Trabajo 3. Mantenibilidad y refactorización

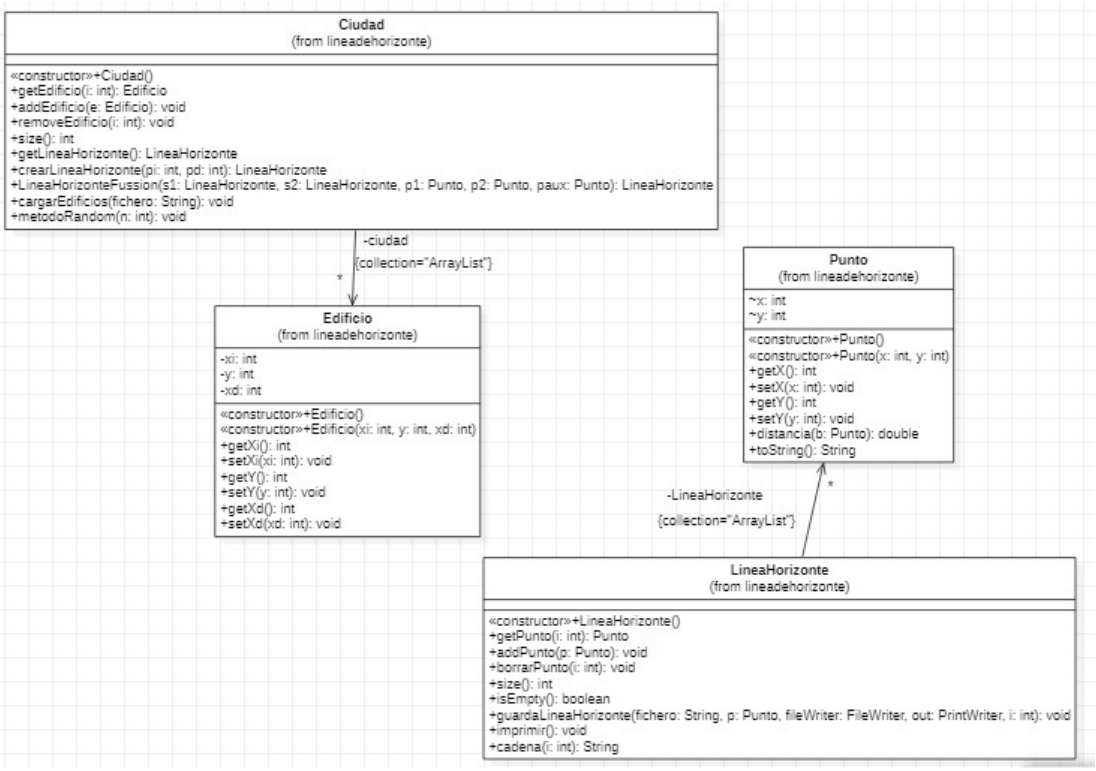
EVOLUCIÓN Y MANTENIMIENTO DEL SOFTWARE ©2021

Héctor Cantero Álvarez bl0523

Contenido

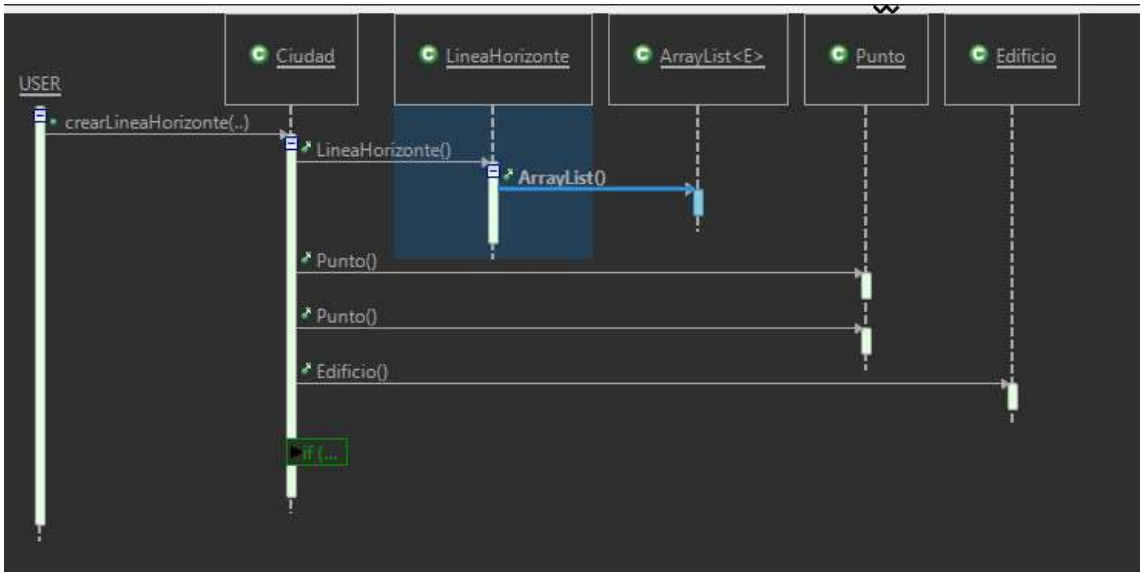
Análisis inicial

Diagrama de clases inicial



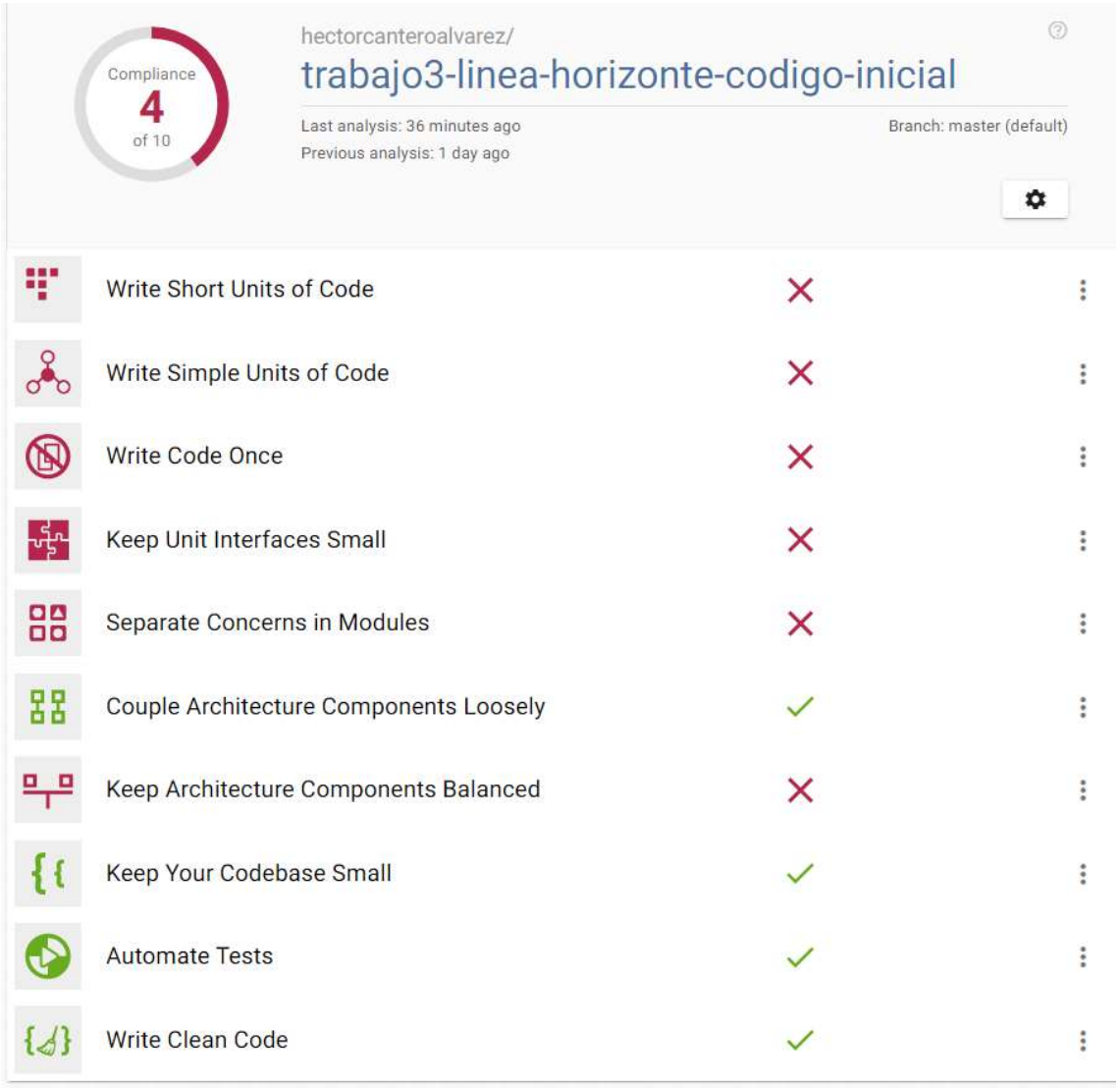
El diagrama de clases inicial muestra la relación entre las clases Ciudad-Edificio y Punto-LineaHorizonte. Una Ciudad está compuesta de edificios y una LineaHorizonte por puntos, lo cual concuerda con los atributos de las clases vistos en el código, sin embargo, como veremos a continuación las clases están más relacionadas.

Diagrama de secuencia del método “crearLineaHorizonte”



