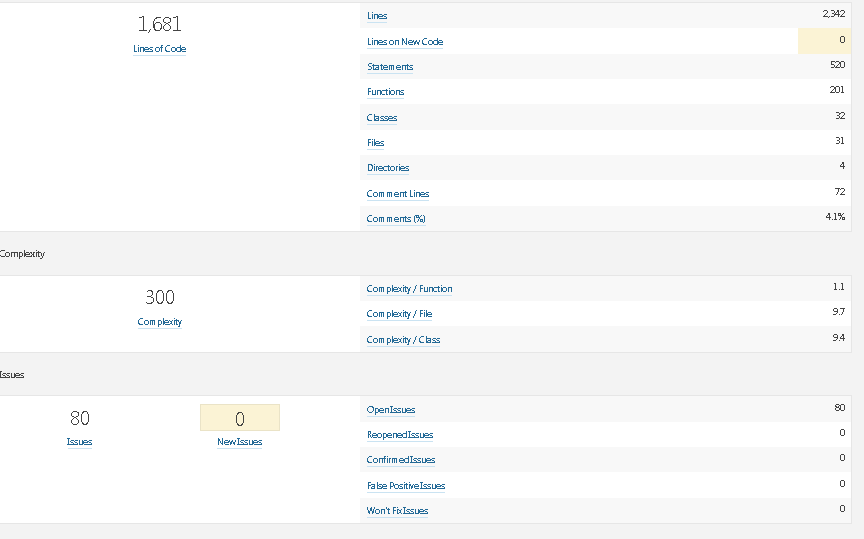
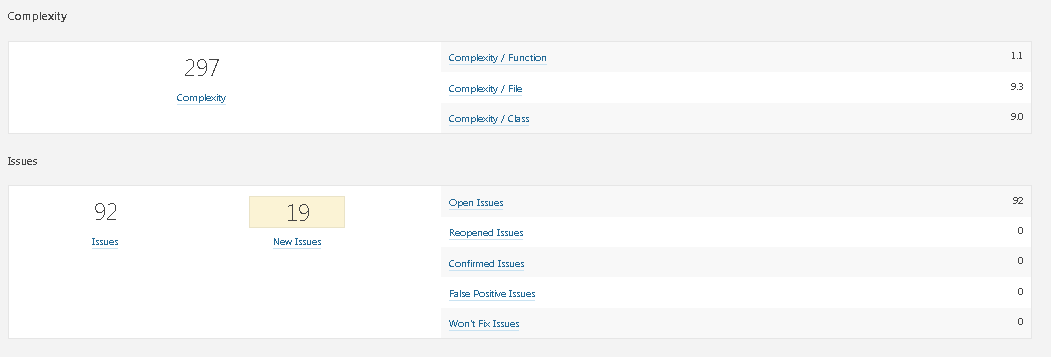
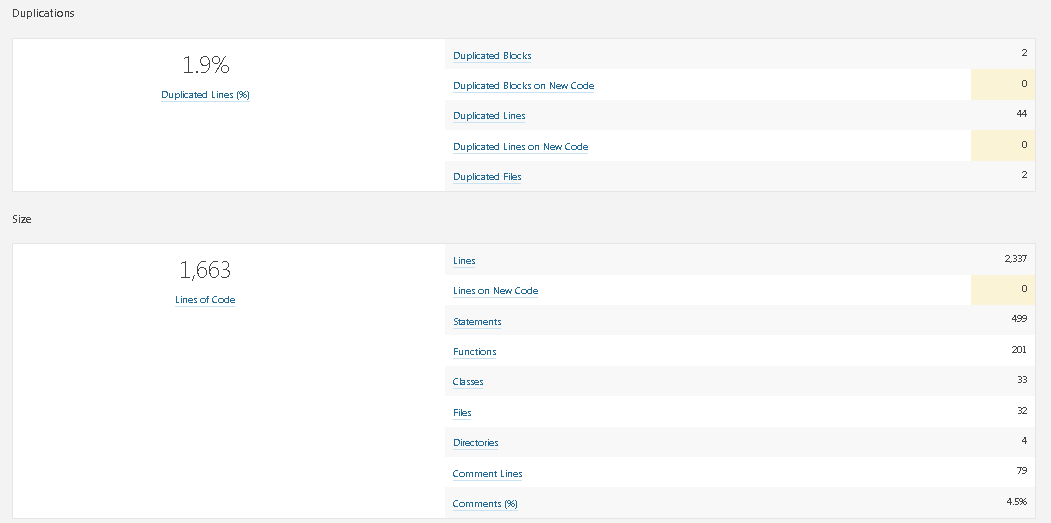
# Metricas sobre patrones de diseño

## Originales



Estas fueron las métricas obtenidas por SonarQube antes de aplicar los patrones. Este es el punto de referencia base para contextualizar las otras métricas.

