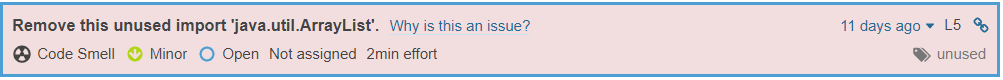
**-InformeController class Refactoring-**

**REFACTOR #1 Eliminar imports que no se usan**

Imagen que contiene interior, viendo, oscuro, naranja

Descripción generada automáticamente

En este *minor code smell* observamos un import que no se está usando. Se ha solucionado eliminando dicho import.

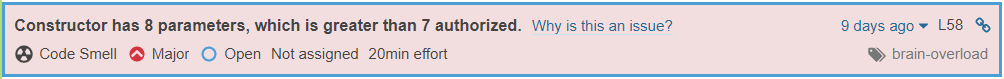
**REFACTOR #2 Eliminar campos privados que no se usan**



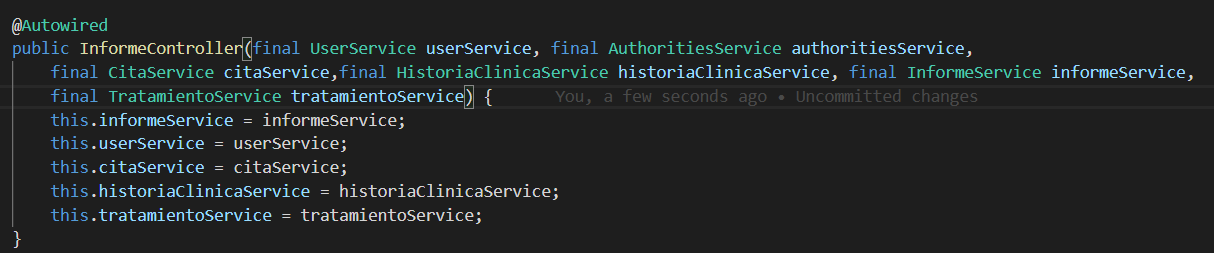
En este *major code smell* observamos unos campos privados ‘’pacienteService’’ y ‘’medicoService’’ que no están siendo utilizados. Se ha solucionado eliminando dichos campos.

**REFACTOR #3 Reducir el número de parámetros que se le pasan al controlador (MAX 7)**

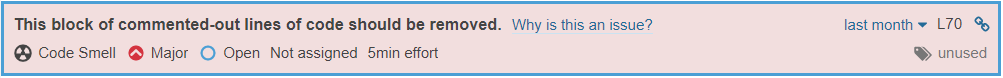
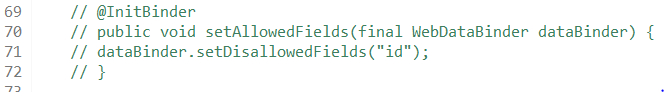
Captura de pantalla de un celular

Descripción generada automáticamente

En este *major code smell* observamos que el constructor recibe 8 parámetros, cuando no debe superar los 7. Se ha solucionado eliminando parámetros del constructor, pues dos de ellos no estaban siendo utilizados. Por lo tanto, el código sin code smell quedaría de la siguiente forma:



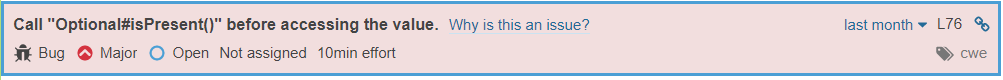
**REFACTOR #4 Eliminar bloque de líneas de código comentadas**



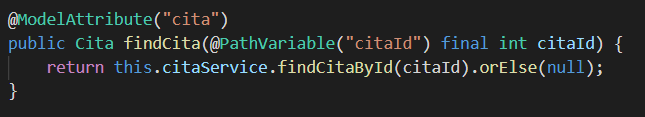
En este *major code smell* obsevamos un *disposable* que se trata de un bloque de líneas comentadas. Se ha solucionado eliminando dicho bloque.

