# Examen Final – Ramos Paredes Roger Anthony (18200096)

Repositorio: <https://github.com/r3gor/patrones-veranito-chesss>

## Fork al proyecto

Como primer paso hacemos fork al proyecto original para generar una copia en nuestra cuenta personal, en mi caso luego de hacer el fork este es el resultado (mi username en Github es r3gor):

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Clonación del proyecto

Procedemos a clonar el proyecto localmente para poder trabajar con nuestro editor favorito en este caso InteliJ Idea de Jetbrains, adicionalmente se usó Visual Studio Code ya que provee funcionalidades y pluggins que apoyan en el trabajo de la refactorización. Resultado:

Texto

Descripción generada automáticamente

## Creación del proyecto en SonarQube

Levantamos el servidor de SonarQube en este caso con el script StartSonar que nos aparece en los binarios para Windows cuando descargamos el software. Luego ingresamos con nuestras credenciales y creamos el proyecto. Resultado:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## SonarQube Análisis - Estado inicial

Una vez creado el proyecto procedemos a ejecutar el análisis de SonarQube a nuestro proyecto para esto corremos el comando de Maven que nos provee la interfaz de SonarQube, en este caso es:

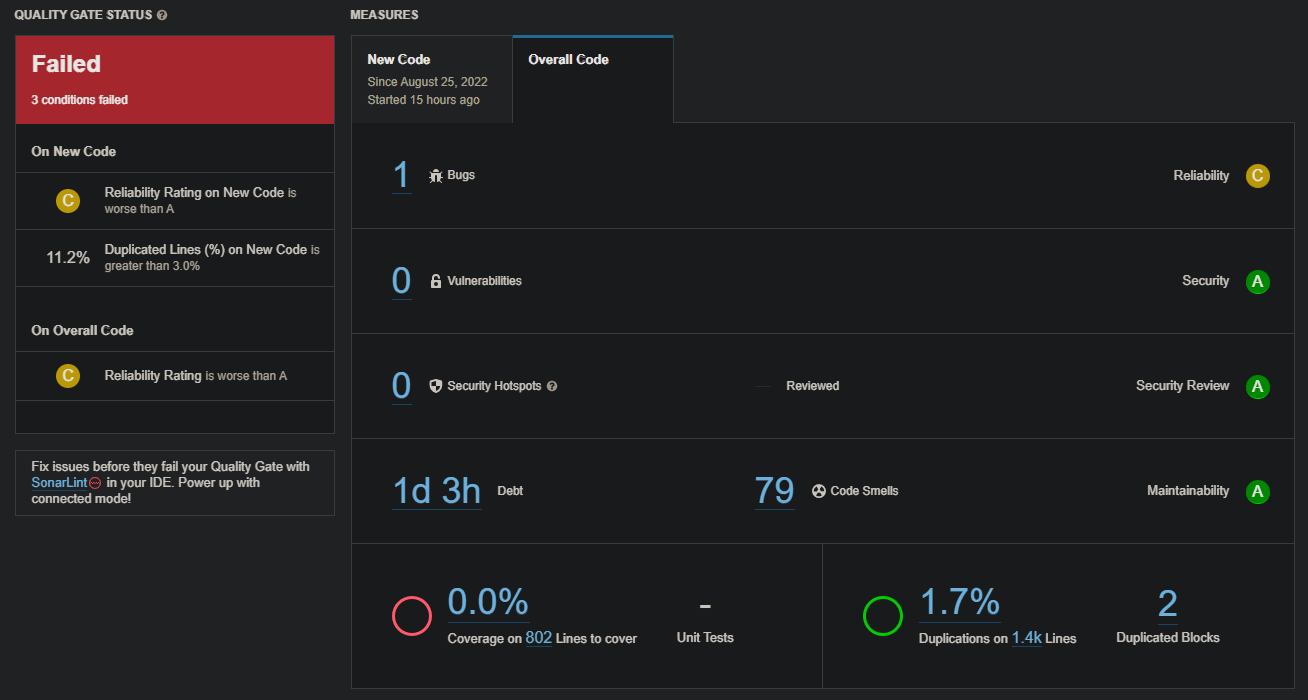
mvn clean verify sonar:sonar -Dsonar.projectKey=chess-exam-final -Dsonar.host.url=http://localhost:9000 -Dsonar.login=sqp\_0f829cf747154794fd169c0c2392c763304896a9

Resultado:

Imagen que contiene Texto

Descripción generada automáticamente

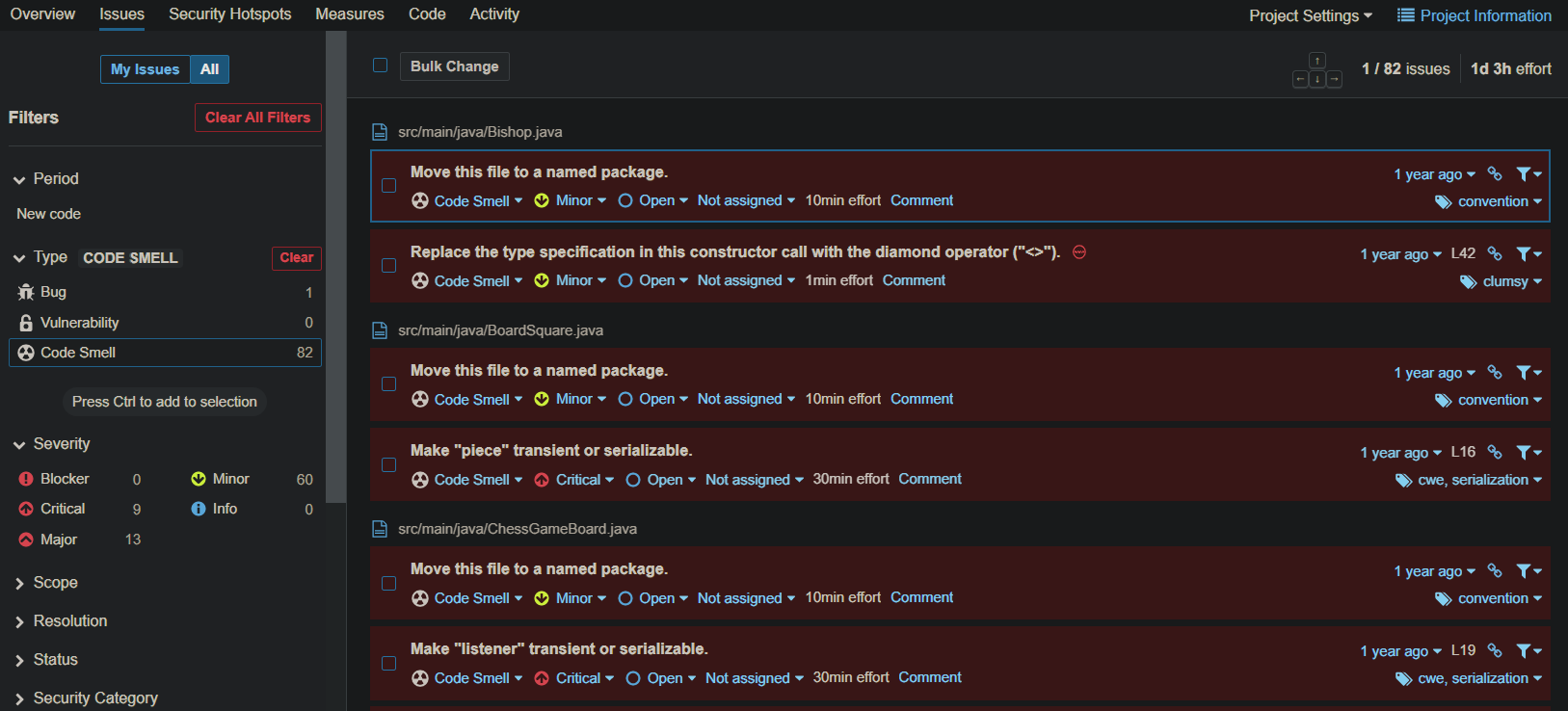
Una vez ejecutado podemos ir a la interfaz de SonarQube para observar los resultados el análisis realizado:



Como resultados de esta pasada inicial con sonar observamos que el test no ha pasado la prueba, tambien podemos observar que existen 79 code smells, 1 bug y 2 bloques duplicados de código. En los siguientes pasos nos encargaremos de hacer las refactorizaciones necesarias para reducir la máxima cantidad de estos problemas.

## Refactorización

Podemos navegar en sonar para ver cada detalle por ejemplo de los code smell:



Captura de pantalla de computadora

Descripción generada automáticamente

De esta forma podemos navegar en cada una de las observaciones del sonar he ir refactorizando una por una en nuestro editor de texto.