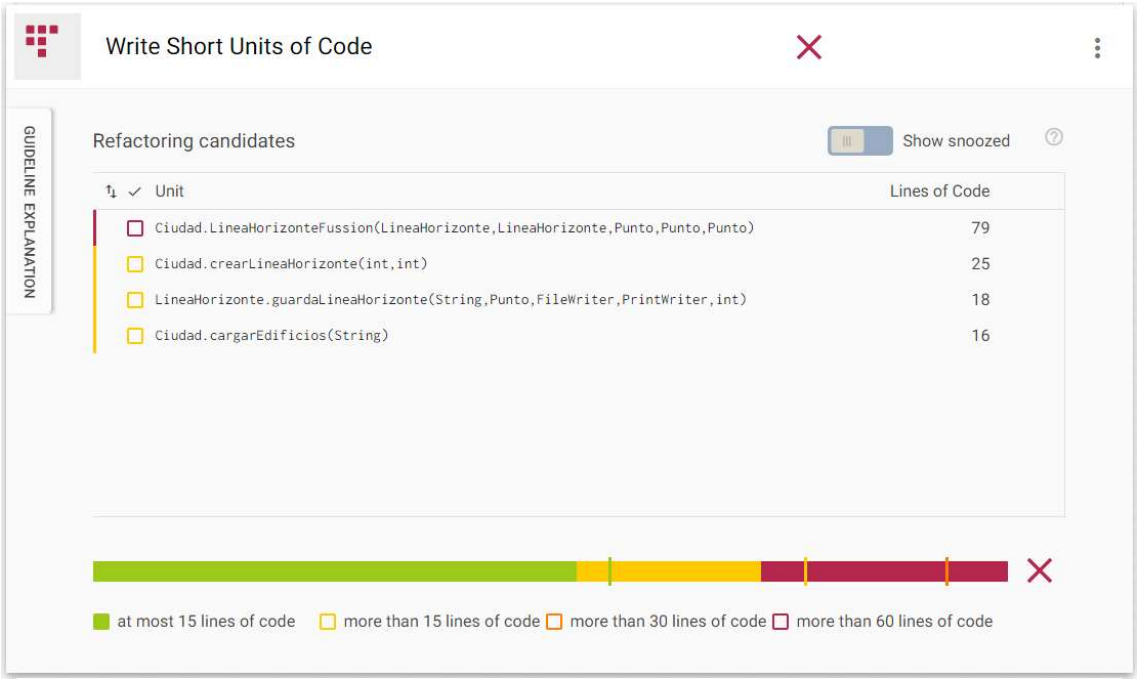
Podemos ver que el método es bastante complejo, ya que hace uso de todas las clases, a pesar de que una *Ciudad* solo tiene *Edificios,* para generar la *LineaHorizonte* de una Ciudad es necesario usar está clase, que a su vez trata con la clase *Punto*.

Análisis de “bad smells” con Better Code Hub



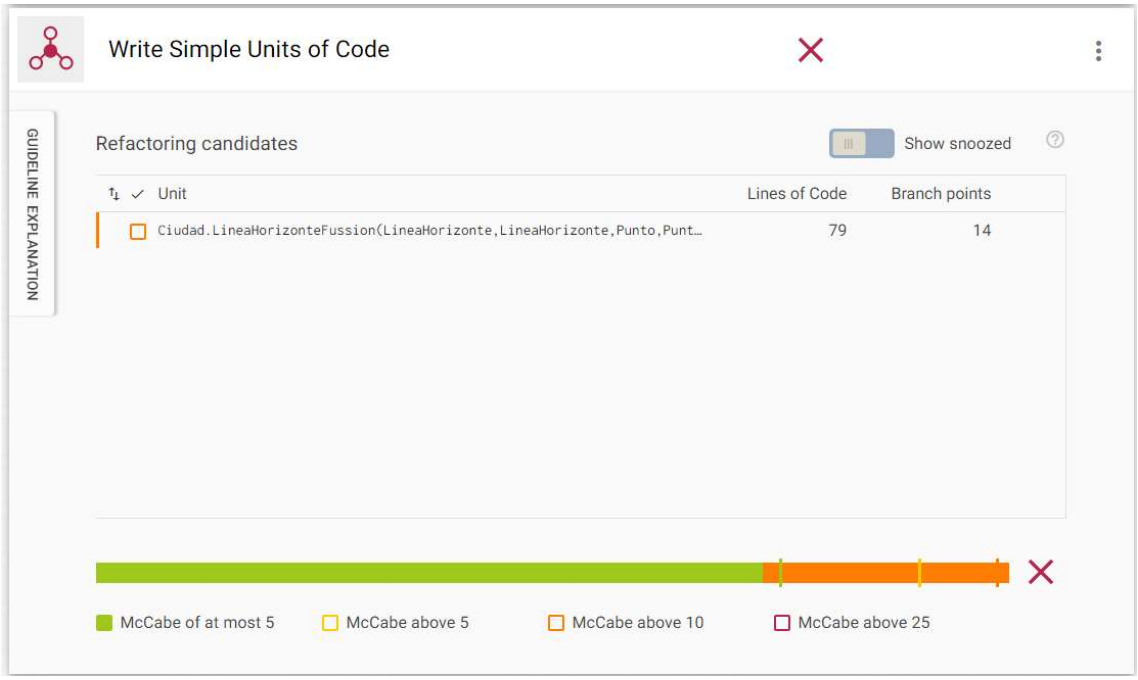
El nivel de conformidad inicial es 4. A continuación se muestra en detalle cada “bad smell”.