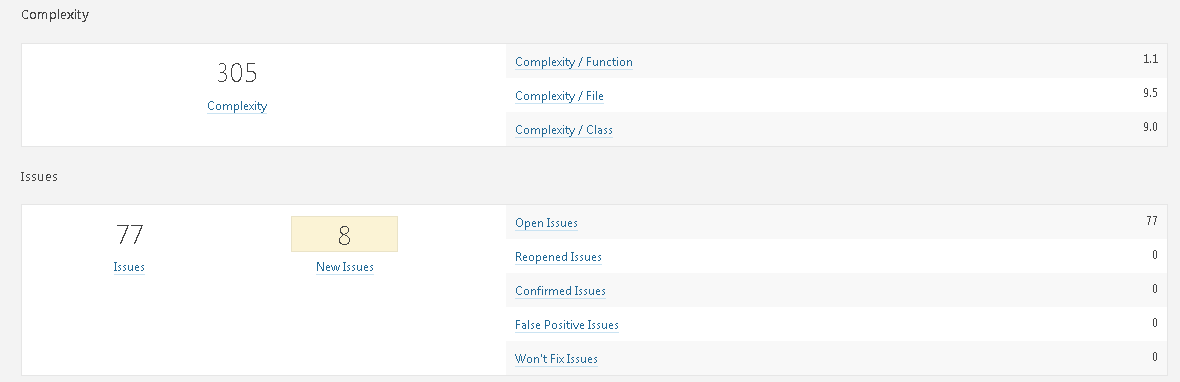
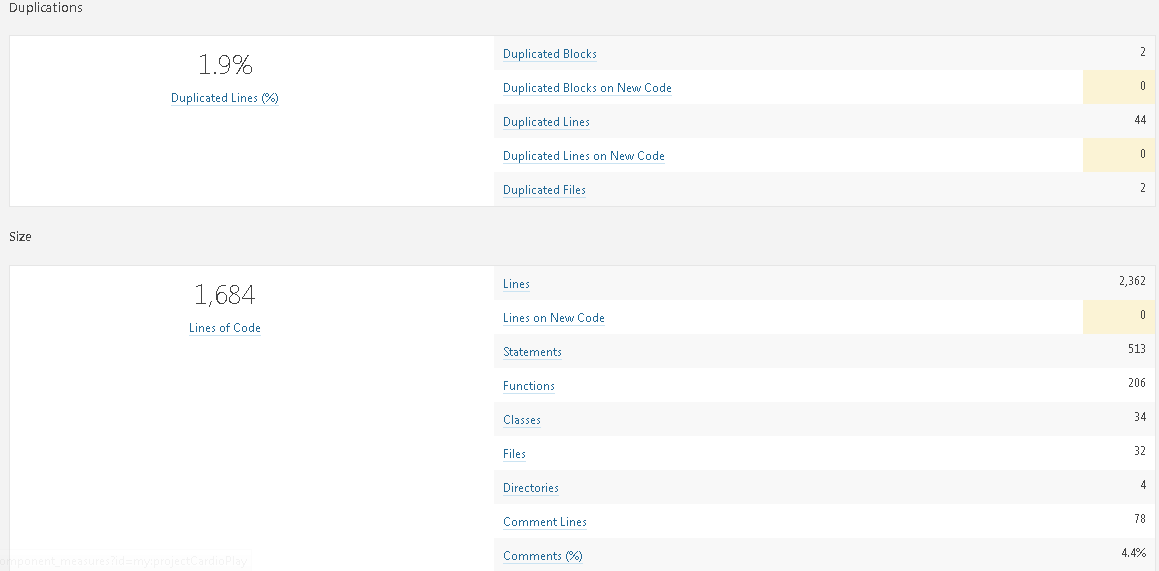
## Después de Factory



Como se puede ver las evaluaciones de las métricas se mantuvieron igual. Los detalles más interesantes se relacionan con la complejidad, esta se vio reducida al aplicar el patrón. Ya que las clases de los modelos que manejaban la lógica de traducir los JSON en objetos estaba en la clase, parte de este código se pudo extraer.

La decisión de aplicar Factory para las mediciones estuvo basada en las similitudes entre las mediciones y la posibilidad de agregar información de otros sensores en el futuro. Debido a la variedad de posibilidades facilitar esta tarea parece una buena idea centralizar el manejo de cómo se lee la información y el formato que esta debe tener.

## After Builder



Se puede apreciar que aumentar las líneas de código afecta la cantidad de issues que se presentan. Aunque las calificaciones principales se mantienen iguales, la complejidad aumento incluso más que la versión original. El builder probablemente no era el mejor patrón para aplicar con solo dos tipos de médicos, pero la alta cantidad de especializaciones puede resultar en variaciones interesantes de características que se pueden manejar bien con un builder.

## After Fachada

## 

La fachada del historial permite separar la información propia del paciente con las entradas que se van teniendo a lo largo del tiempo. Esta se pensó con la finalidad de abstraer la información medica del paciente con la personal. En un futuro si se quiere compartir la información o modificar el historial tener esta ayuda.

Las métricas muestran que la mayoría se mantuvieron igual. Esto se debe a que la fachada implico reorganizar el código, pero no agrego nuevos métodos o funcionalidades; la cantidad de issues se disminuyó, pero aparecieron unos menores asociados con la nueva clase.

## Conclusiones

La aplicación de patrones de diseño busca facilitar la mantenibilidad del código, pero no siempre ayudan a disminuir la complejidad del mismo, muchas veces es necesario agregar clases las cuales aumentan las líneas de código y pueden generar otros problemas que afecten a las métricas. Cuando se comenzó el proyecto se planeó en función de facilidad de hacer cambios y por eso se tuvo buenos valores iniciales. De hecho, los patrones fueron los que agregaron más issues, aunque la escala actual del proyecto no es lo suficientemente grande para que se puedan apreciar los beneficios de los cambios.