



# Sonarqube

## *Análisis estático del código*

Luis Ramón Álvarez

Version 2.0.0 2020-11-15

# Contenidos

1. Sonarqube .....	1
1.1. Funcionalidades .....	1
1.1.1. Quality Gates .....	1
1.1.2. Security Analysis (Hotspots) .....	1
1.1.3. Security Analysis (Vulnerabilities) .....	2
1.2. ¿Qué lenguajes de programación soporta? .....	3
1.3. ¿Qué produce el proceso de análisis? .....	3
1.4. ¿Se analizarán todos los archivos de nuestro proyecto? .....	3
1.5. ¿Podemos tratar el análisis de ramas "pull request"? .....	4
1.6. ¿Qué sucede durante el proceso de análisis? .....	4
1.7. Lab: Instalación de SonarQube (Docker Compose) .....	5
1.7.1. Ajuste en el servidor .....	5
1.7.2. Creación del archivo de orquestación .....	5
1.7.3. Configurando el primer acceso como administrador .....	7
2. Deuda Técnica .....	9
2.1. ¿Qué es la deuda técnica? .....	9
2.2. ¿Cómo podemos medir la deuda técnica? .....	10
2.3. ¿Qué nivel de deuda técnica nos podemos permitir? .....	10
2.4. Control de la deuda técnica en SonarQube .....	11
2.5. Sobre el tiempo necesario para resolver la deuda técnica .....	12
2.6. Seguimiento de la deuda técnica .....	13
2.7. Moraleja y consejo estrella .....	15
2.8. Lab: Creación de aplicación Java Maven base .....	15
2.8.1. Creación de la carpeta para proyectos .....	15
2.8.2. Creando el proyecto .....	15
3. SonarScanner .....	16
3.1. Lab: SonarScanner con un proyecto de Java Maven .....	16
3.1.1. Localizando y abriendo el archivo de configuración de Maven (settings.xml) .....	16
3.1.2. Configurando el plugin .....	16
3.1.3. Creando y configurando el profile .....	17
3.1.4. Obteniendo token de seguridad .....	17
3.1.5. Analizando el proyecto (Ejecución directa mediante el goal) .....	19
3.1.6. Analizando el proyecto (Regulando propiedades de análisis) .....	20
4. Administración de usuario, grupos y roles .....	22
4.1. Autenticación en SonarQube .....	22
4.2. Autenticación de endpoints API .....	22
4.3. Mecanismos de autenticación .....	22
4.4. Tokens de acceso a SonarQube .....	23

4.5. Credenciales por defecto .....	23
4.6. Gestión de la Autorización en SonarQube .....	23
4.7. Gestión de usuarios en SonarQube .....	24
4.8. Gestión de grupos en SonarQube .....	24
4.9. Permisos globales en SonarQube .....	24
4.10. Permisos de un proyecto .....	25
4.11. Plantillas de permisos para permisos por defecto .....	26
4.12. Permisos de creadores .....	27
4.13. Reseteo de permisos de un proyecto .....	27
4.14. Lab: Administración de usuarios, grupos y roles .....	27
4.14.1. Creando los proyectos .....	28
4.14.2. Creando los grupos .....	29
4.14.3. Creando los usuarios .....	30
4.14.4. Asignando los usuarios a los grupos .....	32
4.14.5. Asignando los grupos a los proyectos .....	34
4.14.6. Generando tokens de acceso a los usuarios .....	36
4.14.7. Enviando solicitud de análisis a SonarQube .....	38
5. Plugins de integración de SonarQube con Eclipse .....	39
5.1. Lab: Integración de plugins SonarQube con Eclipse .....	39
5.1.1. Instalación de plugin SonarLint .....	39
5.1.2. Solicitud de análisis en el proyecto .....	40
5.1.3. Configuración en modo conectado .....	42
5.1.4. Configurando la conexión .....	43
5.1.5. Creando un nuevo proyecto en SonarQube .....	52
5.1.6. Ajuste el pom.xml (Sonar Properties) .....	54
5.1.7. Configuración de la fase de Maven para Sonar .....	54
5.1.8. Enviando análisis a SonarQube .....	56
6. Quality Gates .....	59
6.1. Utilizando la mejor configuración de Quality Gate .....	59
6.2. Quality Gate recomendada .....	60
6.3. Posibilidad de notificaciones .....	60
6.4. Sobre la seguridad de acceso .....	60
6.5. Lab: Quality Gates .....	60
6.5.1. Creando una nueva QualityGate .....	61
6.5.2. Asignando permisos .....	61
6.5.3. Asignando condiciones .....	61
6.5.4. Asignando la QualityGate al proyecto .....	62
6.5.5. Probando la QualityGate .....	63
7. Quality Profiles .....	65
7.1. Perfiles de calidad por defecto .....	65
7.2. Permisos en los perfiles de calidad .....	66

7.3. Lab: Quality Profiles .....	66
7.3.1. Creando un nuevo QualityProfile .....	66
7.3.2. Asignando reglas .....	67
7.3.3. Vinculando el QualityProfile con el proyecto .....	68
7.3.4. Realizando ajustes en nuestro proyecto .....	70
7.3.5. Ejecutando análisis contra SonarQube .....	73

# Capítulo 1. Sonarqube

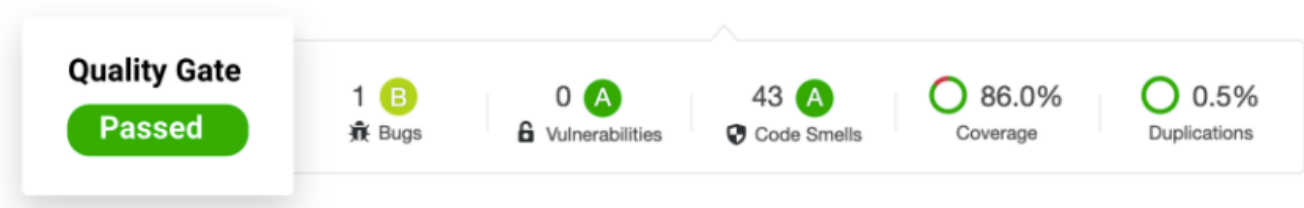
Consiste en una plataforma de evaluación de código fuente.

Se trata de un Software Libre que usa distintas herramientas de análisis estático de código fuente para llegar a obtener distintas métricas que nos pueden ayudar a mejorar la calidad de nuestro código fuente.

## 1.1. Funcionalidades

Entre las distintas funcionalidades que nos ofrece la plataforma podemos encontrar:

### 1.1.1. Quality Gates



- Permite disponer de información en cada análisis realizado de nuestro código para poder comprobar cuando este se encuentra listo para proceder a su lanzamiento en producción.
- Las Quality Gates pueden tener dos estados, **Passed** o **Failed**, dependiendo de si hemos pasado las premisas de calidad según el lenguaje de programación con el que nos encontremos.

### 1.1.2. Security Analysis (Hotspots)



- Permite informar sobre los llamados **Hotspots** (puntos de seguridad vulnerables).
- En principio se puede dar el caso de que esta información sea correcta, pero para descartar un falso positivo, se requiere por parte del sistema intervención humana.
- Conforme los programadores interactúan con los **Hotspots** aprenden a evaluar los riesgos de seguridad mientras al mismo tiempo, aprenden más sobre las prácticas de codificación segura.

Make sure that hashing data is safe here.

Add Comment

Get Permalink

Category Weak Cryptography

Review priority MEDIUM

Assignee Duarte Meneses

Status: To review

This Security Hotspot needs to be reviewed to assess whether the code poses a risk.

SonarQube server/.../sonar/ce/task/projectanalysis/source/PersistFileSourcesStep.java

```
105
106 private void persistSource(FileSourceDataComputer.Data fileSourceData, Component file) {
107     DbFileSources.Data lineData = fileSourceData.getLineData();
108
109     byte[] binaryData = FileSourceDto.encodeSourceData(lineData);
110     String dataHash = DigestUtils.md5Hex(binaryData);
111     String srcHash = fileSourceData.getSrcHash();
112     List<String> lineHashes = fileSourceData.getLineHashes();
113     Changeset latestChangeWithRevision = fileSourceData.getLatestChangeWithRevision();
114     int lineHashesVersion = sourceLinesHash.getLineHashesVersion(file);
115     FileSourceDto previousDto = previousFileSourcesByUuid.get(file.getUuid());
```

What's the risk?	Are you at risk?	How can you fix it?
<p>Hashing data is security-sensitive. It has led in the past to the following vulnerabilities:</p> <ul style="list-style-type: none"><li>• <a href="#">CVE-2018-9233</a></li><li>• <a href="#">CVE-2013-5097</a></li><li>• <a href="#">CVE-2007-1051</a></li></ul> <p>Cryptographic hash functions are used to uniquely identify information without storing their original form. When not done properly, an attacker can steal the original information by guessing it (ex: with a <a href="#">rainbow table</a>), or replace the original data with another one having the same hash.</p> <p>This rule flags code that initiates hashing.</p>		

### 1.1.3. Security Analysis (Vulnerabilities)



- La detección de vulnerabilidades en el software requiere de una acción inmediata.
- SonarQube nos va a proporcionar descripciones detalladas y aspectos destacados donde nos explica por qué el fragmento de código se encuentra en riesgo.
- Con la ayuda que nos proporciona la plataforma, simplemente debemos de seguir las instrucciones que se nos indican para corregir dichas vulnerabilidades.



## 1.2. ¿Qué lenguajes de programación soporta?

Actualmente soporta más de 27 lenguajes de programación diferentes, dependiendo de la edición que dispongamos.

## 1.3. ¿Qué produce el proceso de análisis?

El resultado del análisis de SonarQube resultará en medidas y problemas de calidad (casos en los que se rompieron las reglas de programación).

Debemos de tener en cuenta que dependiendo del lenguaje de programación a escanear, quizás podamos obtener algunas diferencias en los análisis:

- En todos los lenguajes, los resultados de los análisis a modo de "problemas" se obtendrán automáticamente del código fuente que esté alojado en proveedores de control de versiones como Git o SVN, ambos proveedores de control de versiones son compatibles con SonarQube.
- En todos los lenguajes, el análisis de código estático es realizado (Archivos Java, COBOL, etc.)
- El análisis estático de código compilado puede ser realizado para ciertos lenguajes (.class en Java, .dll en C#, etc.)

## 1.4. ¿Se analizarán todos los archivos de nuestro proyecto?

De forma predeterminada, sólo los archivos que son reconocidos por nuestra edición de SonarQube son cargados en el proyecto durante el análisis.

Por ejemplo, si estamos utilizando **SonarQube Community Edition**, la cual incluye análisis de lenguajes como Java, JavaScript, podremos realizar análisis de esos lenguajes de programación, pero sin embargo, no podremos realizar análisis de lenguaje C++.

Los archivos .java y .js serán cargados para su análisis, pero los archivos con extensión .cpp serán ignorados para los análisis.

## 1.5. ¿Podemos tratar el análisis de ramas "pull request"?

Si, con la versión **Developer Edition** se añade la posibilidad de analizar las ramas de nuestro proyecto así como las pull request, de forma que se generen informes automáticos en la plataforma.

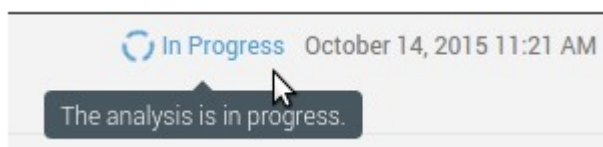
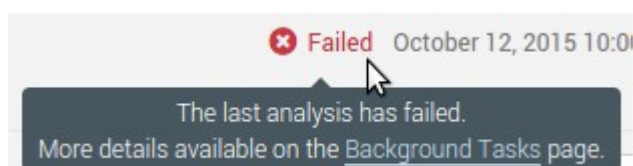
## 1.6. ¿Qué sucede durante el proceso de análisis?

Durante el proceso de análisis, se solicitan datos del servidor, se analizan los archivos proporcionados para el análisis y los datos resultantes se envían de vuelta al servidor al final en forma de informe, que luego se analizan de forma asíncrona en el propio servidor de SonarQube.

Los informes de análisis se ponen en cola y se procesan secuencialmente, por lo que es muy posible que durante un breve período de tiempo después SonarQube no muestre los resultados hasta finalizar el proceso.

Sin embargo, si que podremos saber que está pasando por que se añadirá un icono en la página de inicio del proyecto a la derecha del nombre del proyecto.

Si ponemos el ratón por encima, nos indicará más detalles.





## 1.7. Lab: Instalación de SonarQube (Docker Compose)

Mediante este laboratorio vamos a llevar a cabo la instalación del sistema SonarQube.

Adicionalmente a la instalación de la propia plataforma SonarQube, instalaremos una base de datos anexa, con la idea de que se persista toda la información sobre estadísticas/métricas que el Sonar pueda llegar a generar.

Utilizaremos la tecnología de Docker, ayudados del orquestador Docker Compose para llevar a cabo dicha instalación.

### 1.7.1. Ajuste en el servidor

En el servidor donde vayamos a ejecutar las plataformas del Sonar y su correspondiente motor de base de datos, debemos de crear el fichero en la siguiente ruta: `/etc/sysctl.d/sonar.conf` y añadirle dentro lo siguiente:

```
vm.max_map_count=262144
```

A continuación, reiniciamos la máquina para que los cambios entren en operación y de camino, testear que en el nuevo arranque de la máquina los cambios han sido cargados correctamente.

### 1.7.2. Creación del archivo de orquestación

Para proceder a poner en marcha los dos sistemas que necesitamos, vamos a crear un archivo con nombre **docker-compose.yml** y le añadimos el siguiente contenido:

```

version: '3.8'

services:
  sonarqube-postgresql:
    image: postgres:12
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
    volumes:
      - sonarqube_postgresql_db:/var/lib/postgresql
      - sonarqube_postgresql_data:/var/lib/postgresql/data
    restart: always

  sonarqube:
    image: sonarqube:community
    environment:
      - SONAR_JDBC_URL=jdbc:postgresql://sonarqube-postgresql:5432/sonar
      - SONAR_JDBC_USERNAME=sonar
      - SONAR_JDBC_PASSWORD=sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
    ports:
      - "9002:9000"
    restart: always
    depends_on:
      - sonarqube-postgresql

volumes:
  sonarqube_data:
    name: sonarqube_data

  sonarqube_logs:
    name: sonarqube_logs

  sonarqube_extensions:
    name: sonarqube_extensions

  sonarqube_postgresql_db:
    name: sonarqube_postgresql_db

  sonarqube_postgresql_data:
    name: sonarqube_postgresql_data

```

A continuación, ejecutamos en el directorio donde se encuentre nuestro archivo de orquestación **docker-compose.yml** la siguiente instrucción:

```
$ docker-compose up -d
```

Esta orden llevará a cabo tanto la descarga como arranque de las dos plataformas.