Unidades de código extensas



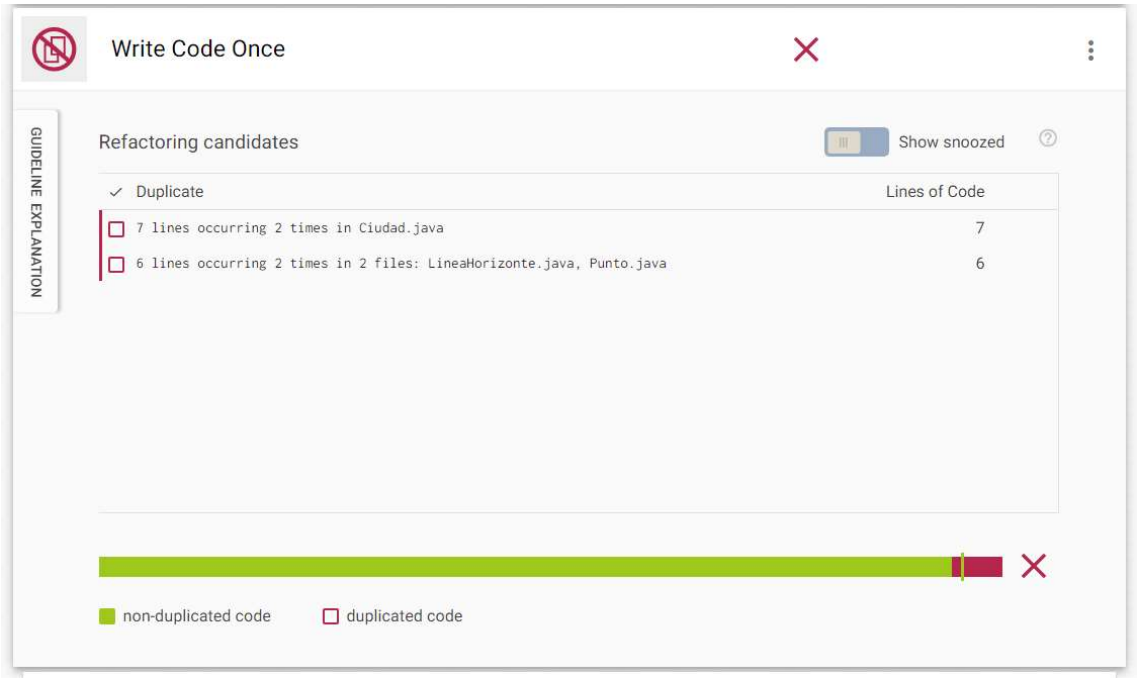
Cómo podemos ver, la longitud de los métodos es demasiado grande, en especial en la clase *Ciudad* donde el método *LineaHorizonteFusion* tiene más de 60 líneas. Es necesario refactorizar estos métodos descomponiéndolos en unidades de código más cortas.

Unidades de código complejas



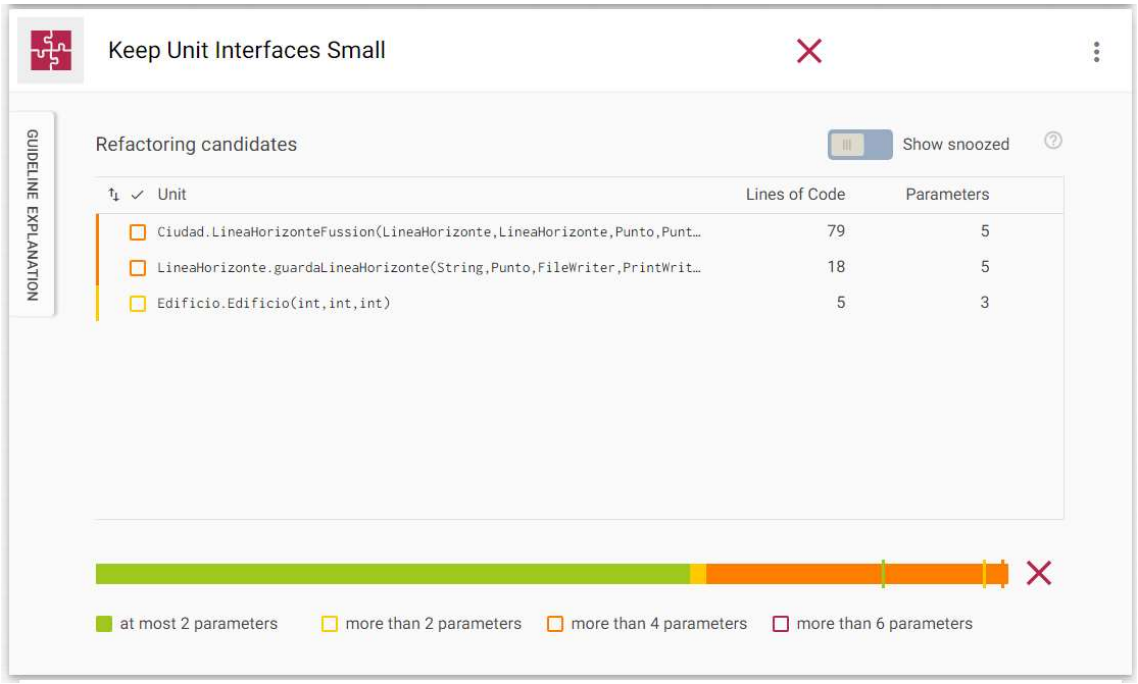
La complejidad ciclomática del método *LineaHorizonteFussion* es demasiado elevada y esto daría lugar a un número de pruebas mínimo muy elavado para cubrir todos los caminos de ejecución.

