ANALISIS SONAR SCANNER

Enlace de repositorio: https://github.com/training-practice-sofkau/taller-pruebas-junit

Usuario de Github: Julián Lasso

Contexto: ¿Qué es SonnarQube?

Es una herramienta para realizar análisis estático de código, revisión automática de código para detectar bugs, vulnerabilidades y código apestoso (code smells).

Análisis realizado:

Métricas que observaremos en el análisis:

- Bug: un error que romperá el código y necesita ser arreglado de inmediato

- Vulnerability: parte del código que hace que el software sea vulnerable a ataques y que tambien debe ser arreglado de inmediato

- Security Hotspot: parte del código vulnerable a ataques pero que, al no comprometer la seguridad de todo el software, no necesita un arreglo inmediato

- Code smells: no es un error, pero hace que el código sea confuso y difícil de mantener. Se pueden incluir aquí: patrones de código que son difíciles de entender o mantener, código duplicado, métodos o funciones demasiado largos o complejos, nombres de variables o funciones confusos o poco descriptivos, y otras características

- Cobertura: parte del código que no está cubierta, es decir, que no esta siendo probada por test unitarios. Sonarqube no detecta esto, pero es capaz de leer los reportes de herramientas que si lo hacen ej: Jacoco en java o Istanbul en JavaScript

- Duplicaciones: número de bloques de código o archivos duplicados. Tener estas duplicaciones dificulta la mantenibilidad y lectura del código

- Deuda técnica: es un concepto que refleja el futuro trabajo adicional que va a llevar algo al poder elegir una solución rápida pero limitada, en vez de una costosa

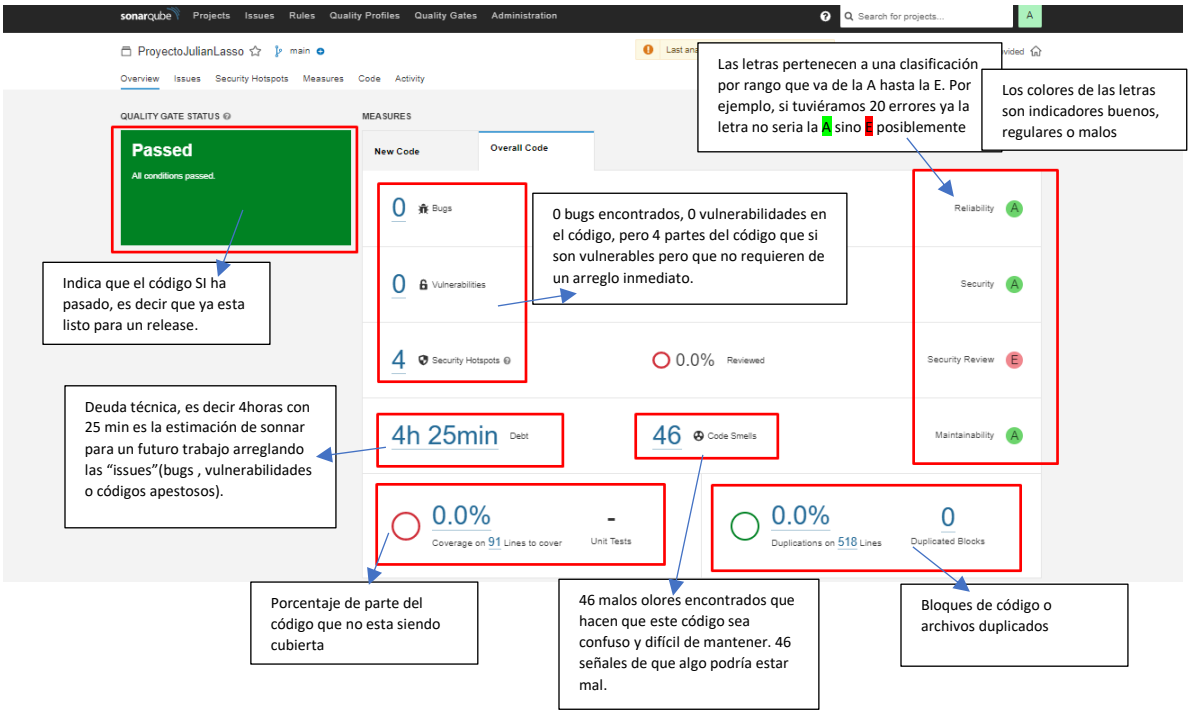
pero mejor. Sonarqube hace una estimación sobre el tiempo que se va a tardar en arreglar todas las “issues” que se han detectado en el análisis.