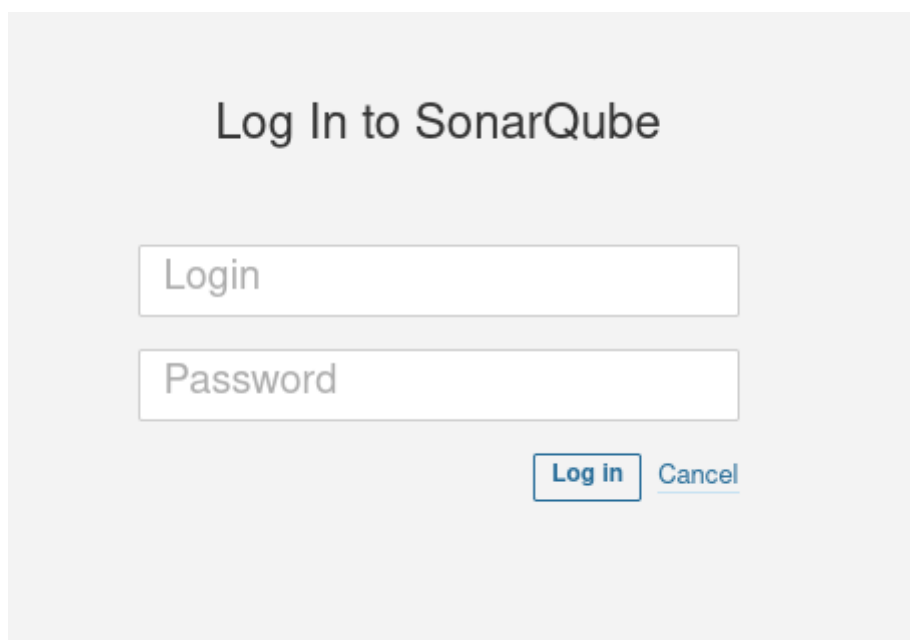


La primera vez que arranquemos los sistemas, debemos de ser un poco pacientes, ya que el proceso puede demorarse algunos minutos.

### 1.7.3. Configurando el primer acceso como administrador

Una vez finalice el procedimiento de instalación, deberíamos de tener nuestro servidor SonarQube funcionando.

Accedemos a la siguiente url: <http://localhost:9002> y nos aparecerá una pantalla para indicar el usuario y contraseña de la cuenta de administrador para llevar a cabo el acceso inicial:



The image shows the 'Log In to SonarQube' login page. It features a title 'Log In to SonarQube' at the top. Below the title are two input fields: 'Login' and 'Password'. At the bottom right, there are two buttons: 'Log in' and 'Cancel'.

Introducimos las siguientes credenciales por defecto:

- **Usuario**
  - admin
- **Contraseña**
  - admin

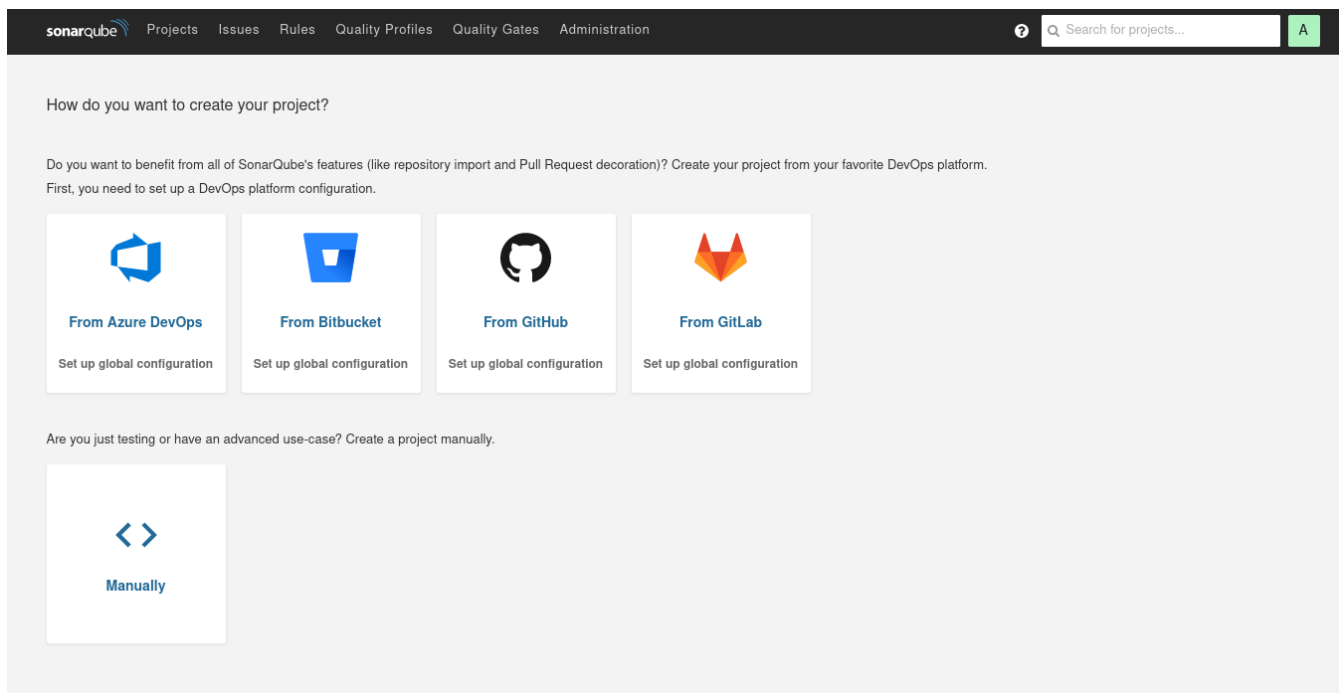
El sistema por seguridad nos pedirá que actualicemos las credenciales por defecto de administrador.

Indicamos como nuevas credenciales las siguientes:

- **Usuario**
  - admin
- **Contraseña**

- AdminAdmin3

Una vez estemos dentro, veremos un dashboard como se indica en la imagen:



# Capítulo 2. Deuda Técnica

Cuando oigamos el término de **deuda**, ya podemos imaginarnos que algo bueno no debe de ser :p

Si no la tratamos a tiempo, afectará de alguna u otra forma a nuestros proyectos.

## 2.1. ¿Qué es la deuda técnica?

Para entender bien el concepto de la **deuda técnica** vamos a realizar una comparativa con un concepto que en teoría todo el mundo conoce bastante bien y es la **deuda económica**.

En ocasiones necesitamos una elevada cantidad de dinero para adquirir ciertos bienes como podrían ser:

- Compra de una vivienda
- Creación de una empresa
- Etc...

En esas ocasiones, necesitas pedir dicha cantidad para que te la presten, y posteriormente, devolverla en pequeños pagos durante una cantidad de tiempo pactada (varios meses, años, etc.) Ya que, de otra forma, sería complicado haber reunido el dinero suficiente a tiempo para poder hacer frente a esas compras.

Teniendo en cuenta esta premisa, el término de deuda técnica hace referencia a cuando necesitamos, en este caso, una cantidad de tiempo (en el fondo el tiempo es dinero), para poder entregar el trabajo acordado en la fecha pactada.

En nuestros proyectos esto ocurre por ejemplo en situaciones cuando:

- Copiamos código en lugar de crear una nueva clase que realice la función que necesitamos (Duplicamos código)
- Duplicamos literales en lugar de usar constantes (Magic Numbers)

Cuando sucede esto, **lo que estamos haciendo es pedir tiempo para entregar antes el código**, pero igual que en la deuda económica, ese tiempo tendremos que devolverlo, ya que de no hacerlo, tendremos muchísimos problemas en forma de errores en el proyecto, o en el mejor de los casos, de invertir posteriormente el tiempo necesario para que el código siga las buenas prácticas.

Adicionalmente a esto, otra cosa en lo que se parece la deuda técnica con la económica, es que se tiene que **pagar con intereses**.

- **En la deuda económica**
  - Se paga un importe adicional para costear un préstamo
- **En la deuda técnica**
  - Los continuos cambios de contexto en el código fuente nos van a hacer que se tarde más en resolver un **code smell** (código que huele mal), que si se hubiera hecho correctamente desde el principio.

- **¿Con qué no está relacionada directamente**
  - Con la fiabilidad o seguridad de nuestro proyecto, ya que ello son errores que ocurren en nuestro código, principalmente por desconocimiento ante desarrollos que debemos de realizar.
- **¿Con qué está relacionada directamente?**
  - Con la mantenibilidad, es decir, con la facilidad para que podamos actualizar o hacer evolucionar nuestros proyectos.

## 2.2. ¿Cómo podemos medir la deuda técnica?

Como hemos comentado anteriormente, en la deuda técnica, lo que se pide prestado es tiempo, así que lo que debemos de hacer es **devolver tiempo**.

También sabemos que la deuda técnica afecta directamente a la mantenibilidad del proyecto, y es por eso que podemos ver la primera medida en la que SonarQube nos puede ayudar con la deuda técnica, ya que cuando detecta algún **code smell**, nos indica **el tiempo estimado para solucionarlo**

1 / 6,858 issues

.../apache/struts2/showcase/AsyncTest.java

**Remove this use of "Thread.sleep()".**  
Code Smell

...e/struts2/showcase/StaticContentTest.java

**Refactor the body of this try/catch to have only one invocation possibly throwing a runtime exception.**  
Code Smell +2

**Refactor the body of this try/catch to have only one invocation possibly throwing a runtime exception.**  
Code Smell +2

...ts2/osgi/admin/actions/BundlesAction.java

**Remove this unused import 'java.util.Comparator'.**  
Code Smell

28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45 ...

```
import com.gargoylesoftware.htmlunit.html.*;
import com.gargoylesoftware.htmlunit.html.*;
import com.gargoylesoftware.htmlunit.html.*;

public class AsyncTest {
    @Test
    public void testChatRoom() throws Exception {
        try (final WebClient webClient = new WebClient()) {
            final HtmlPage page = webClient.getPage(ParameterUtils.getBaseUrl() + "/async/index.html");

            final HtmlForm form = page.getForms().get(0);

            final HtmlTextInput textField = form.getInputByName("msg");
            textField.type("hello");

            final HtmlSubmitInput button = form.getInputByValue("Send");
            final HtmlPage page2 = button.click();

            Thread.sleep(4000);
        }
    }
}
```

**Remove this use of "Thread.sleep()". Why is this an issue?** 3 years ago L45  
Code Smell Major Open Not assigned 20min effort bad-practice, tests

46 ...  
47  
48

**"Thread.sleep" should not be used in tests** java:S2925

Code Smell Major Main sources bad-practice, tests Available Since Oct 02, 2015 SonarQube (Java) Constant/issue: 20min

Using `Thread.sleep` in a test is just generally a bad idea. It creates brittle tests that can fail unpredictably depending on environment ("Passes on my machine!") or load. Don't rely on timing (use mocks) or use libraries such as `Awaitility` for asynchronous testing.

## 2.3. ¿Qué nivel de deuda técnica nos podemos permitir?

Es importante ser consciente que ese tiempo depende en gran medida de:

- La experiencia del programador
- Puede verse afectado por las particularidades del proyecto
- La organización que tengamos en el mismo

- Incluso puede depender su infraestructura, tecnología que utilizamos para llevar a cabo el despliegue
- Etc.

Así que, aunque no es una información exacta, puede servir como una primera aproximación.

Pero, por otro lado, saber el tiempo que **debemos**, no es suficiente para saber el nivel de deuda técnica del proyecto.

Para entender esto, vamos a pensar en el siguiente ejemplo:

No es lo mismo que una persona deba 5.000€ en un préstamo para la compra de un coche cuyo precio son 10.000€, a que una organización deba 5.000€ en un préstamo para comprar unas oficinas que cuestan 500.000€.

En el primer caso, la deuda es la mitad del bien (el coche) y en algún caso, si hay un imprevisto (por ejemplo, que el endeudado quedara sin trabajo) la deuda pendiente supondría un problema.

Mientras que en el segundo caso, la deuda está prácticamente pagada y no debería ocasionar ningún contratiempo a la organización.

Es por ello, que la **deuda técnica debe relacionarse con el tiempo total del desarrollo del proyecto** (lo que equivaldría al precio total del bien), y de esta forma mostrar también un porcentaje.

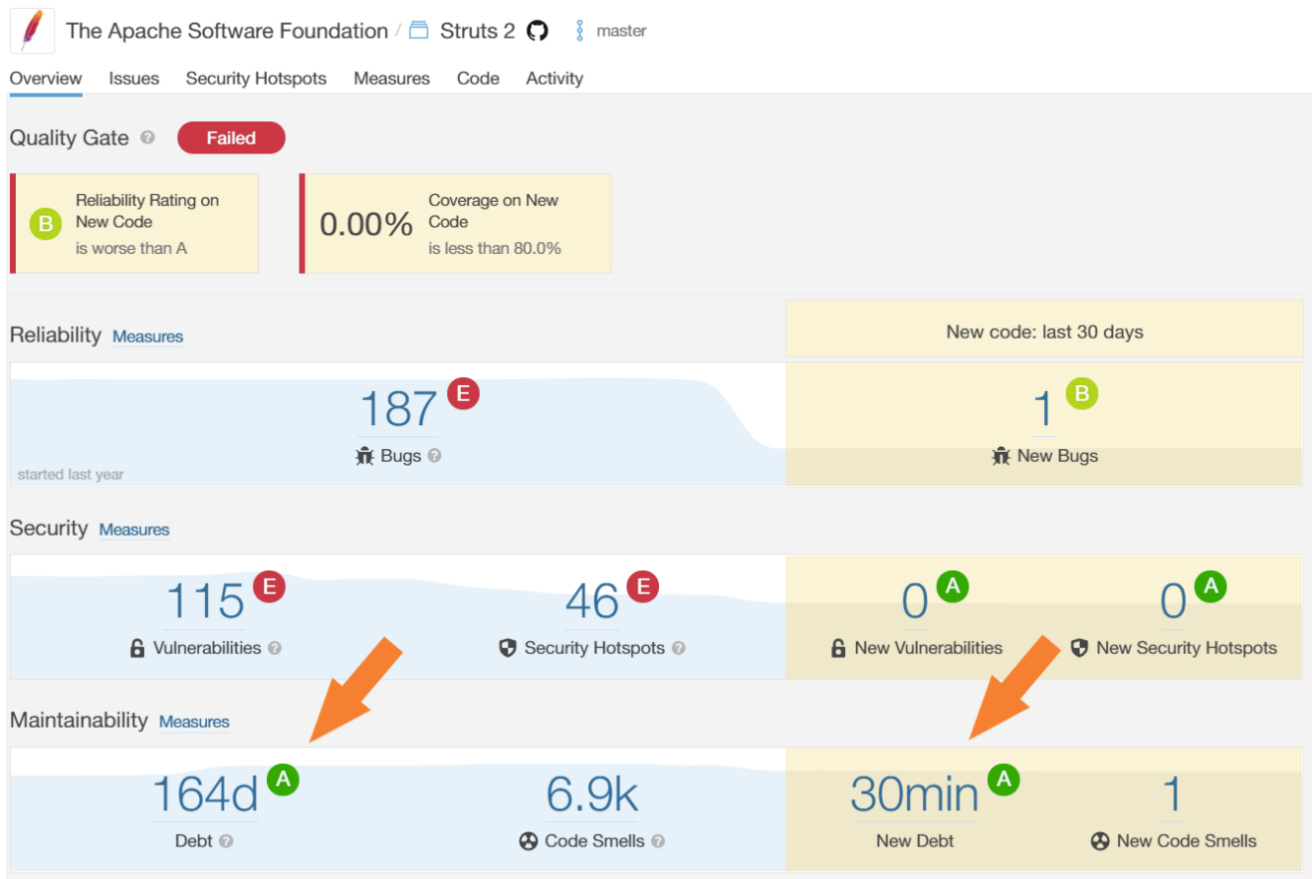
Por ello, cuando hablamos de una deuda técnica del 10%, quiere decir que, si el proyecto se ha desarrollado en unas 1.000 horas, el tiempo de corregir las evidencias relacionadas con la mantenibilidad (**code smells**) es aproximadamente unas 100 horas.