Código repetido



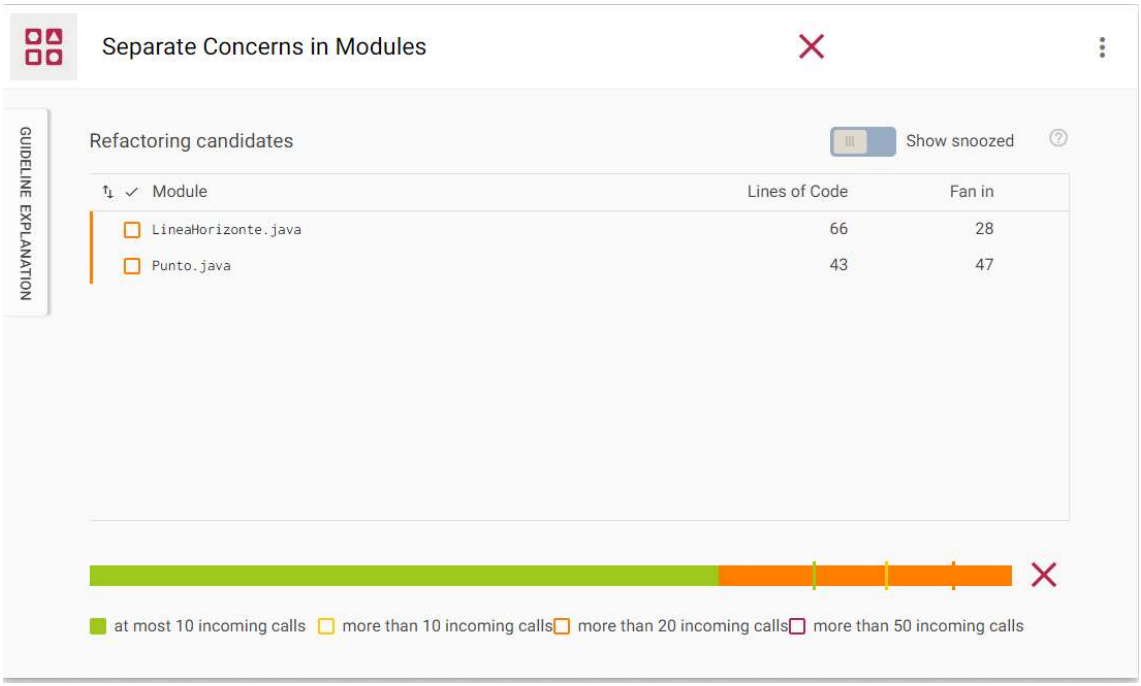
Tenemos código idéntico en varias funciones, por lo que es posible refactorizar el código e implementar un método que satisfaga a ambas funciones evitando repeticiones.

Interfaces extensas



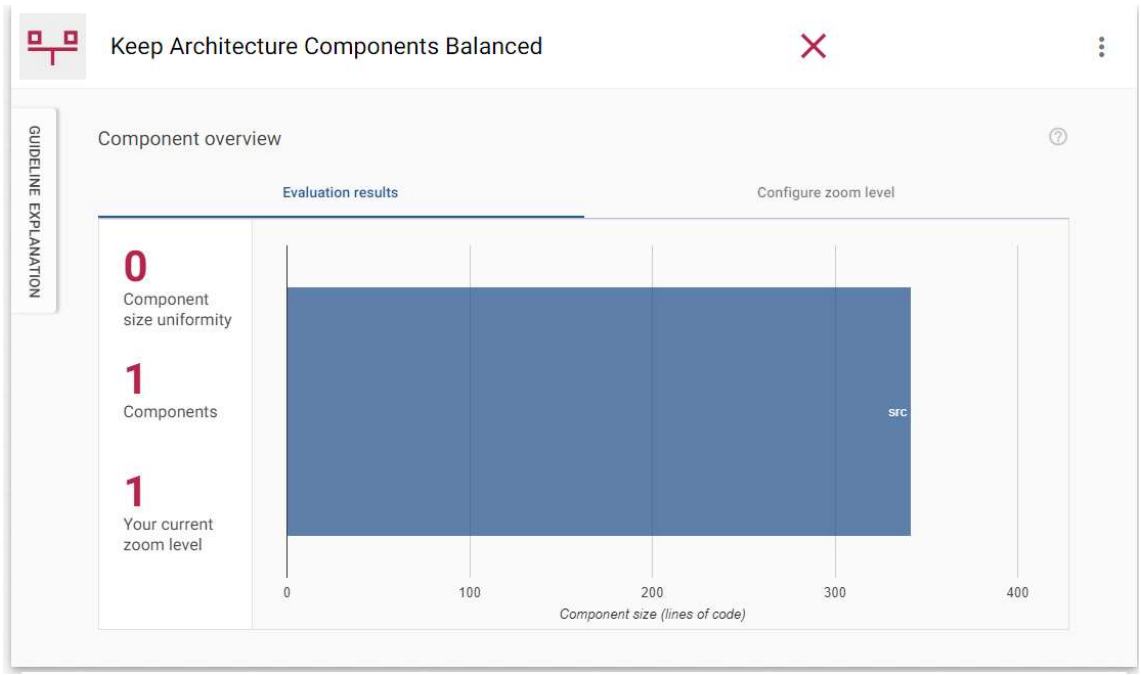
El número de parámetros de entrada es bastante elevado, hay que comprobar si realmente es necesario pasar todos esos parámetros y en caso de ser así, implementar una manera más eficaz de hacerlo.

Conceptos fuera de módulos



Las llamadas a métodos de otras clases son muy elevadas, por lo que es posible que sea necesario pasar las responsabilidades de las funcionalidades que se esté tratando directamente a dichas clases.

