabéticamente
Departamento:	Calidad y Auditoría, dentro del proyecto integrado
Tipo:	Análisis de calidad
Privacidad:	CONFIDENCIAL
Estado:	ENTREGABLE
Versión:	1.0.0
Fecha:	26/10/2017
Autores:	Martínez Vila, Javier
Revisores:	

HISTORIAL DE CAMBIOS

Versión	Fecha	Cambio	Responsable
1.0.0	01/11/2017	Creación del documento.	Martínez Vila, Javier

1. Introducción

El presente documento contiene el proceso de análisis de calidad llevado cabo para la historia de usuario #242062 – Ordenar paradas alfabéticamente.

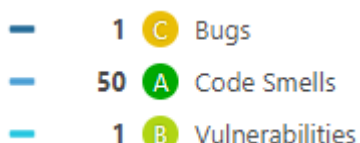
2. Desarrollo del proceso de análisis de calidad

Para efectuar el análisis de calidad de la historia de usuario previamente citada y realizar las correcciones correspondientes, he seguido el proceso descrito a continuación.

En primer lugar, procedí a subir el código inicial al servidor *SonarCloud*. A pesar de encontrar algunos problemas al realizar la subida (concretamente, problemas asociados a la lectura de ficheros con codificación no estándar), finalmente fue posible efectuarla sin inconvenientes que revistiesen gravedad.

El análisis inicial del código arrojó los siguientes resultados:

- **Deuda técnica:** 6h
- **Medidas, calificaciones e *issues*:**



— 1 C Bugs
— 50 A Code Smells
— 1 B Vulnerabilities

Respecto a la severidad de los *issues* encontrados en el primer análisis, la distribución fue la siguiente:

- 1 *critical*.
- 13 *major*.
- 37 *minor*.
- 1 *info*.

Una vez analizados los resultados obtenidos, procedí a realizar las correcciones que estimé más oportunas para mejorar la calidad del código desarrollado.

El plan de acción que seguí fue el de dar prioridad en primer lugar al *bug* y la *vulnerability* encontrados, en segundo lugar, a los *code smells* de mayor severidad, y, en último lugar, los *code smells* de severidad menor / informativos. Asimismo, tuve también en cuenta el coste que suponía arreglar cada uno de ellos, ya que en alguno de los casos un cambio simple en una única línea de código suponía la corrección de dicho *issue*.

Para agilizar el proceso, utilicé el plugin *SonarLint* para *Android Studio*, de manera tal que basándome en las anotaciones que dicho plugin realizaba podía encontrar los *issues* más fácilmente.

A continuación, adjunto un listado con los errores que, a mi juicio, resultaban más relevantes y, por tanto, han sido tenidos en cuenta de manera prioritaria al ejecutar acciones correctivas.

Errores tratados

Bugs

- En la clase *Parada.java*, existía un condicional que no contenía instrucciones de código útil en su interior. Dado que la función de dicho condicional era más bien explicativa, he sustituido dicho trozo de código por un comentario explicativo.

Vulnerabilities

- En la clase *ListParadasPresenter.java*, se empleaba *e.printStackTrace* en lugar de emplear el *Logger*. Con un simple cambio, se ha resuelto el *issue*.

Code Smells

Críticos

- En la clase *MainActivity.java*, el *switch* no recogía el caso por defecto. Este *issue* se ha solventado introduciendo un *default* en el bloque de código.

Severidad mayor

- En la clase *ParserJSON.java*, no existía un constructor por defecto. Se ha solventado añadiéndolo.
- En la clase *ListParadasPresenter.java*, no constaba la anotación *@Override* en la signature del método *onPostExecute*. Se ha solventado añadiendo dicha anotación.
- En varias de las clases aparecían comentarios que contenían código en su interior. Dichos comentarios se han eliminado, habiéndose corregido todos los *issues* de ese tipo siguiendo este esquema de actuación.
- En la clase *MainActivity.java*, la variable *mTextMessage* no se utilizaba. Se ha solucionado eliminando dicha variable.
- En la clase *ParadasFragment.java*, no se utilizaba la variable *dataCommunication*. Se ha solucionado eliminando dicha variable.
- En la clase *ParadasFragment.java*, se utilizaba una variable entera como *String* utilizando un casteo poco ortodoxo. Dicho casteo se ha sustituido por la invocación del método *toString* de la clase *Integer*.
- En la clase *ParadasFragment.java*, no constaba la anotación *@Override* en la signature del método *onOptionsItemSelected*. Se ha solventado añadiendo dicha anotación.