## 2.4. Control de la deuda técnica en SonarQube

Como estamos observando, SonarQube le da una gran importancia al seguimiento de la deuda técnica.

Tanto es así que lo utiliza como uno de sus tres principales indicadores que son:

- La fiabilidad
- La seguridad
- La mantenibilidad



En el caso de la calificación global de mantenibilidad, SonarQube mostrará el resultado en función de la deuda técnica obtenida.

Las calificaciones globales van desde A (mejor calificación), hasta la letra E (peor calificación), y sobre la mantenibilidad, SonarQube puede mostrar los siguientes resultados:

- **A**
  - Deuda técnica inferior al 5%
- **B**
  - Deuda técnica entre 6% y 10%
- **C**
  - Deuda técnica entre 11% y 20%
- **D**
  - Deuda técnica entre 21% y 50%
- **E**
  - Deuda técnica superior superior al 50%

## 2.5. Sobre el tiempo necesario para resolver la deuda técnica

SonarQube muestra el tiempo total que suman todos los code smells del proyecto.



En la imagen anterior, podemos ver que en el nuevo código (últimos 30 días) existe 1 code smell, cuyo tiempo aproximado de solución es de 30 minutos, así que la deuda técnica del código nuevo será ese tiempo: 30 minutos.

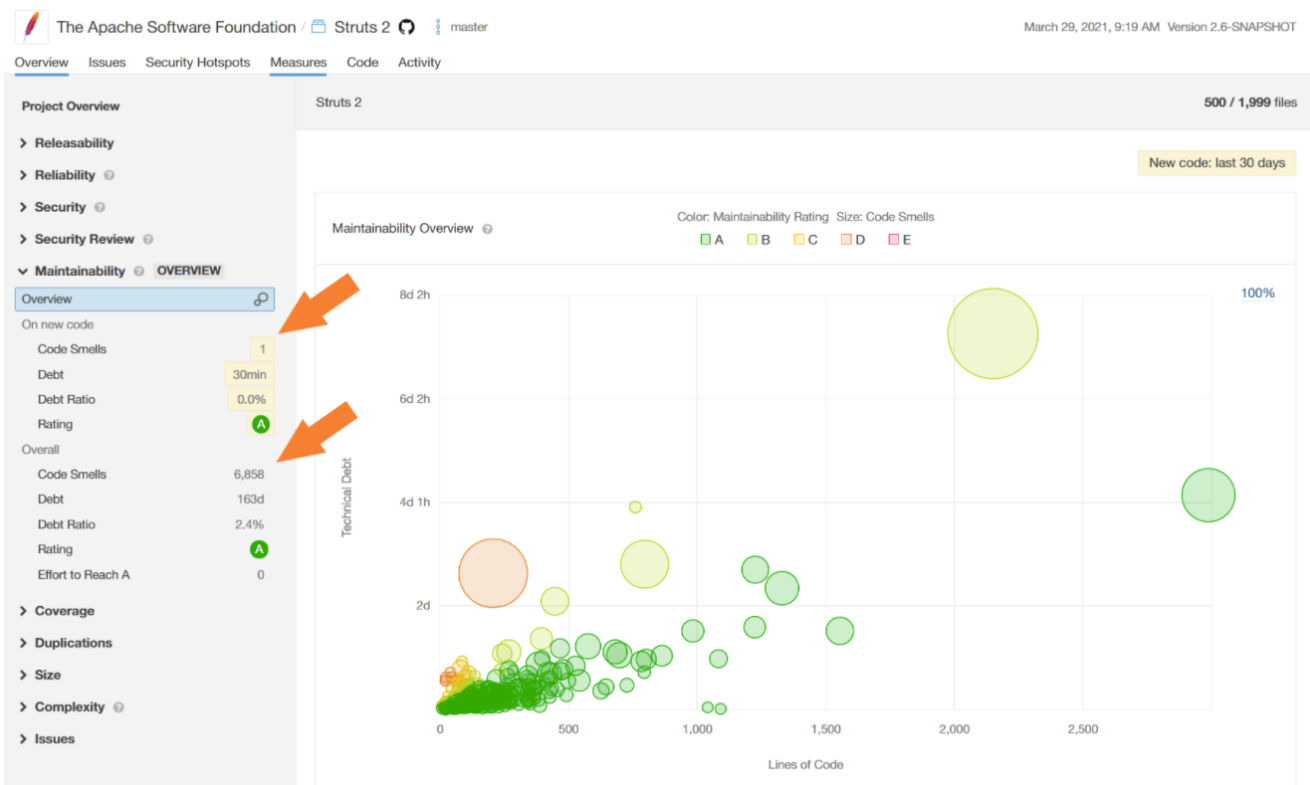
Mientras que en el proyecto completo, existen más de 6.900 code smells, y el tiempo total de solucionarlos será aproximadamente de 164 días.

Hay que tener en cuenta que Sonarqube almacena la deuda técnica en minutos, pero la puede mostrar en horas o en días (1 día = 8 horas de trabajo).

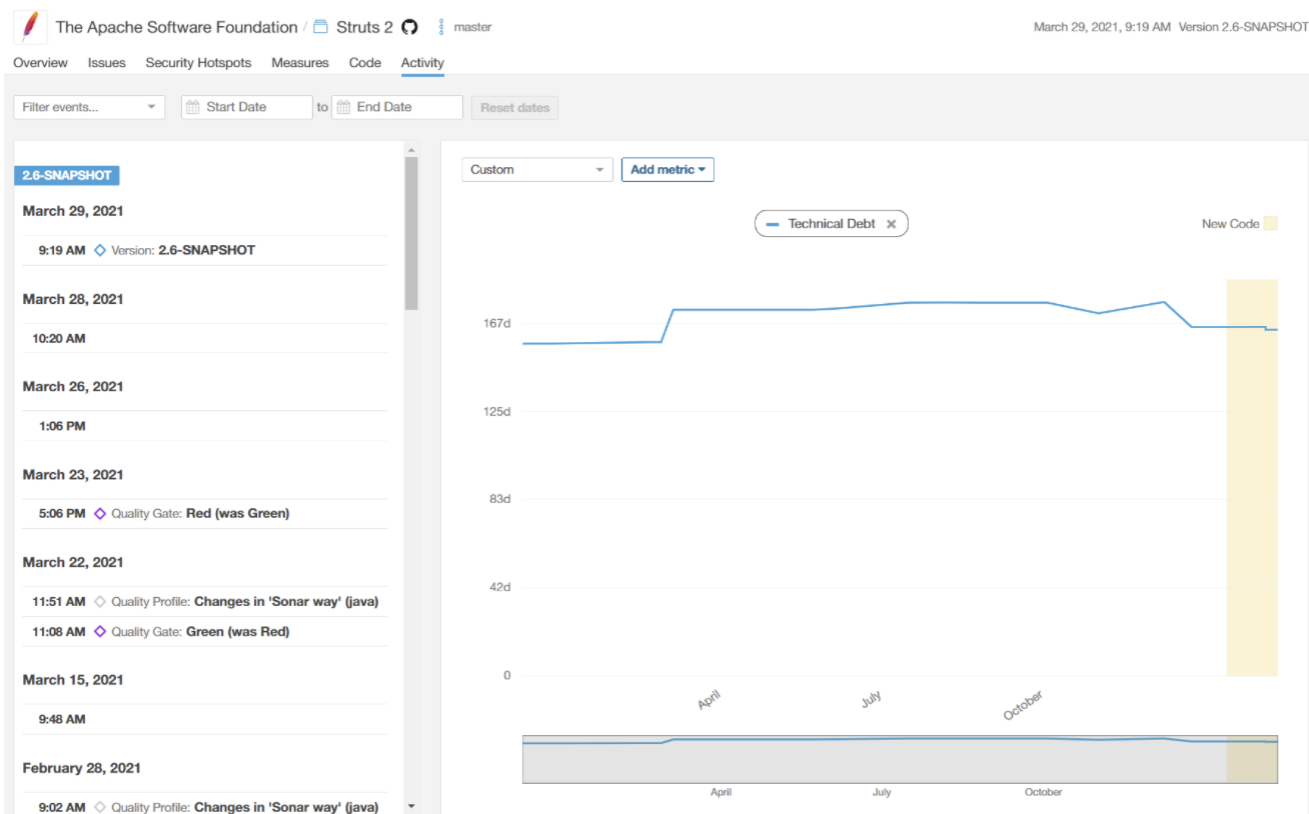
## 2.6. Seguimiento de la deuda técnica

Para hacer el seguimiento de la deuda técnica en SonarQube, tenemos distintas opciones.

Por un lado, en la sección de **Measures** (medidas), podemos ver la deuda técnica de todo el código, o únicamente la del nuevo.

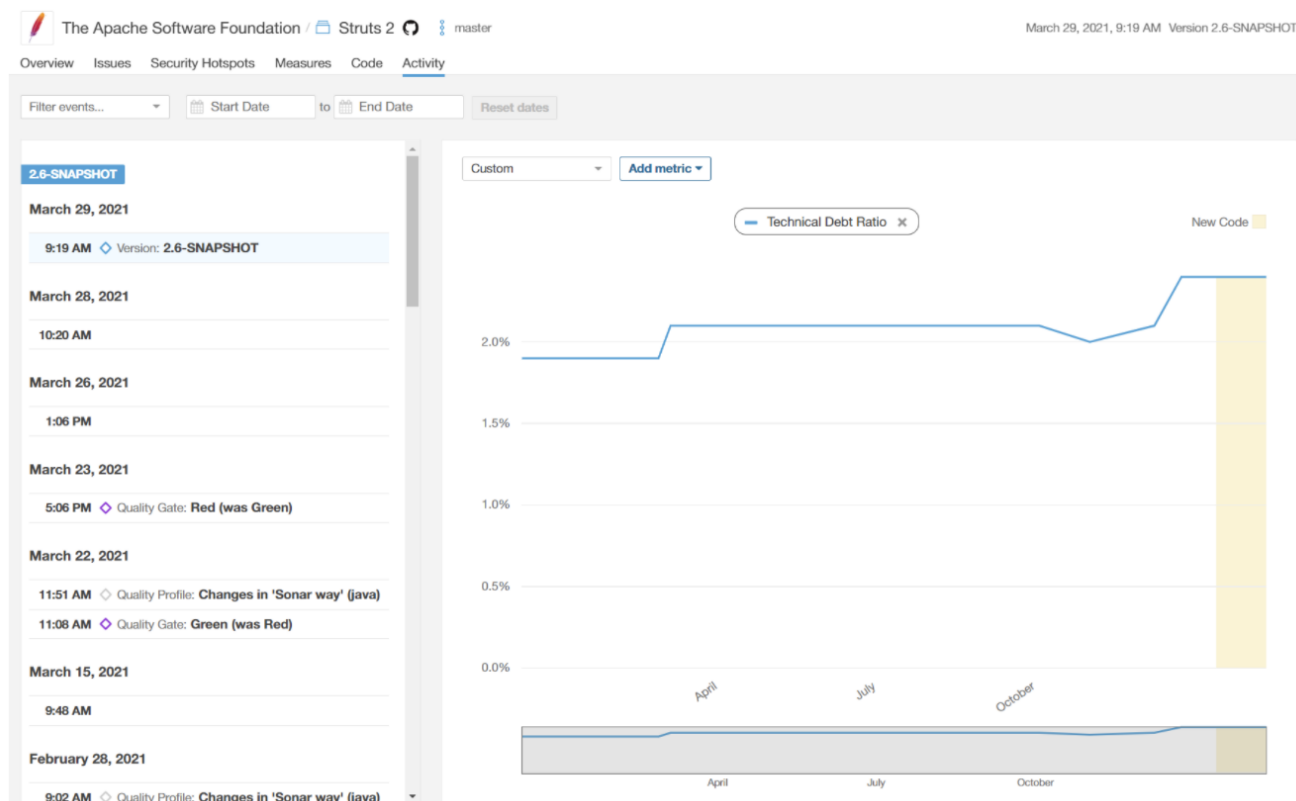


Por otro lado, en la sección de **Activity** (actividad), podemos ver el historial del resultado de la deuda técnica a lo largo de la vida del proyecto.



De esta forma, se puede observar la evolución de nuestra deuda técnica aunque como hemos dicho antes, únicamente en días.

Si realmente queremos ver lo que supone para el proyecto, deberíamos comprobar el ratio de deuda técnica, y de esta forma, podremos ver mejor su evolución, independientemente de que el proyecto haya crecido en número de líneas.



## 2.7. Moraleja y consejo estrella

Intenta siempre, como en la vida económica, no endeudarte mucho, ya que en el futuro tendrás que pagarlo... ¡¡¡Y con intereses :D!!!

## 2.8. Lab: Creación de aplicación Java Maven base

Mediante este laboratorio, crearemos una aplicación base de Java - Maven con la que llevaremos a cabo diferentes análisis mediante el sistema SonarQube.