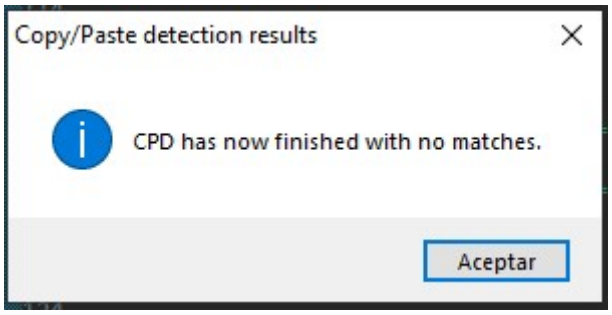
Componentes arquitectónicos desbalanceados



El tamaño relativo de los componentes está desequilibrado, algunas clases tienen un tamaño mucho más extenso que otras.

Anális de “bad smells” con PMD

Copy/Paste



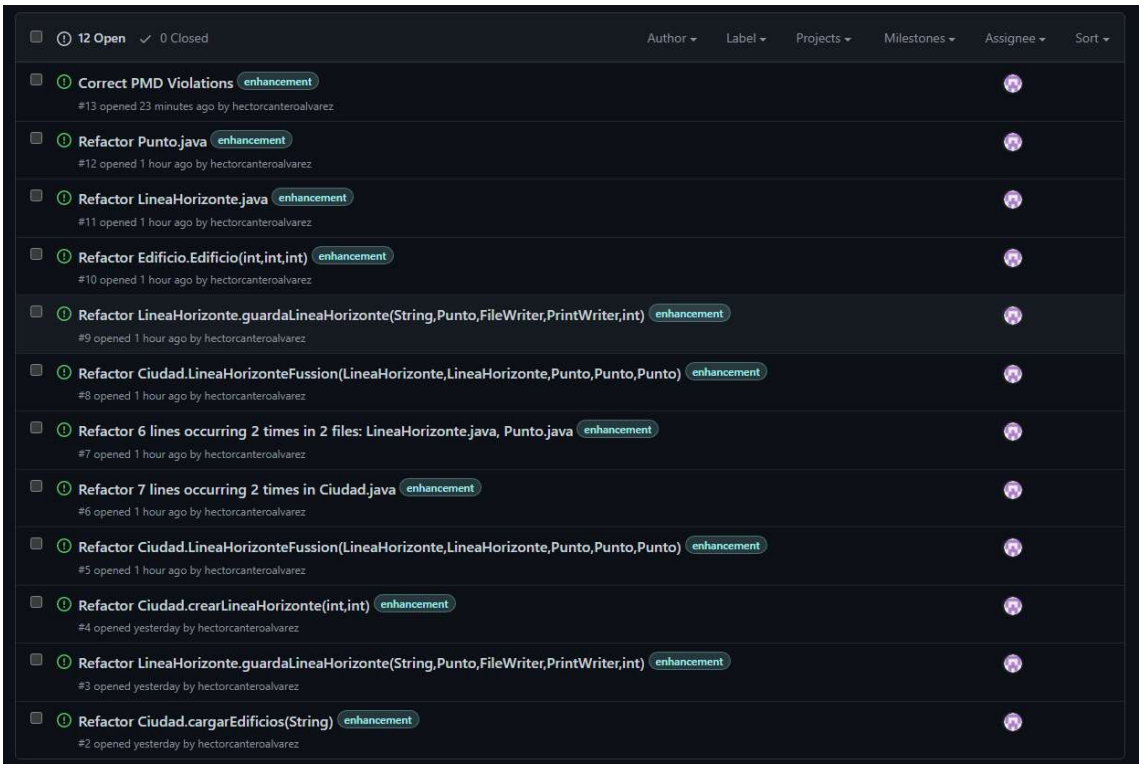
El análisis de Copia/Pega no encuentra ningún bloque de código sospechoso

Violaciones

Element	# Violations	# Violations/KLOC	# Violations/Method
etsisi.ems2020.trabajo3.lineadehorizonte	74	N/A	N/A
etsisi.ems2020.trabajo3.lineadehorizonte	241	923.4	4.30
Ciudad.java	134	827.2	4.79
Edificio.java	22	956.5	2.75
LineaHorizonte.java	48	1230.8	4.00
Main.java	14	1076.9	14.00
Punto.java	23	958.3	3.29

El análisis de violaciones en cambio, si que muestra un número considerable de ellas, por lo que habrá que revisarlas y solucionar tantas cómo sea posible.

