PS17 – Informe de calidad: Historia de usuario: "Consultar listado de líneas"

Resumen

Este documento recoge el informe de calidad asociado a la historia de usuario #241953– "Consultar listado de líneas".

Document ID:	PS17/00/2017-QR003-US242580-ConsultarListadoDeLineas
Departamento:	Calidad y Auditoría, dentro del proyecto integrado
Tipo:	Análisis de calidad
Privacidad:	CONFIDENCIAL
Estado:	ENTREGABLE
Versión:	1.0.2
Fecha:	15/11/2017
Autores:	Sainz-Maza Ruiz, Javier
Revisores:	Martínez Vila, Javier; Cerezo Fernández, Elsa

HISTORIAL DE CAMBIOS

Versión	Fecha	Cambio	Responsable
1.0.0	15/11/2017	Creación y redacción del documento	Sainz-Maza Ruiz, Javier
1.0.1	15/11/2017	Revisión ortográfica	Cerezo Fernández, Elsa
1.0.2	15/11/2017	Revisión estructural y gramatical	Martínez Vila, Javier

1. Introducción

Este documento recoge el análisis de calidad efectuado para la historia de usuario **#241953 – Consultar listado de líneas** del proyecto integrado.

2. Análisis de calidad

2.1. Análisis inicial

Tras realizar el primer análisis de la calidad del código, los resultados obtenido son los siguientes:

- Seis bugs.
- Una vulnerabilidad.
- Treinta code smells.
- Código duplicado en un 4,6%.
- Deuda técnica de seis horas.

Respecto a la severidad de los *issues* encontrados (contenidos dentro de lo citado anteriormente), pueden distinguirse:

- Dos críticos.
- Doce de severidad mayor.
- Quince de severidad menor.
- Ocho informativos.

2.2. Acciones correctivas

Se ha establecido el siguiente orden de prioridad a la hora de efectuar las acciones correctivas:

1. Bugs.
2. Vulnerabilidades.
3. Code smells.

Dentro del orden previo, se ha dado mayor importancia a aquellos issues que presentaban mayor severidad.

2.2.1. Bugs

Clase Línea

- Bug de severidad mayor : *NullPointerException* sin tratar.
    - Comprobar que el objeto afectado no fuese null.
- Bug de severidad menor : el tipo de objeto pasado como parámetro podría no ser apropiado.
    - Comprobación del tipo de objeto en el propio método (*getClass()*).
- Bug de severidad menor: debe sobrescribirse el método *hashCode*.

    - Override del método *hashCode()* calculando el código mediante la multiplicación de los distintos campos que conforman el objeto.

Clase Parada

- Bug de severidad menor: el tipo de objeto pasado como parámetro podría no ser apropiado.
    - Comprobación del tipo de objeto en el propio método (*getClass()*).
- Bug de severidad mayor: *NullPointerException* sin tratar.
    - Comprobar que el objeto afectado no fuese null.
- Bug de severidad menor: debe sobrescribirse el método *hashCode*.
    - *Override* del método *hashCode* calculando el código mediante la multiplicación de los distintos campos que conforman el objeto.

2.2.2. Vulnerabilidades

Clase ParadasFragment

- La variable paradas debe convertirse en estática o privada.
    - Reconversión de variable en privada.

2.2.3. Code smells

Clase LoadDataAsync

- Code smell de severidad crítica: idénticos *Strings* duplicados no inicializados como constantes.
    - Creada una constante con el valor a utilizar (ERROR).
- Code smell de severidad menor: constante no estática que debería serlo.
    - Declaración de dicha constante como estática.
- Code smell de severidad mayor: comentarios con código.
    - Eliminados los comentarios que contenían instrucciones Java.

Clase ListLineasAdapter

- Code smell de severidad mayor no corregido (bloque de código duplicado).
    - No tratado.

Clase ParserJSON

- Code smell de severidad mayor: bloque de código duplicado en dos casos (lectura de paradas/líneas y lista de ambas).
    - Creado método que realiza lectura general (para cada caso uno) y concreta en función del parámetro recibido.

Clase ListLineasFragment

- Code smell de severidad mayor: comentarios con código.
    - Eliminados los comentarios que contenían instrucciones Java.

Clase Database

- Code smell de severidad crítica: idénticos *Strings* duplicados no inicializados como constantes.
    - Creada una constante con el valor a utilizar (DROP TABLE IF EXISTS).

Otros code smells comunes en varias clases

- Tener declarados *imports* que no se utilizan han producido code smells de severidad menor en distintas clases.
    - Eliminados dichos *imports*.
- Code smells informativos corregidos parcialmente: sólo aquellos que con un bajo esfuerzo suponen una mejora sustancial.
- La declaración de variables que no han sido posteriormente utilizadas ha producido code smells de severidad menor en diferentes clases.
    - Eliminadas dichas variables y creados métodos observadores (*getters)*.

2.3. Análisis final

Los resultados obtenidos, tras aplicar las correcciones citadas y realizar el análisis final, indican una mejora notable en la calidad del código. A continuación, se listan los resultados:

- Cuatro horas de deuda técnica.
- Once code smells.
    - Diez informativos.
    - Uno de severidad mayor.
- Calificación “A” en todos los apartados.
- *Failed* en la cobertura de pruebas por motivos desconocidos.

3. Sumario

En este documento se ha recogido el proceso de análisis de calidad llevado a cabo para optimizar la calidad del código asociado a la historia de usuario *"Consultar listado de líneas"*.

Los resultados obtenidos tras la corrección presentan un balance positivo, habiéndose establecido la calificación máxima (A) en todos los elementos evaluables. El fallo al pasar el análisis se afrontará en posteriores sprints, tras consultar con el profesor responsable de la asignatura.