### 2.8.1. Creación de la carpeta para proyectos

Creemos el directorio **projects** en la ruta que mejor nos convenga para nuestros proyectos Java, esta carpeta será nuestro workspace

### 2.8.2. Creando el proyecto

Vamos a crear una carpeta para el proyecto java que vamos a crear, en nuestro caso, a dicha carpeta la llamaremos **app-backend**

Creemos la carpeta **projects/app-backend/**

Vamos a crear nuestro módulo maven para trabajar con el, nos situamos en nuestra carpeta **app-backend**, que acabamos de crear y ejecutamos la siguiente instrucción:

```
$ mvn archetype:generate -DgroupId=com.tutorial.sonarqube -DartifactId=sonarqube-maven-project
```

- Indicará que elijamos el formato a aplicar, en principio dejamos por defecto la sugerencia que nos indique, ya que será el formato básico de maven
- Indicamos la versión del arquetipo, ahora mismo la más moderna, indicamos **8: 1.4**, debe de aparecer seleccionado por defecto, pulsamos intro
- Indicamos la versión del módulo maven, indicamos **1.0-SNAPSHOT**, debe de aparecer seleccionado por defecto, pulsamos intro
- Nos confirma la información con la que se creará el módulo, groupId, artifactId, etc. Únicamente pulsamos **Y** para confirmar

Si todo ha ido bien, debería de aparecer en la consola **BUILD SUCCESS**

# Capítulo 3. SonarScanner

SonarScanner para Maven, es la vía por defecto recomendada para proyectos de tipo Java Maven.

Esta herramienta nos va a posibilitar de forma conjunta con el uso de Maven, que el análisis de código estático con SonarQube esté disponible en cualquier lugar donde tengamos Maven y por supuesto dicha herramienta.

Al utilizar Maven, las fases de compilación de Maven ya tienen gran parte de la información necesaria para que SonarQube lleve a cabo el escaneo de forma correcta.

Como gran ventaja, mediante el uso de esta herramienta, tenemos preconfigurado el análisis en función de esa información, la necesidad de una configuración manual se reduce significativamente casi a 0.

En definitiva, realizaremos la solicitud de análisis mediante la invocación de un goal del propio Maven.

## 3.1. Lab: SonarScanner con un proyecto de Java Maven

Mediante este laboratorio vamos a realizar cierta configuración en nuestro proyecto Java para enviar información a analizar al servidor de SonarQube.

### 3.1.1. Localizando y abriendo el archivo de configuración de Maven (settings.xml)

La idea, va a ser crear un perfil de Maven donde pondamos indicar el lugar donde se encuentra el servidor de SonarQube Runner, así como indicar algún que otro plugin que podamos necesitar para llevar a cabo el proceso.

Accedemos a la carpeta donde tengamos nuestro archivo de configuración de maven **settings.xml**, que estará en la ruta **/usr/local/apache-maven-3.X.X/conf/** y lo abrimos para editarlo.



Según la versión de Maven que tengamos instalada, indicaremos la misma sustituyendo los caracteres X, según (Major, Minor, Version)

Ejecutamos en consola:

```
$ sudo nano /usr/local/apache-maven-3.8.4/conf/settings.xml
```

### 3.1.2. Configurando el plugin

Vamos a localizar una sección que indique **<pluginGroups></pluginGroups>**, la editamos de forma que quede así:

```
<pluginGroups>
  <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
</pluginGroups>
```

### 3.1.3. Creando y configurando el profile

Para llevar a cabo la configuración del proyecto de una forma elegante, vamos a crear un profile de Maven que en caso determinado, podamos activar o desactivar a nuestro gusto.

El empleo de profiles en la definición de puntos de enganche de servidores SonarQube, tiene la ventaja de que podríamos tener ciertos proyectos a los que por ejemplo no quisiéramos aplicarle análisis y a otros que si, lo único que tendríamos que tener en cuenta es que cuando ejecutamos maven, indicaríamos en unos casos el profile para análisis y en otros casos el profile que no lleva a cabo tareas de analítica.

Localizamos una sección que indique **<profiles></profiles>**, dentro de la cual crearemos nuestro profile, quedará así:

```
<profiles>
  <profile>
    <id>sonar</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <sonar.host.url>
        http://localhost:9002
      </sonar.host.url>
    </properties>
  </profile>
</profiles>
```

### 3.1.4. Obteniendo token de seguridad

Para poder ejecutar lanzar un análisis de nuestro proyecto Java con SonarQube, necesitamos estar autorizados, con lo cual, vamos a necesitar generar un token de autenticación en el propio SonarQube.

Desde la página principal de SonarQube accedemos a la opción **Administration > Security > Users** y dentro de dicha página le tenemos que dar al botón **Tokens**.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects and files... + A




Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Users Create User

Create and administer individual users.

Search by login or name...

	SCM Accounts	Last connection	Groups	Tokens
 <b>Administrator</b> admin		< 1 hour ago	sonar-administrators sonar-users	0  

1 of 1 shown

En la ventana que nos aparece, tenemos que indicar un nombre al token que vamos a generar.

**Tokens of Administrator**

**Generate Tokens**

Enter Token Name Generate

Name	Last use	Created
No tokens		

[Done](#)

Indicamos como nombre del token **java-maven-token** y pulsamos sobre el botón Generate.



La cadena de caracteres perteneciente al token debemos de guardarla por que la necesitaremos más adelante, ya que el token no volverá a aparecer más.

En caso de que la perdamos, o bien, sospechemos que el token ha quedado comprometido, podemos eliminar el token y crear uno nuevo.

## Tokens of Administrator

### Generate Tokens



New token "jenkins-token" has been created. Make sure you copy it now, you won't be able to see it again!

Name	Last use	Created

### 3.1.5. Analizando el proyecto (Ejecución directa mediante el goal)

Llegado este momento, vamos a proceder a ejecutar el goal de Maven **sonar:sonar** en el directorio donde se encuentra el archivo **pom.xml**, suministraremos el token de autenticación que obtuvimos anteriormente:

Ejecutamos en consola el siguiente comando:

```
$ mvn clean verify sonar:sonar -Dsonar.login=1fe64da6045003f04369b450dd361cca13f0c88d
```

```
...
```

```
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9002/dashboard?id=com.tutorial.sonarqube%3Asonarqube-maven-project
```

```
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
```

```
[INFO] More about the report processing at http://localhost:9002/api/ce/task?id=AX7fVbAifb6UGtR1GPgP
```

```
[INFO] Analysis total time: 1:33.991 s
```

```
[INFO] -----
```

```
[INFO] BUILD SUCCESS
```

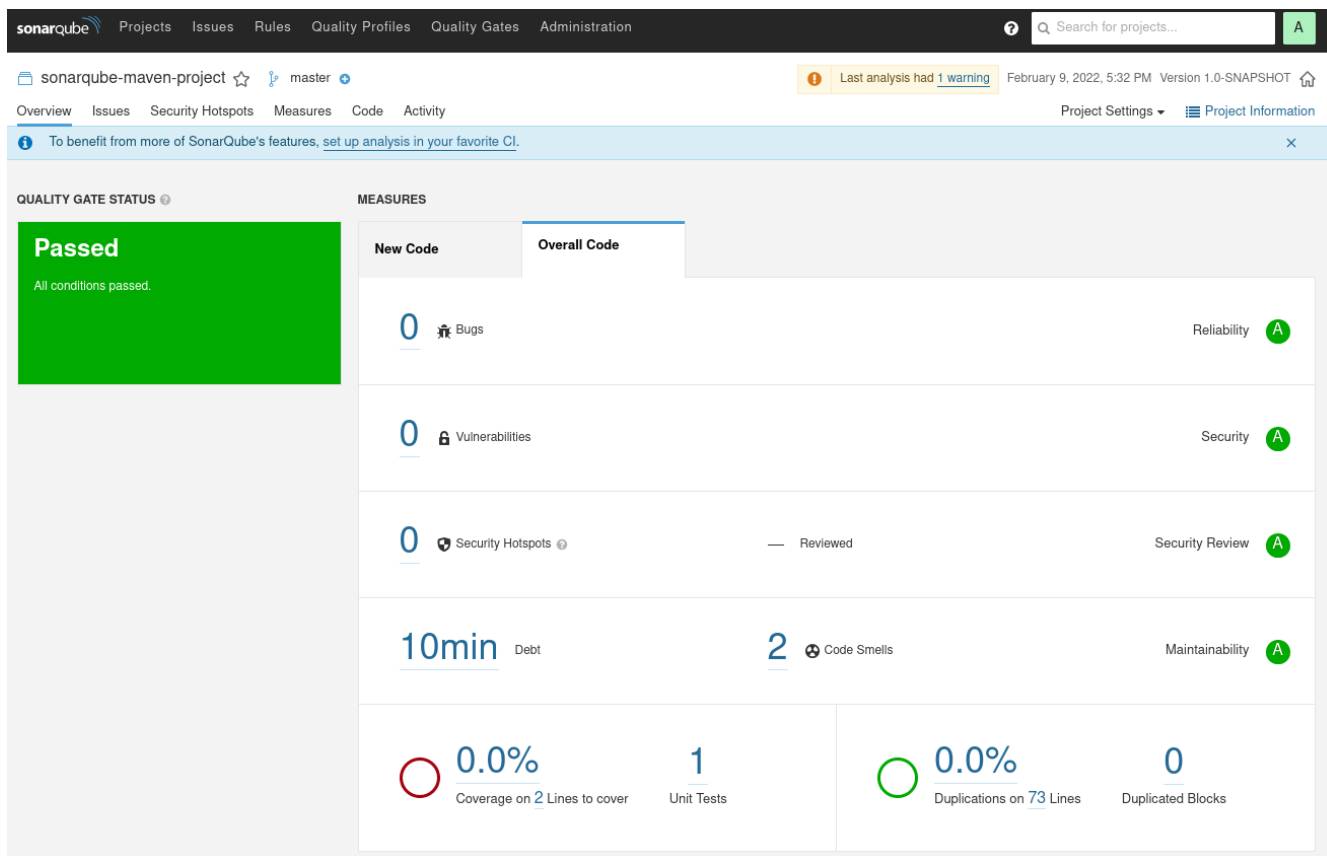
```
[INFO] -----
```

```
[INFO] Total time: 03:12 min
```

```
[INFO] Finished at: 20XX-XX-XXT17:34:02+01:00
```

```
[INFO] -----
```

Si visualizamos el dashboard de SonarQube para nuestro proyecto, observaremos como ya nos empiezan a aparecer distintas métricas según el análisis realizado.



### 3.1.6. Analizando el proyecto (Regulando propiedades de análisis)

La mayoría de las propiedades de configuración para el análisis se leerán del proyecto.

Podríamos querer sobre-escribir algunos valores predeterminados, así como querer especificar valores adicionales, en nuestro proyecto Maven vamos a poder llevar a cabo esta configuración en la sección de `<properties></properties>`, que se encuentra en nuestro archivo **pom.xml**.

Vamos a indicar las siguientes propiedades:

- **sonar.projectKey**
  - Identificador que actúa a modo de clave primaria del proyecto, para que SonarQube sepa diferenciarlos
- **sonar.projectName**
  - Nombre del proyecto que nos aparecerá en SonarQube
- **sonar.login**
  - Indicamos el Token de autenticación con SonarQube, o bien un usuario.
  - Si optamos por indicar un usuario como forma de autenticación, deberemos también de configurar una propiedad con la clave del mismo, dicha propiedad sería **sonar.password**
- **sonar.log.level**
  - Indicamos el nivel de traza deseada en el proceso
- **sonar.projectDescription** \*\*
- **sonar.host.url**



- Url donde se encuentra el servidor de SonarQube

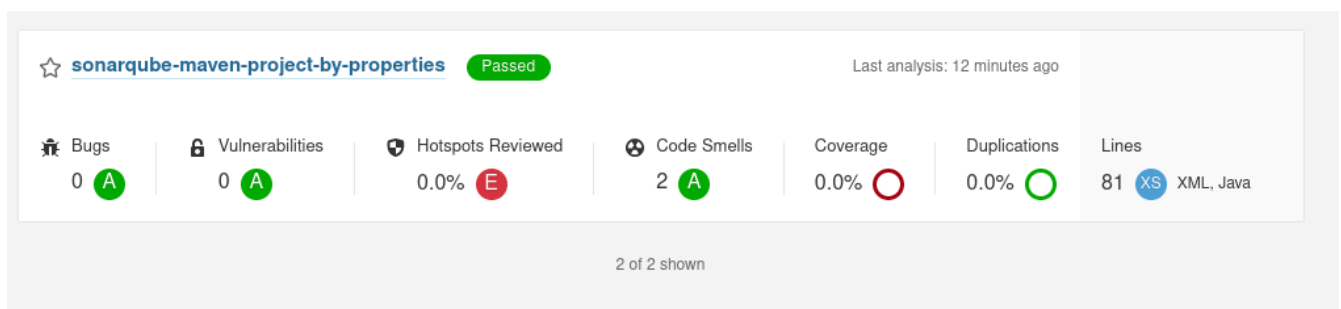
```
<properties>
  <sonar.projectKey>sonarqube-maven-project-by-properties</sonar.projectKey>
  <sonar.projectName>sonarqube-maven-project-by-properties</sonar.projectName>
  <sonar.login>1fe64da6045003f04369b450dd361cca13f0c88d</sonar.login>
  <sonar.log.level>INFO</sonar.log.level>
  <sonar.qualitygate.wait>true</sonar.qualitygate.wait>
  <sonar.qualitygate.timeout>300</sonar.qualitygate.timeout>
  <sonar.projectDescription>App Backend</sonar.projectDescription>
  <sonar.host.url>http://localhost:9000</sonar.host.url>
</properties>
```

Ejecutamos en consola el siguiente comando:

```
$ mvn clean verify sonar:sonar
...

[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=308ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=416ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=1ms
[INFO] Analysis report generated in 939ms, dir size=119.4 kB
[INFO] Analysis report compressed in 214ms, zip size=18.8 kB
[INFO] Analysis report uploaded in 1304ms
[INFO] ----- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] QUALITY GATE STATUS: PASSED - View details on http://localhost:9002/dashboard?id=sonarqube-maven-project-by-properties
[INFO] Analysis total time: 1:25.948 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:10 min
[INFO] Finished at: 20XX-0X-0XT19:11:43+01:00
[INFO] -----
```

Si visualizamos el dashboard de SonarQube para nuestro proyecto, observaremos como ya nos empiezan a aparecer distintas métricas según el análisis realizado con este nuevo proyecto registrado en SonarQube:



# Capítulo 4. Administración de usuario, grupos y roles

SonarQube viene con una serie de características de seguridad globales:

- Mecanismos de autenticación y autorización
- La capacidad de obligar a los usuarios a autenticarse antes de que puedan ver cualquier parte de una instancia de SonarQube
- La capacidad de delegar a la autenticación en sistemas de terceros, como por ejemplo LDAP, etc.

Adicionalmente se pueden configurar a nivel de grupo o usuarios capacidades como:

- Ver si existe o no un proyecto determinado
- La posibilidad de acceder o no al código fuente de un proyecto
- Administrar un proyecto
  - Establecer patrones de exclusión
  - Ajustar la configuración del complemento para ese proyecto
  - Etc.
- Administrar perfiles de calidad, puertas de calidad y la propia instancia de SonarQube.

Otro aspecto de la seguridad es el cifrado de configuraciones como las contraseñas.

SonarQube proporciona un mecanismo integrado para cifrar la configuración.

## 4.1. Autenticación en SonarQube

Por defecto, SonarQube fuerza la autenticación de usuarios.