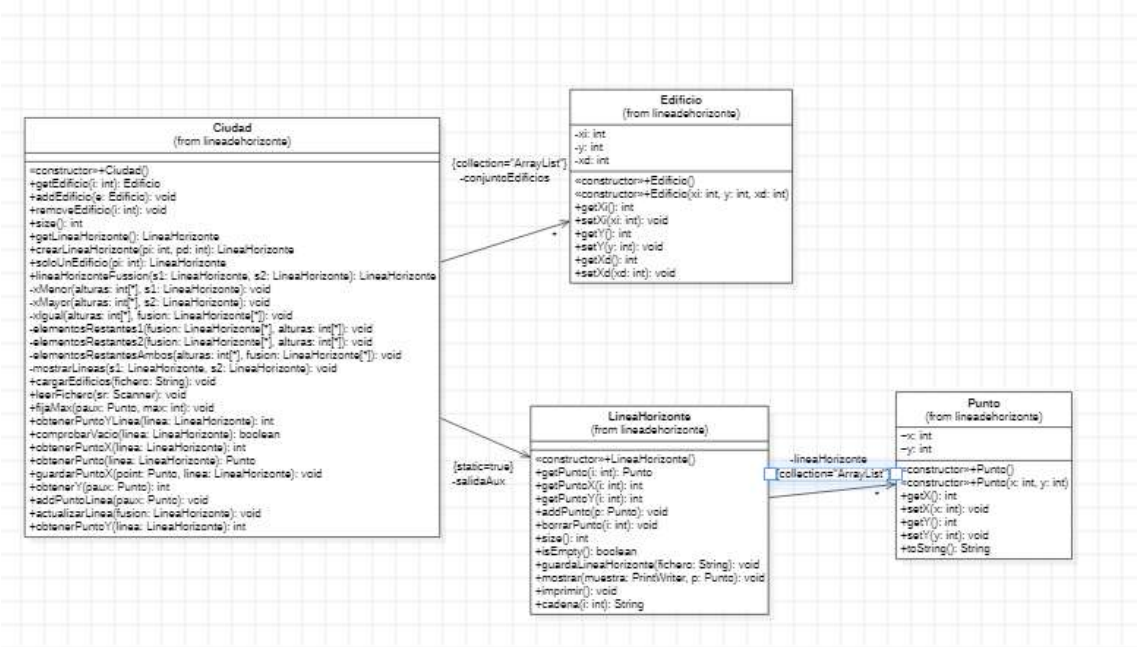
Issues creados



Tras el análisis inicial, estos han sido los “issues” creados que se tratarán de resolver en este trabajo.