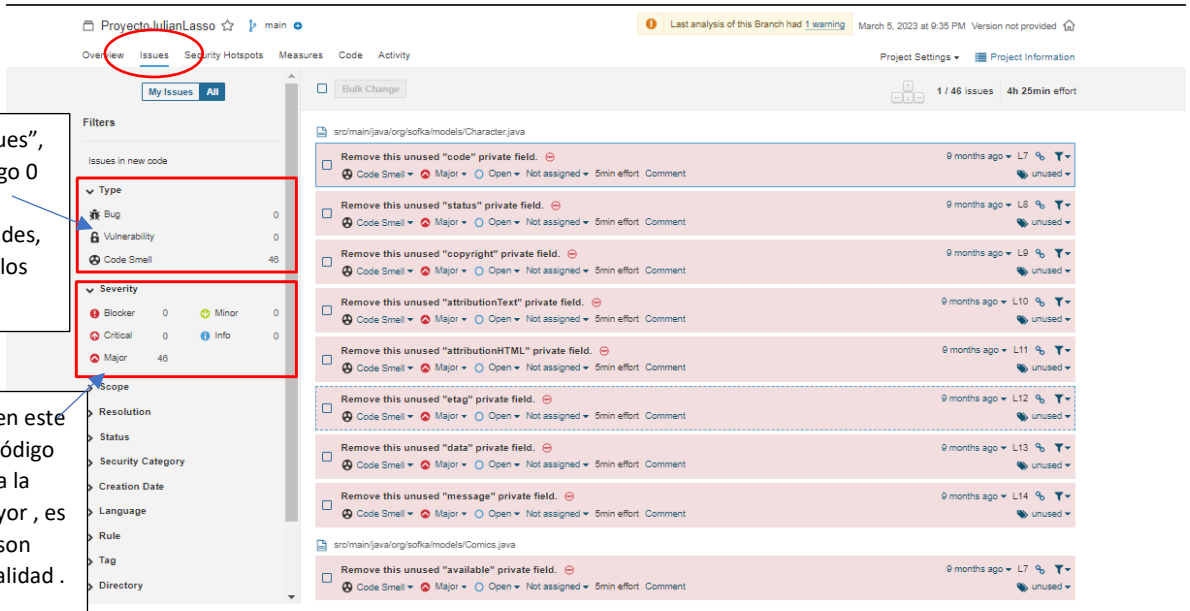
- Quality gate: Conjunto de métricas que tiene cumplir el código para ser considerado “listo para un reléase”. Sonarqube tiene su propio quality gates que se aplica por defecto, pero se pueden crear nuevos con cierto número de condiciones.

- Las severidades que podemos encontrar son: bloqueador (alta probabilidad de afectar el comportamiento de nuestra aplicación en producción), critico (afecta el comportamiento o es un problema que representa una falla de seguridad), mayor (defecto de calidad ejemplo bloques duplicados o parámetros no utilizados), menor (defecto de calidad) y de información (hace referencia a un hallazgo).