Podemos deshabilitar la obligatoriedad de autenticación de usuario y permitir que usuarios anónimos vean proyectos y ejecuten análisis en SonarQube.

## 4.2. Autenticación de endpoints API

SonarQube dispone de un conjunto de API endpoints para llevar a cabo la integración de cualquier otro sistema de nuestra empresa con SonarQube.



Si desactivamos la autenticación de forma explícita en SonarQube, tendremos disponible el acceso de forma pública a los endpoints de SonarQube sin ningún tipo de restricción.

## 4.3. Mecanismos de autenticación

La autenticación se puede gestionar a través de varios mecanismos:

- A través de la base de datos integrada de usuarios/grupos de SonarQube
- A través de proveedores de identidad externos
  - Como un servidor LDAP (incluido el servicio LDAP de Active Directory)
  - GitHub
  - Etc.
- A través de encabezados HTTP

## 4.4. Tokens de acceso a SonarQube

Cuando creas un usuario en la propia base de datos de SonarQube, se considera local y solo se autenticará en la propia base de datos de usuarios/grupos de SonarQube y no en ninguna herramienta externa (LDAP, Active Directory, Crowd, etc.).

Por defecto, el administrador es una cuenta local.

Del mismo modo, todas las cuentas no locales se autenticarán solo con la herramienta externa.

Un administrador puede administrar tokens en nombre de un usuario accediendo a **Administration > Security > Users**.

Desde esa opción, podemos hacer clic en la columna **Tokens** del usuario para ver los tokens existentes del usuario y revocar tokens existentes o generar nuevos.

Una vez establecido, un token es la única credencial necesaria para ejecutar un análisis.

Los tokens deben pasarse como el valor de la propiedad **sonar.login**.

## 4.5. Credenciales por defecto

Nada mas terminar de instalar SonarQube, se crea un usuario local administrador con las siguientes credenciales:

- Login
  - admin
- Password
  - admin

## 4.6. Gestión de la Autorización en SonarQube

La forma en que se implementa la autorización en SonarQube es bastante estándar.

Es posible crear tantos usuarios y grupos de usuarios como sea necesario.

Luego, los usuarios pueden unirse (o no) a (múltiples) grupos.

A los grupos y/o usuarios se les otorgan (múltiples) permisos.

Los permisos otorgan acceso a:

- Proyectos
- Servicios
- Funcionalidades

Para administrar grupos y usuarios, podemos realizarlo desde el menú **Administration > Security**, y utilizando los elementos del submenú.

## 4.7. Gestión de usuarios en SonarQube

Hay disponibles múltiples integraciones que permiten la delegación de la autenticación , pero podemos crear y editar usuarios manualmente en el menú **Settings > Security > Users**.

Para los usuarios creados manualmente, el inicio de sesión y la contraseña se pueden configurar en el momento de la creación.

Los usuarios creados manualmente pueden editar sus contraseñas.

Durante la creación y edición de usuarios, podemos configurar el nombre de pantalla y la dirección de correo electrónico de una cuenta.

## 4.8. Gestión de grupos en SonarQube

Un grupo es un conjunto de usuarios.

Para administrar grupos, podemos hacerlo desde la sección **Administration > Security > Groups**.

Para editar la membresía de un grupo, hacemos clic en el icono junto al total de membresía.

En SonarQube, dos grupos tienen un significado especial:

- **Anyone** es un grupo que existe en el sistema, pero que no se puede administrar.
  - Todos los usuarios pertenecen a este grupo (Anyone), incluidos los usuarios anónimos.
- **sonar-users** es el grupo predeterminado al que se agregan automáticamente los usuarios.

## 4.9. Permisos globales en SonarQube

Para asignar permisos globales, accederemos al sistema en calidad de usuarios de sistema administradores, y desde el menú principal accederíamos a la sección de **Administration > Security > Global Permissions**

Indicamos los permisos globales de SonarQube

- **Administer System**
  - Disponemos de todas las funciones de administración en la instancia de SonarQube
- **Administer Quality Profiles**

- Permite llevar a cabo cualquier acción en los Quality Profiles, incluida la opción de delegar permisos específicos de acceso a los perfiles.
- **Administer Quality Gates**
  - Permite llevar a cabo cualquier acción en las Quality Gates, incluida la opción de delegar permisos específicos de acceso.
- **Execute Analysis**
  - Otorga permiso para poder ajustar toda la configuración necesaria para poder llevar a cabo un análisis de código, esto incluye la gestión/configuración que haya que realizar en los proyectos privados, pero no incluye la gestión de contraseñas de acceso.
- **Create Projects**
  - Permite llevar a cabo la inicialización de la estructura de un nuevo proyecto antes de llevar a cabo el primer análisis del mismo.
  - Este permiso es igualmente requerido cuando realizamos el primer análisis de un proyecto que no ha sido creado previamente desde la interfáz web
- **Create Applications**
  - Permite crear nuevas aplicaciones dentro de un proyecto de análisis

## 4.10. Permisos de un proyecto

De igual forma que podemos contar con permisos globales en SonarQube, también podemos disponer de permisos por proyecto, podemos acceder a su configuración desde el menú **Project Settings > Permissions**

La visibilidad de un proyecto podemos ajustarla a pública o privada.

Si ajustamos la visibilidad de un proyecto a privada, el proyecto oculta el código fuente y las métricas de los usuarios que estén vinculados únicamente con el grupo Anyone.

Tanto si la visibilidad del proyecto es pública o privada, disponemos de estos permisos específicos que podemos ajustar a demanda:

- **Administer Issues**
  - Ejecutar cambios en la severidad de la incidencias detectadas, resolver incidencias
- **Administer Security Hotspots**
  - Cambiar el status de los Security Hotspots detectados
- **Administer**
  - Acceso a la configuración del proyecto y realizar tareas de administración (los usuarios también necesitan el permiso **Examinar** para poder visualizar un proyecto).
  - De forma predeterminada, un usuario con el permiso **Administer** podemos administrar tanto la configuración como los permisos para el proyecto actual.
  - Para permitir que solo los administradores de proyectos actualicen la configuración del proyecto, lo gestionamos desde el menú principal, **Administration > Configuration >**

**General Settings > Security**, y deshabilitamos la propiedad **Enable permission management for project administrators**.

- **Execute Analysis**

- Otorga acceso a toda la configuración requerida para llevar a cabo tareas de análisis, así como la habilidad de subir el resultado de los análisis a la propia instancia de SonarQube.
- Este permiso a nivel de proyecto incluye la posibilidad de llevar a cabo configuración de proyectos privados pero no otorga capacidades como el cambio de contraseñas.

Los proyectos privados tienen dos permisos adicionales:

- **Browse**

- Permite el acceso al proyecto, navegación, consulta de las métricas, incidencias, security spots
- Llevar a cabo algunas operaciones de edición (confirmar, resolver, reabrir, comentar)
- Realizar comentarios sobre un cambio asignado a un usuario sobre una security hotspot

- **See Source Code**

- Permite visualizar el código fuente del proyecto



Los permisos no son acumulativos.

Por cada instancia de SonarQube, si queremos habilitar la capacidad de administrar el proyecto, debemos otorgar adicionalmente el permiso de **Browse**.

Este permiso es otorgado por defecto para proyectos públicos.

Podemos añadir manualmente permisos para cada proyecto, sobre unos usuarios determinados o bien sobre un grupo, o aplicar permisos basados en plantillas

## 4.11. Plantillas de permisos para permisos por defecto

SonarQube viene de serie con una plantilla de permisos predeterminada, que otorga automáticamente permisos específicos a ciertos grupos cuando se crea un proyecto, una cartera de proyectos o una aplicación.

Es posible editar esta plantilla y crear plantillas adicionales.

Se puede configurar una plantilla separada para cada tipo de recurso.

Además, para los proyectos, podemos hacer que una plantilla se aplique solo a un subconjunto de nuevos proyectos utilizando una expresión regular de clave de proyecto (el Patrón de clave de plantilla **Project Key Pattern**).

De forma predeterminada, a cada nuevo proyecto con una clave que coincida con el patrón proporcionado se le aplicarán los permisos de la plantilla.

Las plantillas están vacías inmediatamente después de su creación.

Al hacer clic en el nombre de la plantilla, accederá a su interfaz de edición de permisos.

Las plantillas se administran a través de la opción de menú **Administration > Security > Permission Templates**

## 4.12. Permisos de creadores

**Creators** es un grupo especial que aparece solo en la interfaz de edición de plantillas de permisos.

Todos los permisos asignados a este grupo se otorgarán en el momento de la creación del **proyecto/portfolio/aplicación** a la única cuenta de usuario utilizada para crear el proyecto.

Esto permite a los administradores de SonarQube dejar que los usuarios creen y administren sus propios proyectos de forma autónoma.

Si bien las plantillas se pueden aplicar después de la creación del proyecto, la aplicación de una plantilla que incluye permisos de **Creators** a un **proyecto/portfolio/aplicación** existente no otorgará los permisos pertinentes al creador original del proyecto porque esa asociación no se almacena.

## 4.13. Reseteo de permisos de un proyecto

Para aplicar plantillas de permisos a proyectos, podemos acceder al menú de **Administration > Projects > Management**.

Podemos aplicar una plantilla a un proyecto específico utilizando la opción **Actions > Apply Permission Template**, específica del proyecto o utilizar la plantilla de permisos de aplicación masiva para aplicar una plantilla a todos los proyectos seleccionados.



Tenemos que tener en cuenta que no hay relación entre un proyecto y una plantilla de permiso.

Esto significa que:

- Los permisos de un proyecto se pueden modificar después de que se haya aplicado una plantilla de permiso a ese proyecto.
- Ninguno de los permisos del proyecto cambia cuando se modifica una plantilla de permiso.

## 4.14. Lab: Administración de usuarios, grupos y roles

Mediante este laboratorio, llevaremos a cabo diferentes operaciones de gestión con los elementos que forman parte del ecosistema del control de acceso en SonarQube.

Crearemos dos proyectos desde SonarQube, la idea es que distintos usuarios puedan acceder a dichos proyectos con diferentes permisos.

- **frontend-project**
- **backend-project**

Crearemos 2 grupos de usuarios, para dos departamentos:

- **development**
- **sqa**

Crearemos la siguiente relación de usuarios asociados a cada grupo/proyecto:

- **frontend-project**
  - **development**
    - Queremos que si que puedan ver el código fuente
    - No queremos que puedan administrar issues
    - No queremos que puedan administrar hotspots de seguridad
    - Queremos que puedan ejecutar análisis directos
    - Usuarios asociados
      - user1
      - user2
- **backend-project**
  - **sqa**
    - No queremos que vean el código fuente
    - Queremos que puedan administrar hotspots de seguridad
    - Queremos que puedan administrar issues
    - No queremos que puedan ejecutar análisis directos
    - Usuarios asociados
      - user3
      - user4

#### 4.14.1. Creando los proyectos

Inicialmente, vamos a crear los proyectos desde el propio SonarQube.

Desde el dashboard principal, accedemos a **Administration > Projects > Management** y pulsamos sobre el botón **Create Project**



## Create Project

All fields marked with \* are required

**Name \***

**Key \***

**Visibility**

☐

Public

☒

Private

Create

[Cancel](#)

Creamos un primer proyecto introduciendo los siguientes datos:

- **Name**
  - frontend-project
- **Key**
  - frontend\_project
- **Visibility**
  - Private

Pulsamos el botón **Create**

Creamos un segundo proyecto introduciendo los siguientes datos:

- **Name**
  - backend-project
- **Key**
  - backend\_project
- **Visibility**
  - Private

Pulsamos el botón **Create**

### 4.14.2. Creando los grupos

A continuación, vamos a crear los grupos, con los que posteriormente relacionaremos a los usuarios.

Desde el dashboard principal, accedemos a **Administration > Security > Groups** y pulsamos sobre el botón de **Create Group**

## Create Group

All fields marked with \* are required

**Name \***

**Description**

Create

[Cancel](#)

Creamos un primer grupo introduciendo los siguientes datos:

- **Name**
  - development
- **Description**
  - Development group for SonarQube

Pulsamos el botón **Create**

Creamos un segundo grupo introduciendo los siguientes datos:

- **Name**
  - sqa
- **Description**
  - SQA group for SonarQube

Pulsamos el botón **Create**

### 4.14.3. Creando los usuarios

Accedemos desde el dashboard principal a la opción **Administration > Security > Users** y pulsamos sobre el botón de **Create User**

## Create User

All fields marked with \* are required

**Login \***

Minimum 3 characters

**Name \***

**Email**

**Password \***

**SCM Accounts**

[Add](#)

Login and email are automatically considered as SCM accounts

[Create](#)

[Cancel](#)

Creamos un primer usuario introduciendo los siguientes datos:

- **Login**
  - user1
- **Name**
  - User1
- **Email**
  - [user1@user1.com](#)
- **Password**
  - user1

Pulsamos el botón **Create**

Creamos un segundo usuario introduciendo los siguientes datos:

- **Login**
  - user2
- **Name**
  - User2

- **Email**
  - [user2@user2.com](mailto:user2@user2.com)
- **Password**
  - user2

Pulsamos el botón **Create**

Creamos un tercer usuario introduciendo los siguientes datos:

- **Login**
  - user3
- **Name**
  - User3
- **Email**
  - [user3@user3.com](mailto:user3@user3.com)
- **Password**
  - user3

Pulsamos el botón **Create**

Creamos un cuarto usuario introduciendo los siguientes datos:

- **Login**
  - user4
- **Name**
  - User4
- **Email**
  - [user4@user4.com](mailto:user4@user4.com)
- **Password**
  - user4

Pulsamos el botón **Create**



Observaremos que por defecto se incluyen en el grupo **sonar-users**

#### 4.14.4. Asignando los usuarios a los grupos

Accedemos desde el dashboard principal a la opción **Administration > Security > Groups**.

Ahí aparecerán varios grupos de usuarios, donde encontraremos una columna que dice **Members**, en esa columna, aparecerá un indicador con el número de usuarios que están vinculados actualmente a ese grupo, hacemos clic sobre el icono de líneas juntas:

Groups

Create and administer groups of users.

Create Group

Search by name...

	Members	Description	
Anyone		Anybody (authenticated or not) who browses the application belongs to this group	
development	0	Development group	
sonar-administrators	1	System administrators	
sonar-users (Default)	1	Any new users created will automatically join this group	
sqa	0	Sqa group	

5 of 5 shown

Asociamos en un primer momento los usuarios al grupo de **development** (user1, user2):

## Update users

Selected

Unselected

All

Search

☐
Administrator  
admin

☒
User1  
user1

☒
User2  
user2

☐
User3  
user3

☐
User4  
user4

Done

Pulsamos sobre el botón **Done**


Seguidamente, vamos a asociar los usuarios al grupo de **sqa** (user3, user4):

## Update users

Selected

Unselected

All

 Search

☐

Administrator  
admin

☐

User1  
user1

☐

User2  
user2

☒

User3  
user3

☒

User4  
user4

Done

### 4.14.5. Asignando los grupos a los proyectos

Llegado este punto, toca vincular los grupos con los proyectos.