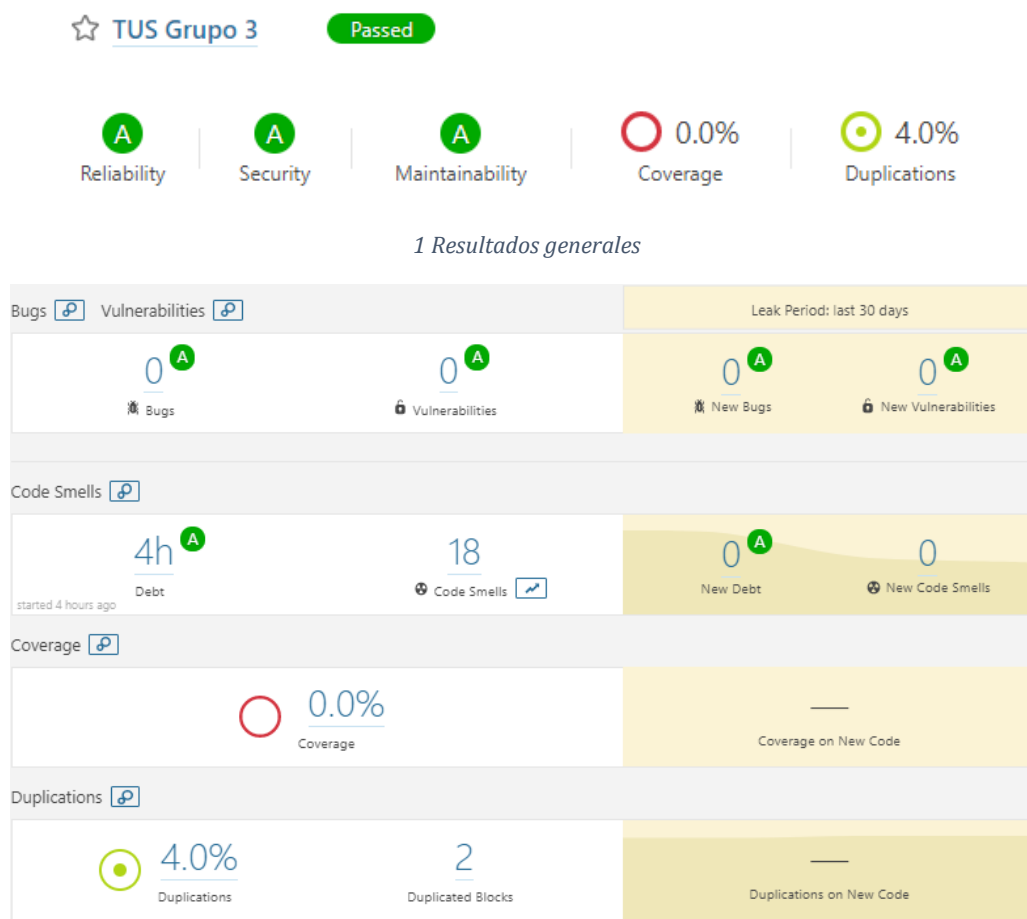
Severidad menor

A continuación, se enumeran algunos de los *issues* de severidad menor tratados. Como se ha mencionado previamente, se han solventado algunos más que no se enumerarán a continuación debido a que eran cambios muy pequeños corregidos con SonarLint y cuya importancia era mínima, sin apenas impacto en la calificación final.

- En la clase *ParserJSON.java*, se ha movido la declaración de la variable local “numero” a la línea de código siguiente.
- En varias clases, se han eliminado *imports* que no se utilizaban.
- En la clase *ActivityInicialLogo.java* se ha modificado el atributo *DURACION_SPLASH* para que sea estático.
- En la clase *ParadasFragment.java*, se ha corregido la creación de una variable local *view* por un retorno directo del resultado necesario.

No se han eliminado los bloques de código duplicado debido a que son imprescindibles para el correcto funcionamiento de la aplicación.

La corrección de todos los *issues* citados anteriormente arroja los resultados siguientes:



2 Deuda técnica y detalles

Display Mode			
Issues		Effort	
▼ Type			Clear
🐛 Bug		0	
🔒 Vulnerability		0	
🔗 Code Smell		18	
▼ Resolution			
Unresolved		18	Fixed
False Positive		0	Won't fix
Removed		0	
▼ Severity			
🔴 Blocker		0	🟢 Minor
🔴 Critical		0	🔵 Info
🔴 Major		2	

3 Tipos de issues y gravedad

Puede comprobarse que la calidad del código ha mejorado sustancialmente: se han reducido en más de un 60% los *code smells* y la deuda técnica ha bajado de 6h a 4h.

En revisiones posteriores, será necesario revisar las duplicidades existentes en las clases *ListParadasAdapter.java* y *ListParadasAdapterSorted.java*. En este análisis he decidido no corregir esta parte de la aplicación debido a que es muy probable que sea modificada en el siguiente sprint, haciendo desaparecer el problema asociado a la misma.

3. Sumario

En este documento se ha recogido el proceso seguido, los pasos analíticos y las acciones correctivas llevadas a cabo para optimizar la calidad del código asociado a la historia de usuario “Ordenar paradas alfabéticamente”.

Los resultados obtenidos tras la corrección arrojan un balance positivo, habiéndose establecido la calificación máxima (A) en todos los elementos evaluables.