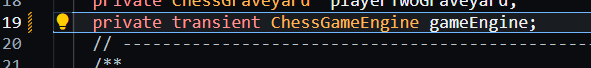
A continuación se muestran algunos ejemplos de la refactorización:

Code smell:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Refactorización:



Code Smell:

Texto

Descripción generada automáticamente

Refactorización:

Captura de pantalla de computadora

Descripción generada automáticamente

Code smell:

Texto

Descripción generada automáticamente

Refactor:

Captura de pantalla con letras

Descripción generada automáticamente

Code smell:

Texto

Descripción generada automáticamente

Refactor:

Texto

Descripción generada automáticamente

Code Smell:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Refactor:

Pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Code Smell:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Refactor:

Texto

Descripción generada automáticamente

Code Smell:

Texto

Descripción generada automáticamente

Refactor:

Texto

Descripción generada automáticamente con confianza media

Bug:

Captura de pantalla de un celular

Descripción generada automáticamente

Refactor:

Texto

Descripción generada automáticamente

Duplicate code:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Refactor:

Movemos el código repetitivo a la clase padre para que de esta forma se pueda generalizar según el parámetro que cambia para cada uno de los casos (en uno debe mandase 1 y en el otro 8)

Método general:

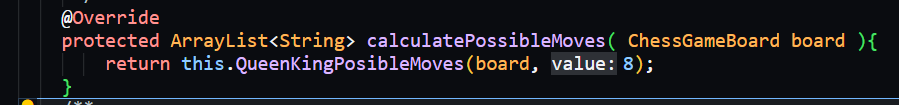
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

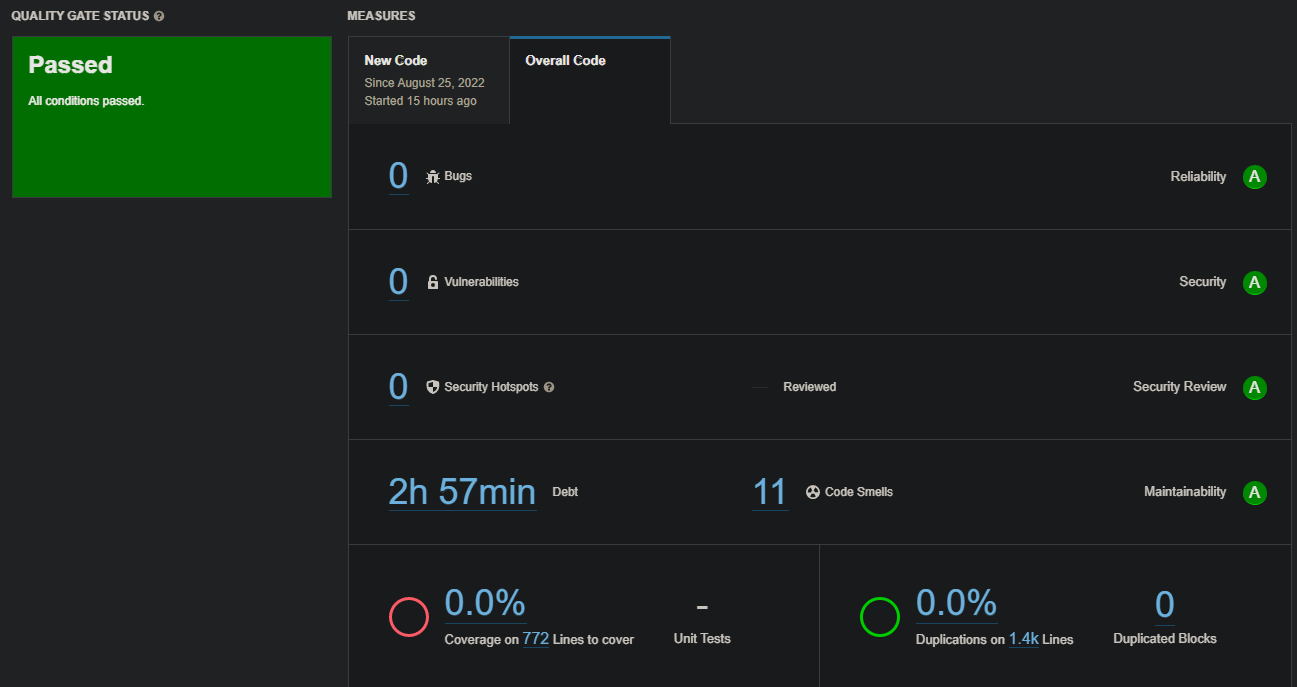
Llamadas especificas para King y Queen:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media



## Resultados de la refactorización – Pasada final SonarQube



**Tanto los duplicated blocks como los bugs se redujeron a cero, mientras que los code smells se redujeron a una cantidad de 11, además el estado ahora es de “Passed”. Por lo tanto se concluye que el código tiene una mejor calidad ahora que antes de la refactorización.**