- Quality profile: conjunto de reglas y configuraciones que se aplican a un proyecto de software con el fin de evaluar la calidad del código y detectar posibles problemas. Puede incluir diferentes tipos de reglas, como reglas de seguridad, reglas de mantenibilidad, reglas de rendimiento, etc. Los desarrolladores pueden personalizar el Quality Profile para adaptarlo a las necesidades específicas de su proyecto y ajustar el nivel de severidad de las reglas según sus objetivos

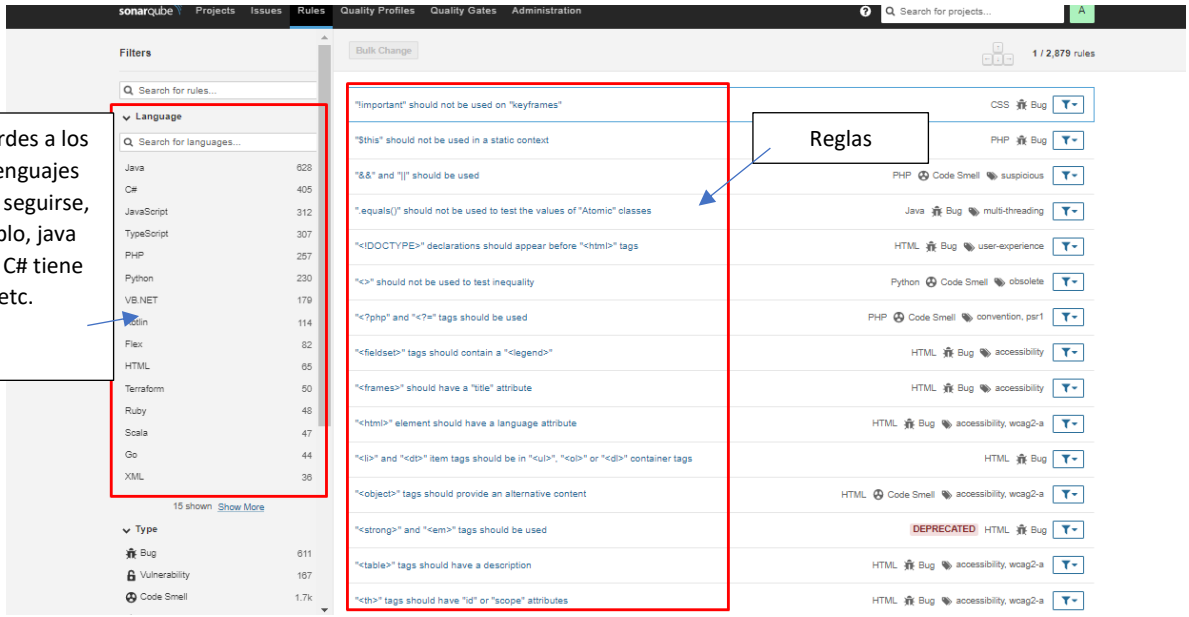
- Rules: Estándares o prácticas de código que se deben seguir y determinan la severidad de los posibles bugs, vulnerabilidades, etc. Estas reglas vienen dadas por plugins asociados a cada lenguaje, que a su vez proporciona “quality profiles” por defecto.
  Se ejecutan en el código fuente para generar los reportes





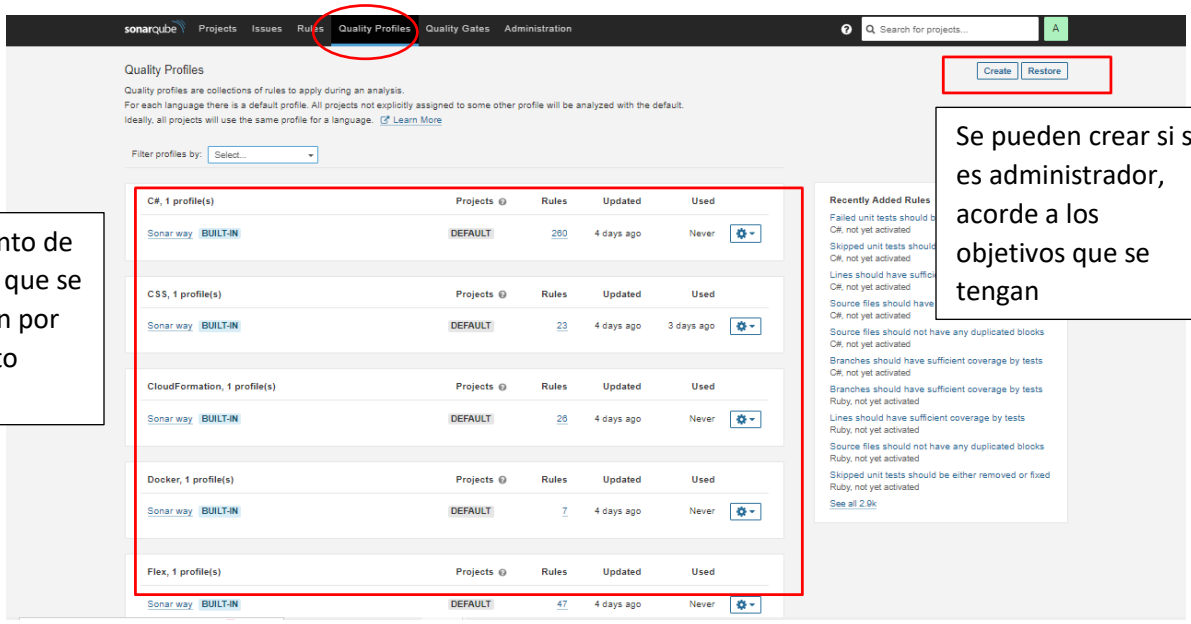
Tipos de "issues", en este código 0 bugs, 0 vulnerabilidades, pero 46 malos olores