Desde el dashboard principal, accedemos a la opción de **Administration > Projects > Management**.

Sobre el proyecto **frontend-project**, pulsaremos sobre el icono de la rueda dentada de cada proyecto, y a su vez, pulsamos sobre la opción de **Edit Permissions**.

En la matriz, para el grupo **development**, indicamos los siguientes permisos:

- **Browse**
- **See Source Code**
- **Execute Analysis**

frontend-project ☆ master

Overview Issues Security Hotspots Measures Code Activity

Project Settings ▾ Project Information

All	Users	Groups	Search for users or groups...	Browse ?	See Source Code ?	Administer Issues ?	Administer Security Hotspots ?	Administer ?	Execute Analysis ?
	<b>sonar-administrators</b> System administrators			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	<b>sonar-users</b> Any new users created will automatically join this group			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>Administrator</b> admin			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>Anyone</b>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User1</b> user1 user1@user1.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User2</b> user2 user2@user2.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User3</b> user3 user3@user3.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User4</b> user4 user4@user4.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>development</b> Development group			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	<b>sqa</b> Sqa group			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

10 of 10 shown

Cambiamos de proyecto a **backend-project**, pulsamos sobre el icono de la rueda dentada de cada proyecto, y a su vez, pulsamos sobre la opción de **Edit Permissions**.

Adicionalmente, en la matriz, para el grupo **sqa**, indicamos los siguientes permisos:

- **Browse**
- **Administer Issues**
- **Administer Security Hotspots**

backend-project ☆ master

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

This project is private. Only authorized users can browse and see the source code.

Public Private

All	Users	Groups	Search for users or groups...	Browse	See Source Code	Administer Issues	Administer Security Hotspots	Administer	Execute Analysis
	<b>sonar-administrators</b> System administrators			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	<b>sonar-users</b> Any new users created will automatically join this group			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>Administrator</b> admin			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>Anyone</b>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User1</b> user1 user1@user1.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User2</b> user2 user2@user2.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User3</b> user3 user3@user3.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>User4</b> user4 user4@user4.com			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>development</b> Development group			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<b>sqa</b> Sqa group			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

10 of 10 shown

#### 4.14.6. Generando tokens de acceso a los usuarios

Vamos a llevar a cabo una prueba de envío de código para analizar en SonarQube, para verificar que el usuario **user1** efectivamente puede llevar a cabo la prueba, y por otro lado, realizaremos la misma operación con el usuario **user3** de forma que se encontraría con un error que le imposibilitaría llevar a cabo esta acción.

Desde el dashboard principal accedemos al menú de **Administration > Security > Users** y hacemos clic en el icono de la columna **Tokens** para el usuario **user1**.

Indicamos como nombre del token **user1\_token** y pulsamos sobre el botón **Generate**.



El token debemos de guardarlo de forma inmediata por que no volverá a aparecer nunca más.



## Tokens of User1

### Generate Tokens

Generate

New token "user1\_token" has been created. Make sure you copy it now, you won't be able to see it again!



Copy

d92dce4686482c1c01e6de69b95a59f3d91ba389

Name	Last use	Created
user1_token	Never	[REDACTED]

Revoke

Done

Ahora, hacemos clic en el icono de la columna **Tokens** para el usuario **user3**.

Indicamos como nombre del token **user3\_token** y pulsamos sobre el botón **Generate**.

## Tokens of User3

### Generate Tokens

Generate

New token "user3\_token" has been created. Make sure you copy it now, you won't be able to see it again!



Copy

fc5d260ffd91a5abaa8f1bda14eb14919c6f3687

Name	Last use	Created
user3_token	Never	[REDACTED]

Revoke

Done

## 4.14.7. Enviando solicitud de análisis a SonarQube

Ejecutamos en nuestro proyecto Java Maven la siguiente instrucción (**user1**):

```
$ mvn clean verify sonar:sonar -Dsonar.login=d46ab3fe979b1d5c226836f10f0829e93917cd6a -Dsonar.projectKey=frontend_project -Dsonar.projectName=backend-project -Dsonar.projectDescription=FrontendProject

...
[INFO] ----- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] QUALITY GATE STATUS: PASSED - View details on http://localhost:9002/dashboard?id=sonarqube-maven-project-by-properties
[INFO] Analysis total time: 1:35.456 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:28 min
[INFO] Finished at: 20XX-0X-15T10:17:40+01:00
[INFO] -----
```

Ahora, vamos a intentar realizar la misma operación con un usuario que no tiene permisos de ejecución de análisis (**user3**):

```
$ mvn clean verify sonar:sonar -Dsonar.login=627b61d641958083a06e13e01d0a05f95d523b9b -Dsonar.projectKey=backend_project -Dsonar.projectName=backend-project -Dsonar.projectDescription=BackendProject

...
WARNING: A connection to http://localhost:9002/ was leaked. Did you forget to close a response body? To see where this was allocated, set the OkHttpClient logger level to FINE:
Logger.getLogger(OkHttpClient.class.getName()).setLevel(Level.FINE);
[INFO] Sensor VB.NET Analysis Log [vbnet] (done) | time=646ms
[INFO] Sensor VB.NET Properties [vbnet]
[INFO] Sensor VB.NET Properties [vbnet] (done) | time=2ms
[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=85ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=335ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=0ms
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 02:06 min
[INFO] Finished at: 20XX-0X-15T12:23:57+01:00
[INFO] -----
[ERROR] Failed to execute goal org.sonarsource.scanner.maven:sonar-maven-plugin:3.9.1.2184:sonar (default-cli) on project sonarqube-maven-project: null: MojoExecutionException: NullPointerException -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://wiki.apache.org/confluence/display/MAVEN/MojoExecutionException
```

# Capítulo 5. Plugins de integración de SonarQube con Eclipse

El ecosistema de SonarQube nos proporciona plugins para proceder con una integración con la plataforma SonarQube a través de nuestro IDE favorito.

Entre otros, hay plugins disponibles para los siguientes IDE's:

- JetBrains
- Eclipse
- Visual Studio
- VS Code

SonarLint actúa como una extensión del IDE de desarrollo, de manera que nos podrá ir detectando bugs y vulnerabilidades mientras creamos nuestro código.

SonarQube por contra, actuaría como la herramienta Code Quality & Code Security de la empresa, operando típicamente en un servidor central para tal cometido.

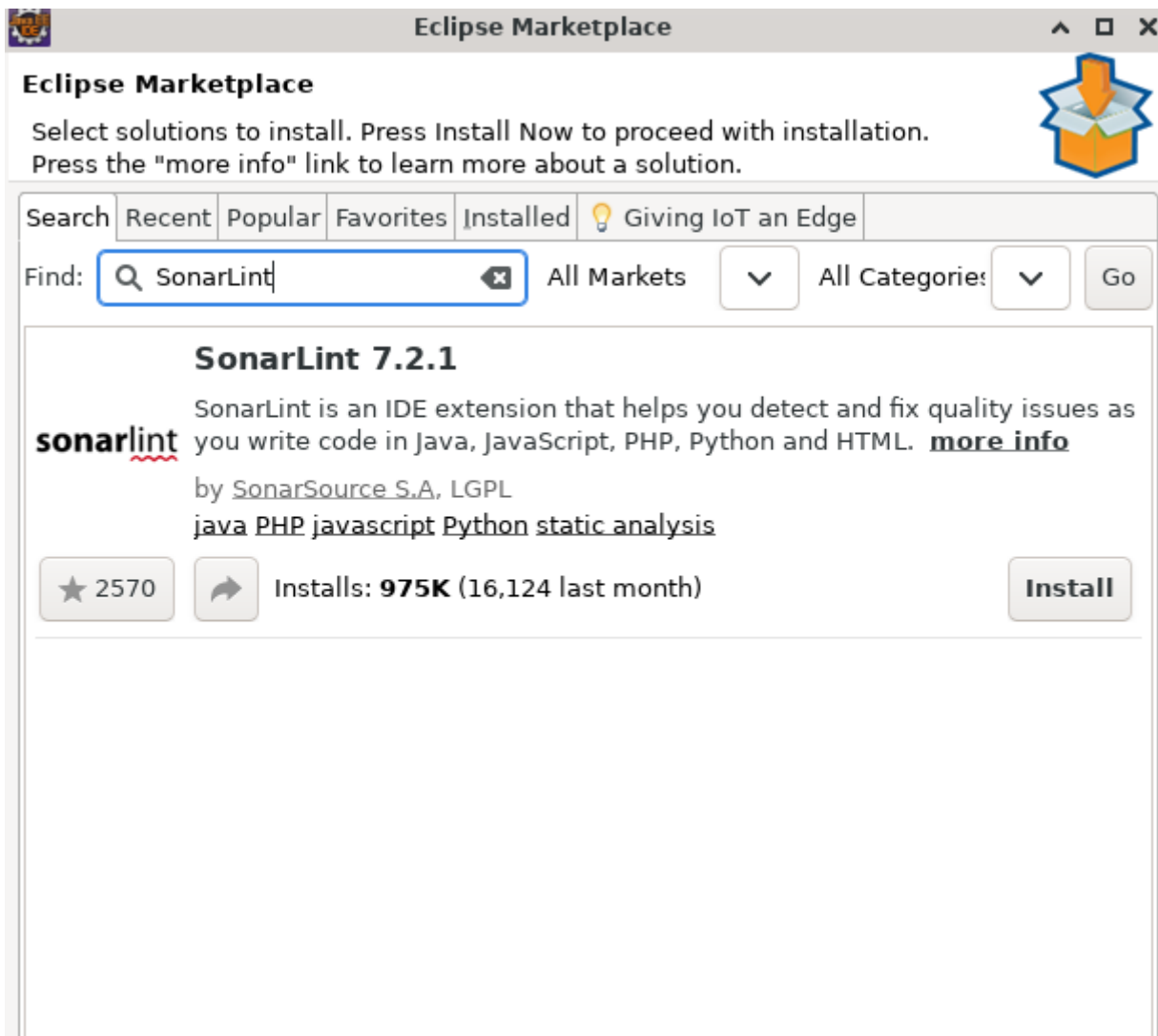
## 5.1. Lab: Integración de plugins SonarQube con Eclipse

Mediante este laboratorio vamos a realizar las acciones necesarias para poder integrar el IDE eclipse con una instancia que se encuentre operando de SonarQube.

### 5.1.1. Instalación de plugin SonarLint

Dentro del IDE nos vamos a la opción **Help > Eclipse Marketplace**

Escribimos SonarLint y debería de aparecer dicho plugin, pulsamos sobre el botón de Install.

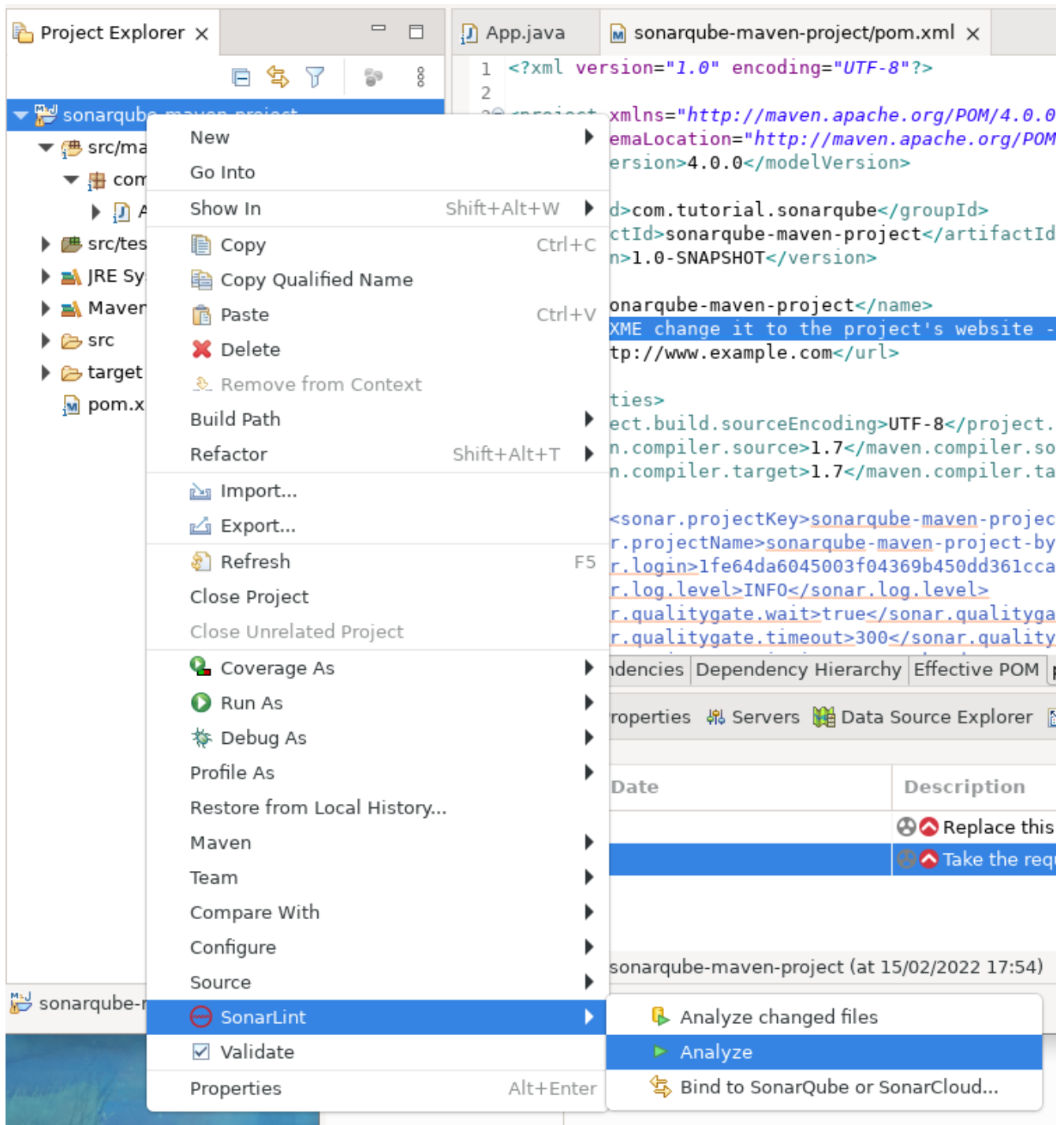


En la pantalla siguiente, aceptamos los términos del acuerdo de licencia y pulsamos **Finish**

### 5.1.2. Solicitud de análisis en el proyecto

Una vez tengamos abierto nuestro proyecto en el IDE, pulsamos con el botón derecho del ratón, **SonarLint > Analyze**

Nos preguntará si en lugar de llevar a cabo un análisis general de todo el proyecto, si por el contrario, deseamos activar el análisis en tiempo real en el código **on-the-fly**, para lo cual, pulsamos sobre el botón de **OK**



Ahora, cada vez que abramos algún archivo, el plugin indicará issues que vaya detectando en los archivos que formen parte del código fuente:

```

1 package com.tutorial.sonarqube;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hello World!" );
12    }
13 }
14

```

Replace this use of System.out or System.err by a logger.  
2 quick fixes available:  
[Open description of rule java:S106](#)  
[Deactivate rule ijava:S106](#)  
Press 'F2' for focus

2 items

Resource	Date	Description
App.jav		Replace this use of System.out or System.err by a logger.
pom.xml		Take the required action to fix the issue indicated by this "FIXME" comment.