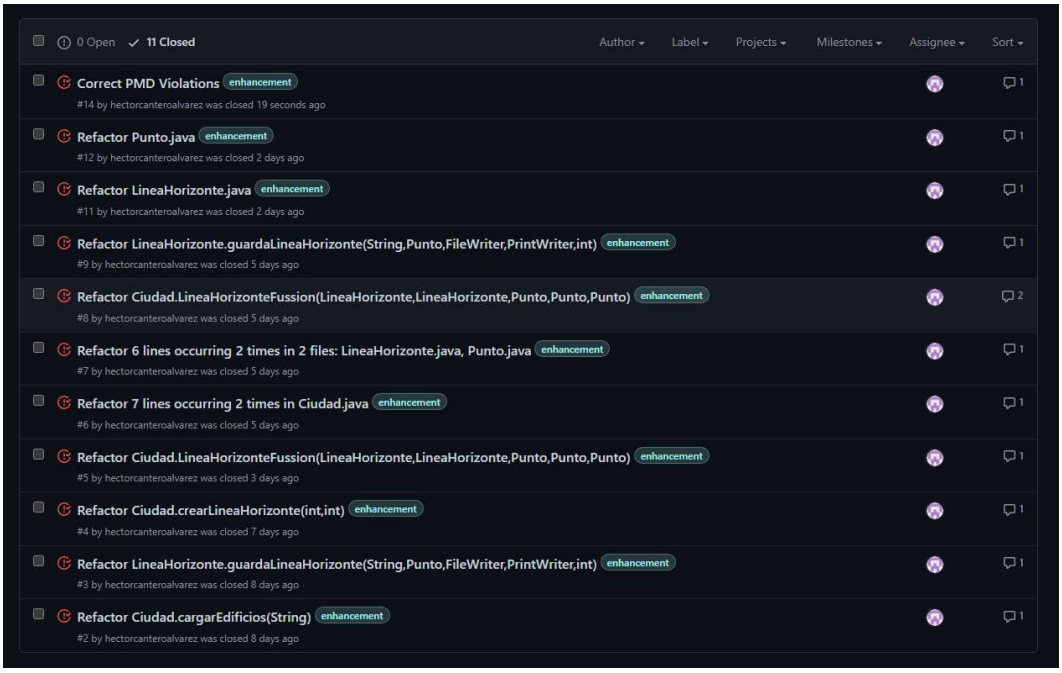
Resultados de la refactorización

Nuevo diagrama de clases



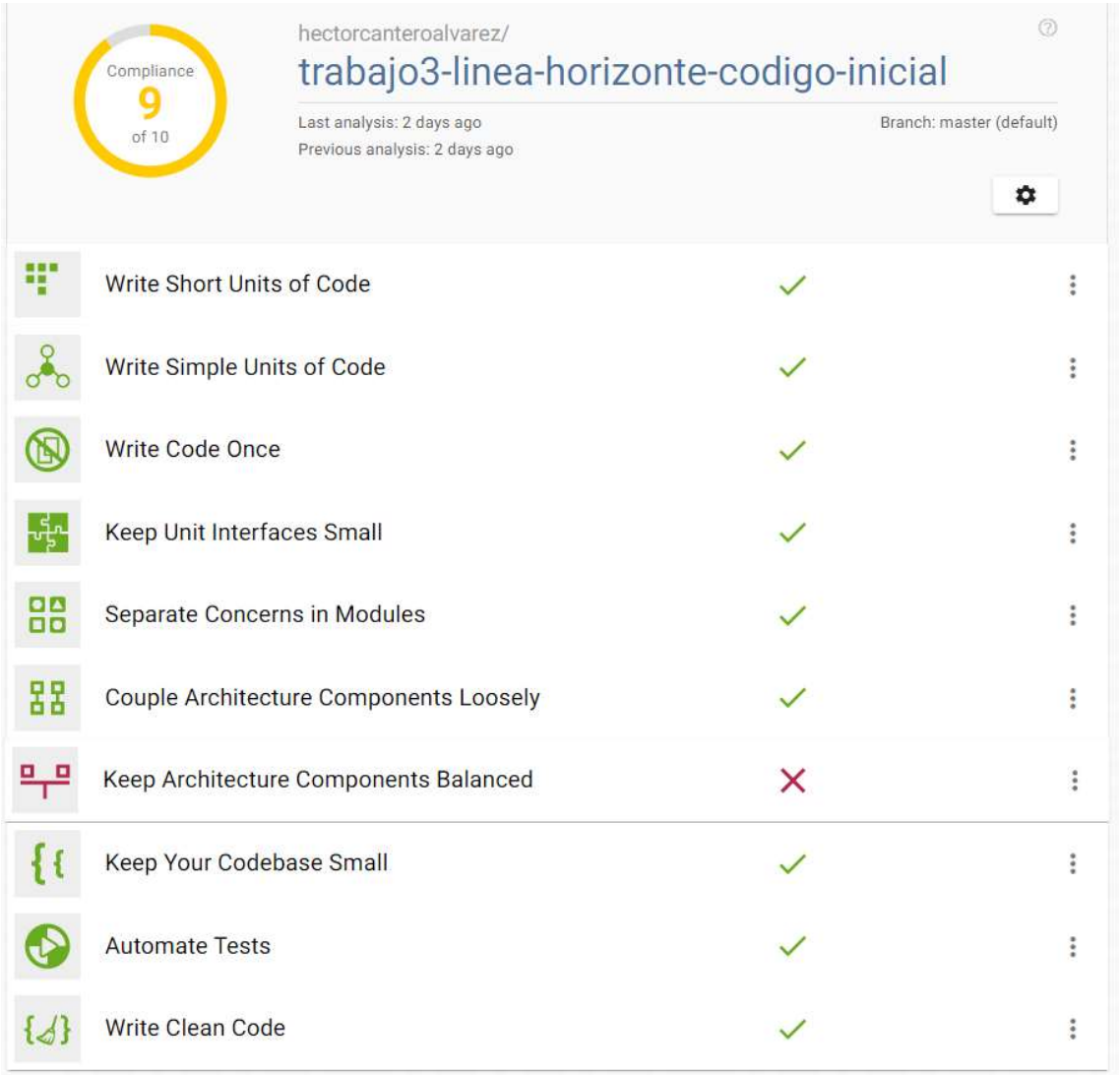
Tras la refactorización, este es en nuevo diagrama de clases. Cómo se puede apreciar, el número de métodos de *Ciudad* ha crecido considerablemente y ahora existe una relación directa entre esta y *LineaHorizonte,* ya que cada ciudad, tiene una *LineaHorizonte.*

Issues resueltos



Estos son los “issues” que se han resuelto al refactorizar el código. No todos han podido ser resueltos, como se puede apreciar, sin embargo, como veremos a continuación, el nivel de conformidad ha aumentado considerablemente.

Nivel final de conformidad alcanzado



#2 Se ha eliminado el código muerto de *cargarEdificios* y añadido un nuevo methodo (aplicando extract method) *leerFichero* para reducir el número de líneas.

#3 Se ha creado un nuevo método *mostrar* que es llamado desde *guardarLineaHorizonte*, lo cual reduce el número de líneas de este último método al separar las funcionalidades.

#4 Se ha añadido el método *soloUnEdificio*, de forma que se ha reducido el número de líneas en *crearLineaHorizonte* separando la funcionalidad de esta. De paso se han movido algunas variables locales del método a otros métodos, para reducir el número de parámetros de entrada en #8.

#5 Se ha reducido el número de líneas en el método *lineaHorizonteFusion* dividiendo este en funcionalidades diferentes implementadas en varios métodos que son llamados desde el método principal. También se han empleado estructuras de datos de tipo array para reducir el número de parámetros de entrada.

#6 Se ha eliminado código muerto que no se usaba y daba lugar a código duplicado.

#7 Se ha cambiado el código duplicado por la llamada al método *toString* de la clase *Punto*, que ya que tenía la misma funcionalidad.

#8 Se han eliminado los parámetros que no eran necesarios como entrada ya que era posible inicializarlos en el cuerpo del método, reduciendo el número de parámetros de entrada a dos.

#11 y **#12** se ha reducido el número de invocaciones directas a los métodos de otras clases, encapsulando en diferentes métodos dentro de la clase *Ciudad*, lo que facilita modificaciones de código futuras si fuera necesario.

#13 Las violaciones de PMD revisadas son las siguientes:

> Blocker violations: Todas han sido corregidas.
>
> Critical violations: No han sido corregidas porque corresponde a llamadas System.Print.
>
> Urgent violations: La mayoría no han sido corregidas ya que se referían a añadir comentarios, tamaño de los comentarios, tamaño del nombre de las variables o poner variables como final. Se han corregido *cast* innecesarios.
>
> Important violations: Algunas no han sido corregidas porque se referían a quitar paréntesis que sí eran necesarios o usar “Varargs” en los parámetros de entrada de algunos métodos. Otras referidas a cambiar el nombre de variables como “out” o quitar paréntesis que, si eran innecesarios, sí han sido corregidas .