Severidades, en este caso , en el código predomina la severidad mayor , es decir que son defectos de calidad .



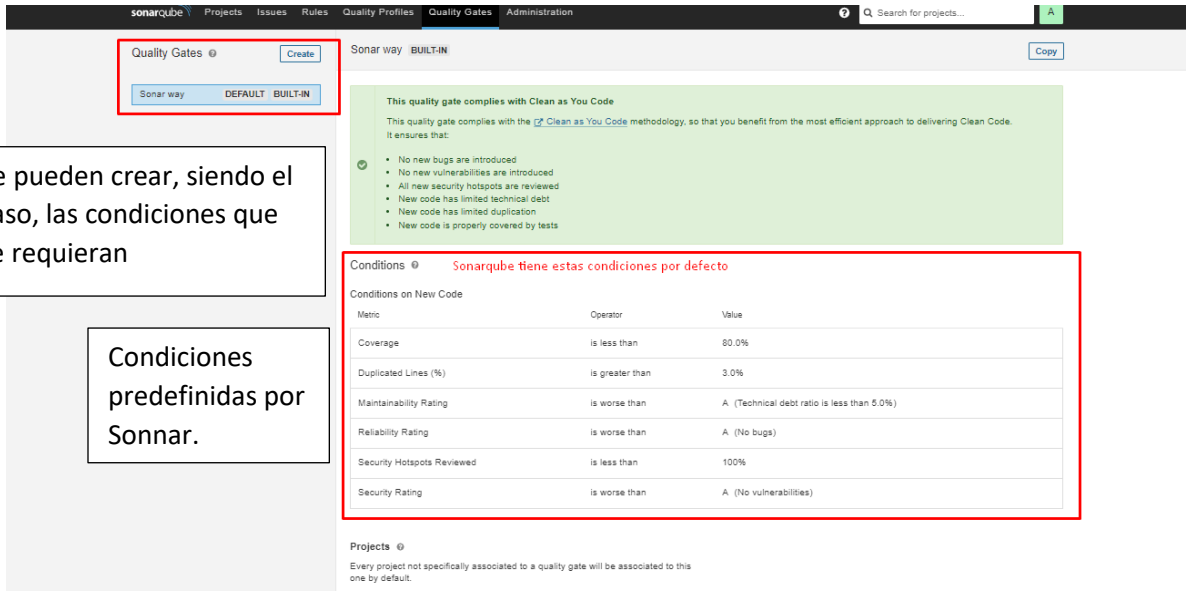
Reglas acordes a los tipos de lenguajes que deben seguirse, por ejemplo, java tiene 628, C# tiene 405, etc.

Reglas



Conjunto de reglas que se aplican por defecto

Se pueden crear si se es administrador, acorde a los objetivos que se tengan



Se pueden crear, siendo el caso, las condiciones que se requieran

Condiciones predefinidas por Sonnar.