### 5.1.3. Configuración en modo conectado

Una posibilidad que tenemos al utilizar el plugin de SonarLint, es utilizarlo en **Modo Conectado**, esto significa que podemos llevar a cabo ciertos ajustes para utilizando en tiempo real conectado con nuestra instancia SonarQube.

Podemos enumerar los siguientes beneficios cuando se utiliza el modo conectado:

- Usamos los mismos analizadores que el servidor, suponiendo que sean compatibles con SonarLint
- Sincronizamos el mismo perfil de calidad (mismas reglas de activación, parámetros, severidad, etc.)
- Sincronizamos algunas configuraciones definidas en el servidor (exclusiones de reglas, parámetros del analizador, etc.)
- Suprime automáticamente los problemas que están marcados como **No arreglar** o **Falso positivo** en el servidor

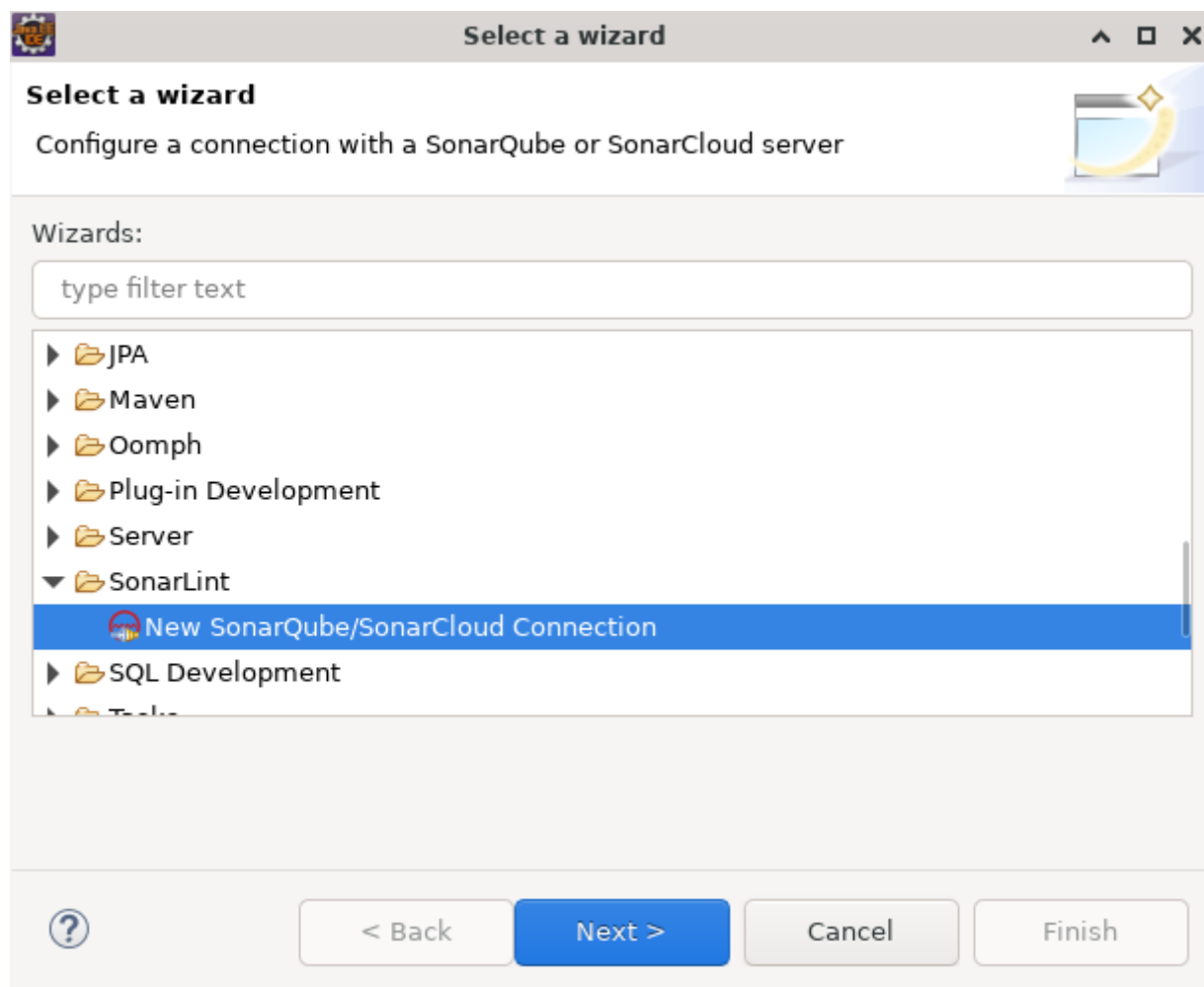


El modo conectado no envía problemas al servidor.

Más bien, su propósito es configurar el IDE para que use la misma configuración que el servidor.

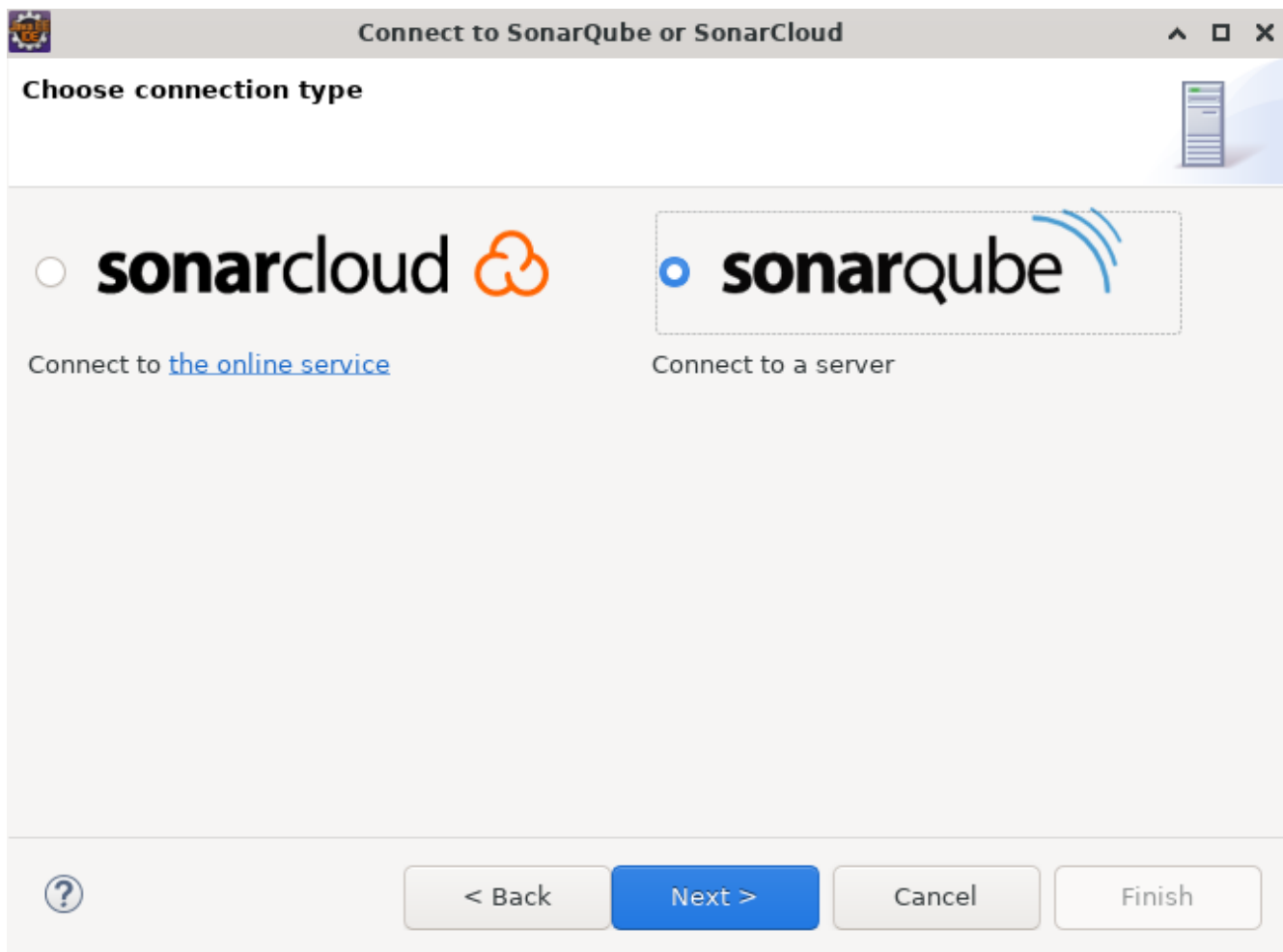
#### 5.1.4. Configurando la conexión

En primer lugar, accedemos en el IDE a la opción **File > New Other > SonarLint > New SonarQube/SonarCloud Connection**



Pulsamos **Next**

A continuación, vamos a indicar el tipo de instancia de SonarQube a la cual queremos conectar, en nuestro caso seleccionamos **SonarQube**

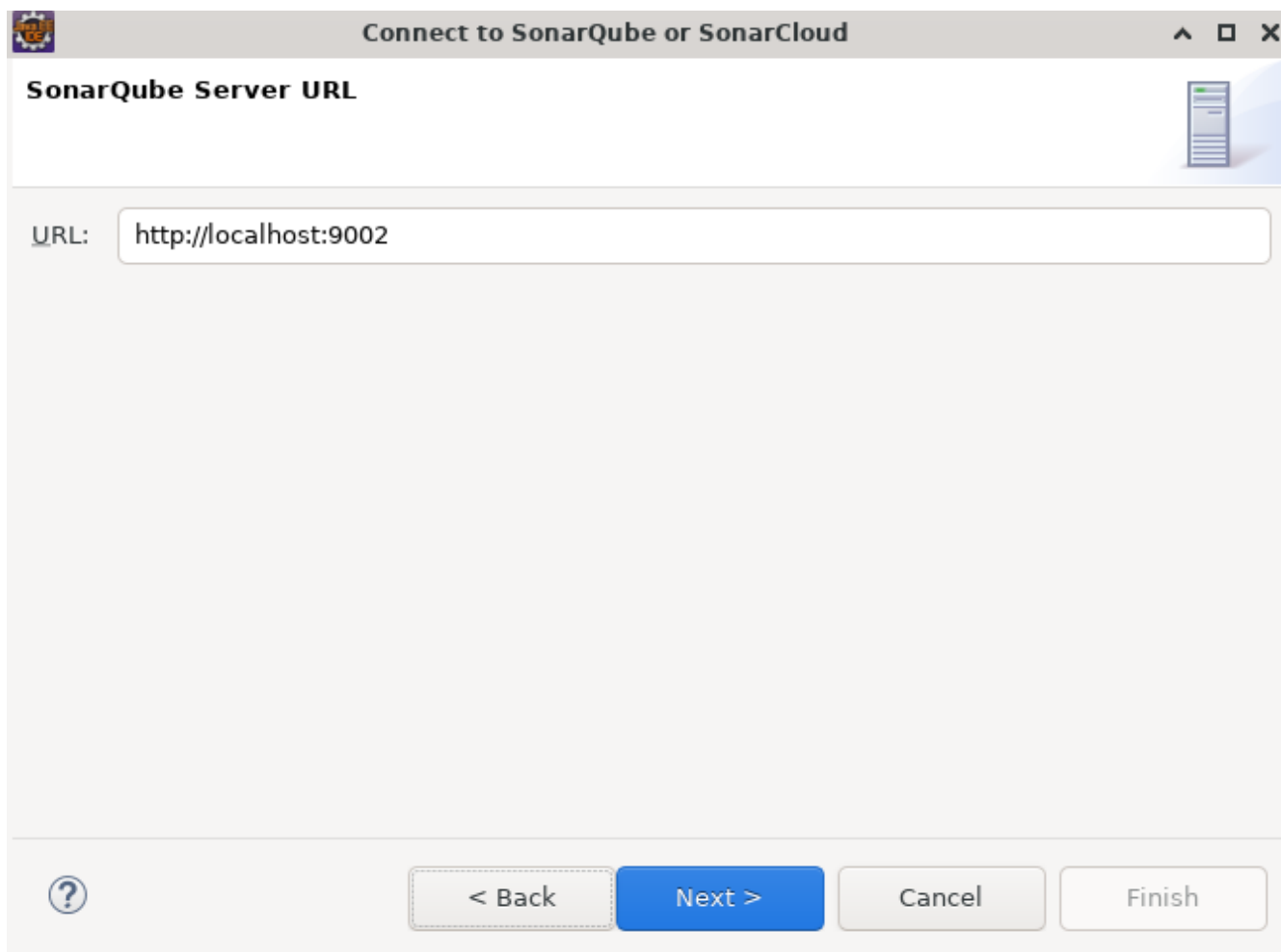


Pulsamos **Next**

En la siguiente pantalla, indicamos la URL donde se encuentra nuestra instancia operando.