**REFACTOR #5 Mal uso del tipo Optional**

Imagen que contiene cuchillo, tabla

Descripción generada automáticamente

En este *major code smell* no es realmente un code smell sino un bug derivado del uso del tipo Optional para obtener objetos desde la base de datos. Basta con comprobar que el objeto obtenido de la BBDD está presente y devolverlo, en caso de que no esté devolverá un null:



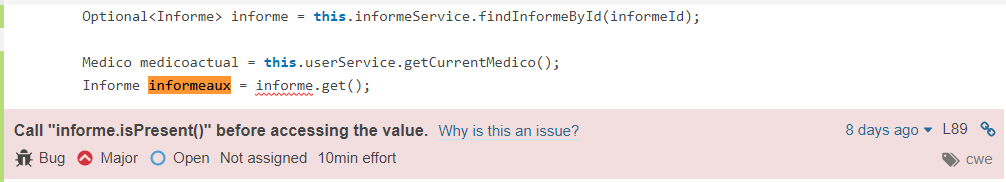
**REFACTOR #6 Lanzamiento innecesario para el método**

Captura de pantalla de un celular

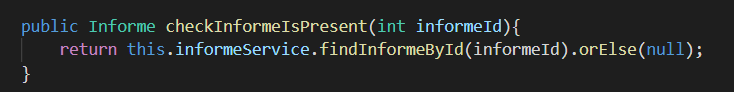
Descripción generada automáticamente

En este *minor code smell* observamos que la excepción DataAccessException debe ser eliminada. Se ha solucionado eliminando dicha excepción.

**REFACTOR #7 Declaraciones innecesarias y solución de Bug derivado de mal uso del tipo Optional**



En este *major code smell* es un bug derivado del uso de Optional (Además de un uso de variables innecesarias) para devolver el Informe. Se ha solucionado mediante un método auxiliar que comprueba si el informe está presente y devolverlo en este caso, en caso de que no lo estuviese se devolvería un null:



Quedando una única declaración para el informe en cuestión así:



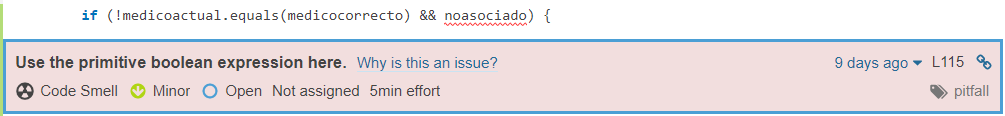
**REFACTOR #8 Mal uso del tipo Optional**

Una captura de pantalla de una red social

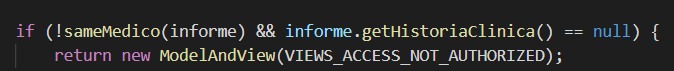
Descripción generada automáticamente

Mismo caso que en el **Refactor#7**, la solución es la misma.

**REFACTOR #9 Declaración Innecesaria**



En este *minor code smell* se empleaba un boolean primitivo en lugar de añadir directamente la condición, por lo que la declaración previa del boolean no era necesaria. La condición queda así tras eliminar la declaración innecesaria.



**REFACTOR #10 Uso repetido del mismo String y declaración innecesaria en la salida del método**

Una captura de pantalla de una red social

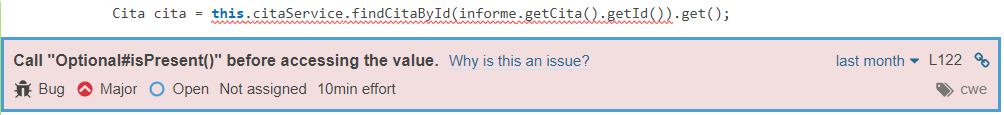
Descripción generada automáticamente

En este *critical code smell* observamos que no se está usando la constante ya definida ‘’VIEWS\_ACCESS\_NOT\_AUTHORIZED’’ y en cambio se está duplicando su valor. Se ha solucionado haciendo uso de la variable ‘’VIEWS\_ACCESS\_NOT\_AUTHORIZED’’.

Respecto al *minor code smell*, se estaba realizando una declaración innecesaria en lugar de devolver el valor directamente. Tras realizar los cambios para ambos code smells queda así, notando que ya no hay una declaración de variable innecesaria y se ha usado la variable string global para el string repetido en cuestión:

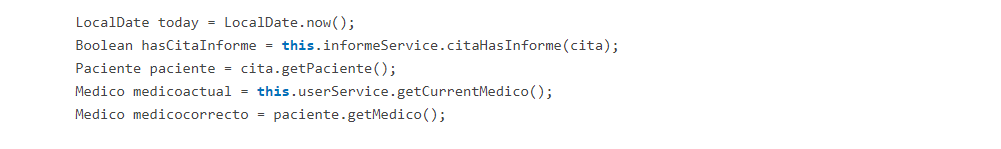


**REFACTOR #11 Bug derivado del uso del Tipo Optional**



Mismo caso que en el **Refactor#5**, se emplea mismo el método auxiliar para obtener la Cita comprobando su presencia.

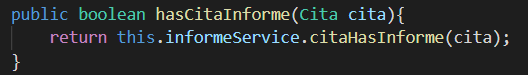
**REFACTOR #12 Uso de objeto Boolean primitivo para una condición y declaraciones innecesarias:**



Captura de pantalla de un celular

Descripción generada automáticamente

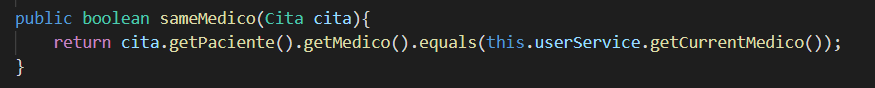
En este *minor code smell* observamos el uso de una expresión boolean primitiva, este code smell se ha solventado mediante la inclusión de un método auxiliar que comprueba si el Informe tiene una Cita:



Quedando la condición así:



También se ha de notar que se estaban realizando declaraciones innecesarias en el método, para lo que hemos creado un método auxiliar que comprueba el médico que accede mediante la Cita:



Respecto a la variable que obtenía el día actual, dado que solamente era usada en una condición if, se ha integrado la declaración del día actual en la condición:



No quedando ninguna declaración previa a las condiciones.

**REFACTOR #13 Uso repetido de String durante el resto del Código**

Captura de pantalla de un celular

Descripción generada automáticamente

En este *critical code smell* podemos observar que el literal ‘’informe’’ se repite hasta 4 veces en el resto de métodos. Por lo tanto, para solucionar este code smell se ha creado una constante estática con el valor ‘’informe’’:



Quedando las declaraciones que hacen uso de ella así:



**REFACTOR #14 Condición que usa un tipo primitivo Boolean**

Captura de pantalla de un celular

Descripción generada automáticamente

En este *minor code smell* observamos el uso de una expresión boolean primitiva, siendo el mismo caso que en el **Refactor#12** se ha empleado el mismo método auxiliar y se ha añadido una llamada a dicho método en la condición.

**REFACTOR #15 Literal duplicado**

Una captura de pantalla de una red social

Descripción generada automáticamente

En este *critical code smell* podemos observar que el literal ‘’redirect:/citas/’’ se repite hasta 5 veces. Por lo tanto, para solucionar este code smell se ha creado una constante estática con el valor ‘’ redirect:/citas/’’:



Quedando este tipo de declaraciones de redirección así:



**REFACTOR #16 Lanzamiento innecesario para el método**

Captura de pantalla de un celular

Descripción generada automáticamente

En este *minor code smell* podemos observar que la excepción DataAccessException debe ser eliminada al no poderse lanzar este tipo de excepción en el método.

**REFACTOR #17 Bug derivado del uso del Tipo Optional**

Una captura de pantalla de una red social

Descripción generada automáticamente

Mismo caso que en el Refactor#7, se emplea el mismo método auxiliar con el mismo funcionamiento para solventarlo.

**REFACTOR #18 Declaraciones de Lanzamiento de Excepción Innecesarias**

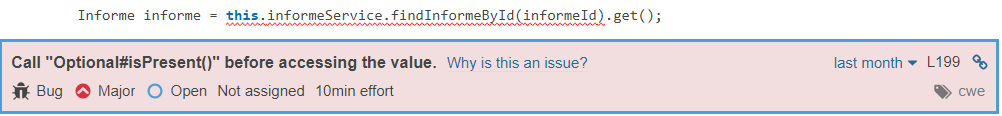
Captura de pantalla de un celular

Descripción generada automáticamente

En el primer *minor code smell*, podemos observar que dicha IllegalAccessException debe ser removida ya que no puede ser lanzada desde el cuerpo del método. Se ha solucionado dicho code smell eliminando del código dicha excepción.

En el segundo *minor code smell*, podemos observar que la excepción DataAccessException debe ser eliminada al ser una runtime exception. Por lo tanto, se ha solucionado eliminando del código dicha excepción.

**REFACTOR #19 Mal uso del Tipo Optional**



Mismo caso que en el **Refactor#7**, se ha empleado el mismo método auxiliar con el mismo funcionamiento para solventarlo. La declaración del objeto queda igual que en dicho refactor.

**REFACTOR #20 Mal uso del Tipo Optional**

Captura de pantalla de un celular

Descripción generada automáticamente

Mismo caso que en el **Refactor#7**, se ha empleado el mismo método auxiliar con el mismo funcionamiento para solventarlo. La declaración del objeto queda igual que en dicho refactor.

**AUTORES:**

**Andrés Aguilar Alhama**

**Iván Hernández Rodríguez**