En nuestro caso indicamos como URL: <http://localhost:9002/>



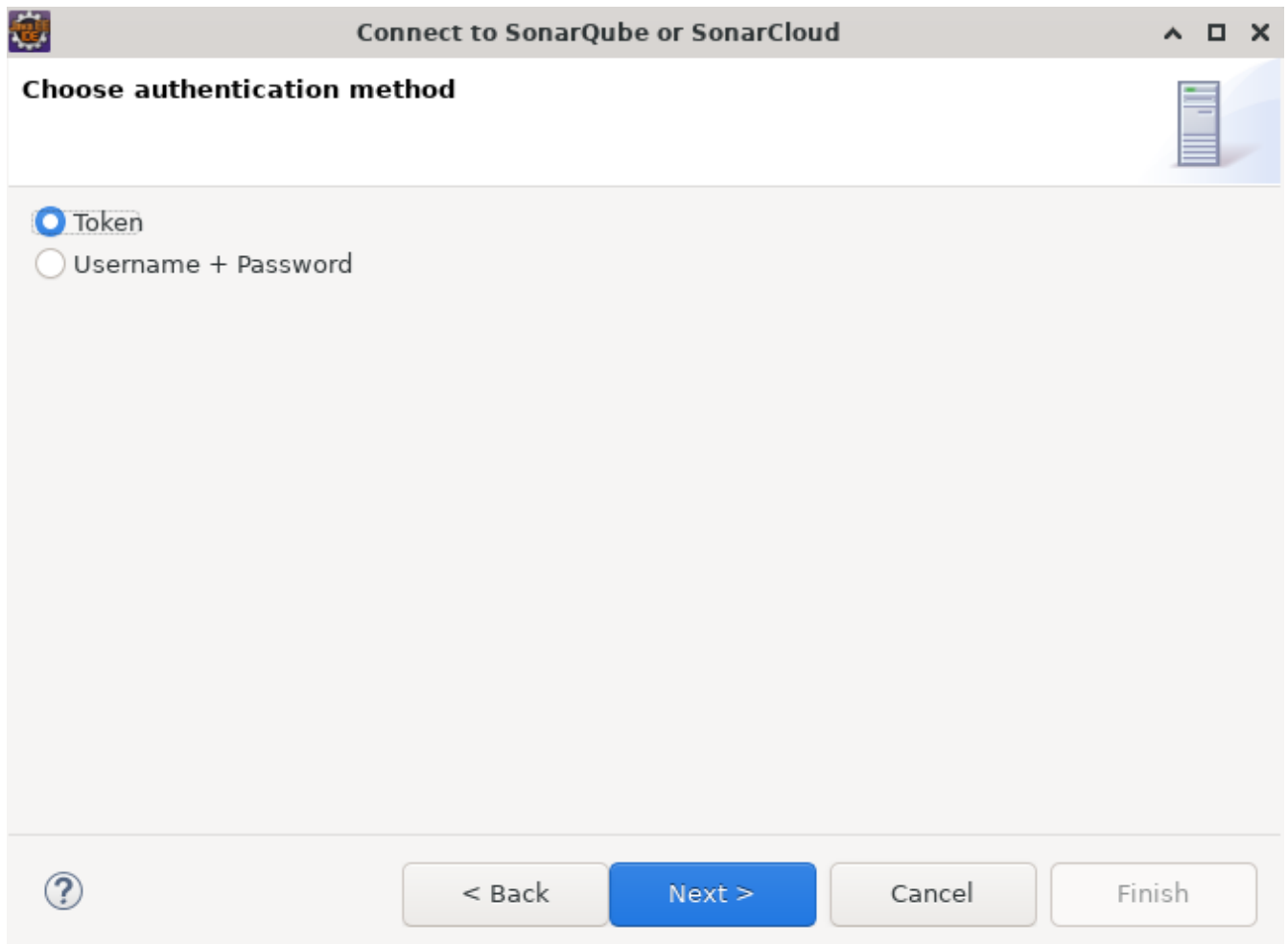


Pulsamos **Next**

En esta pantalla, debemos de elegir el modo de autenticación, bien por token o por usuario/contraseña.

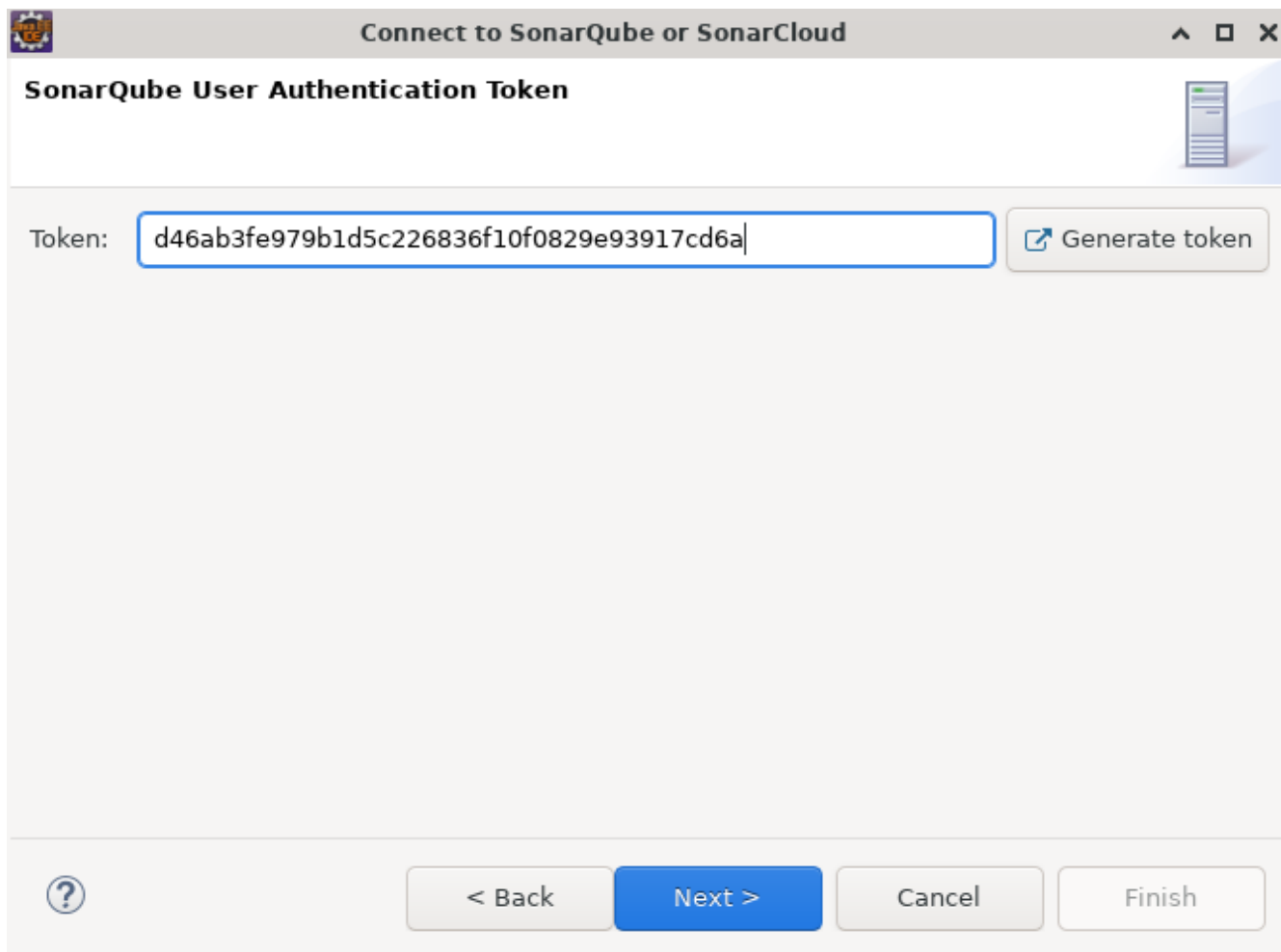
La recomendación sería utilizar el mecanismo de Token, ya que por seguridad, de esta forma no estamos indicando en el IDE local nuestro acceso directo a SonarQube, sino el token.

Si el token quedara comprometido, siendo administradores en SonarQube podemos revocar y generar uno nuevo, manteniendo a salvaguarda las credenciales de acceso del usuario en cuestión.



Elegimos la opción de **Token** y pulsamos **Next**

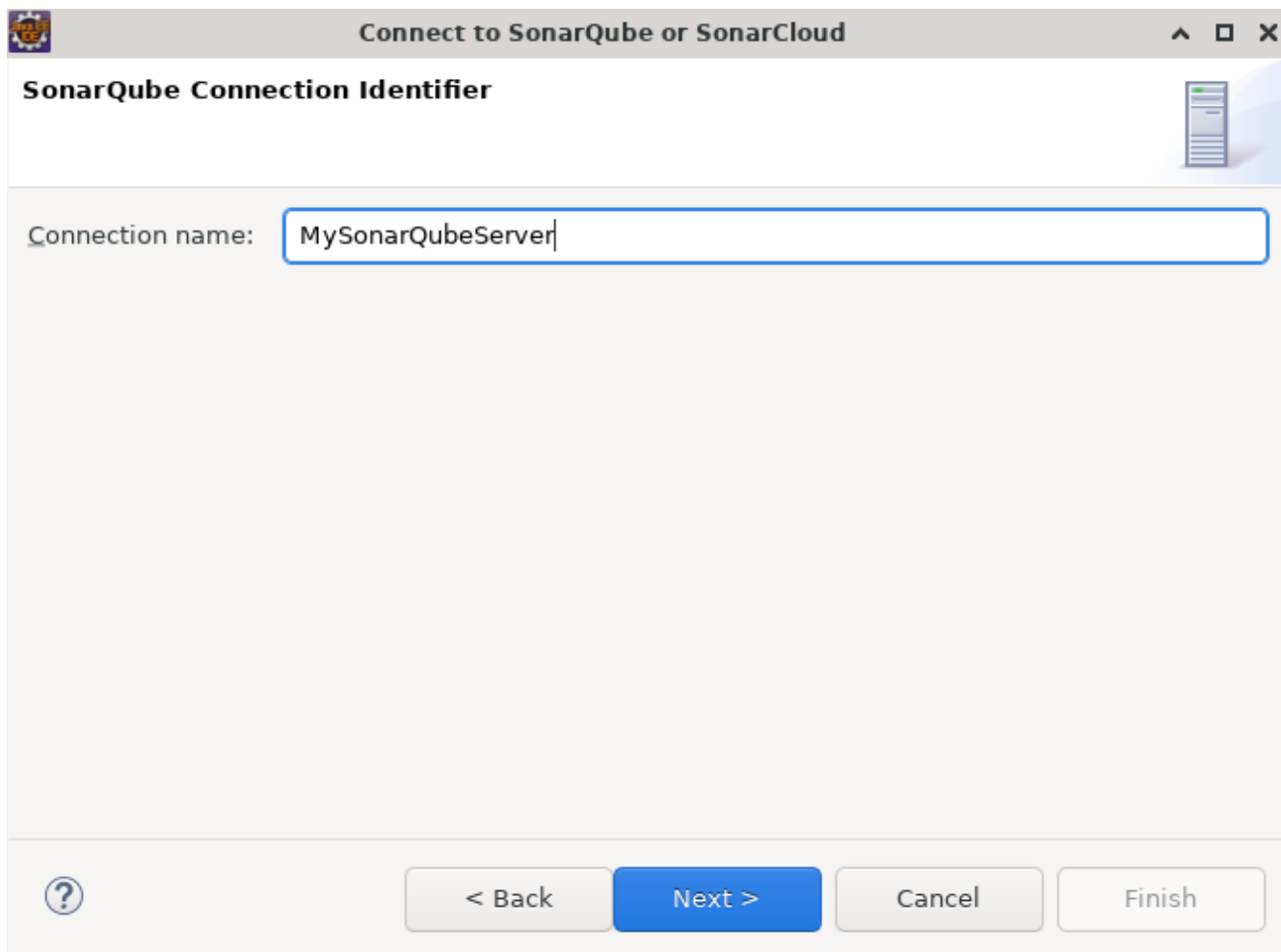
Vamos a poner el Token que generamos en el laboratorio anterior del usuario **user1**.



Pulsamos **Next**

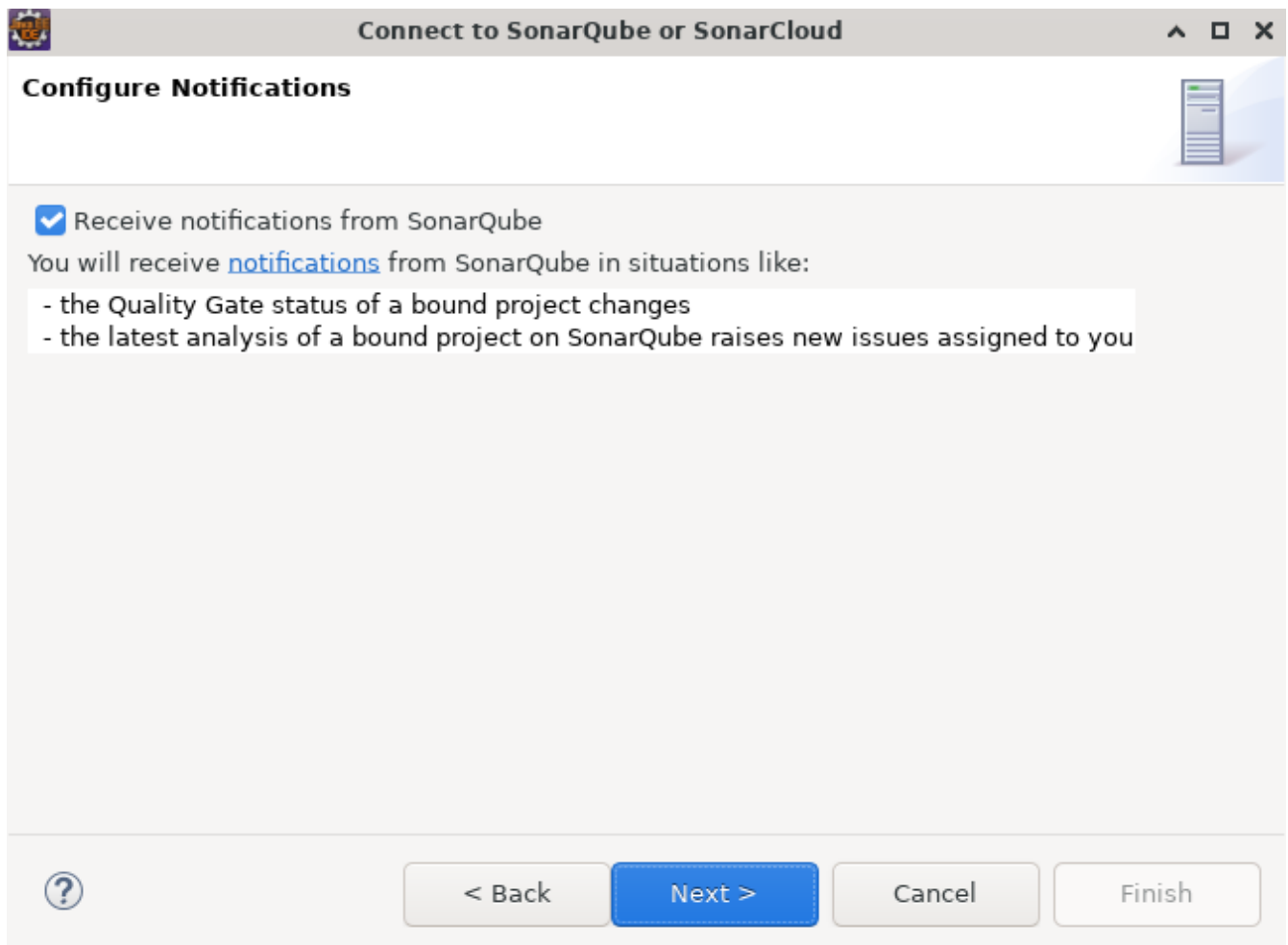
Indicamos en la siguiente pantalla el nombre de la conexión para poder hacer referencia a la misma.

Indicamos como nombre **MySonarQubeServer**



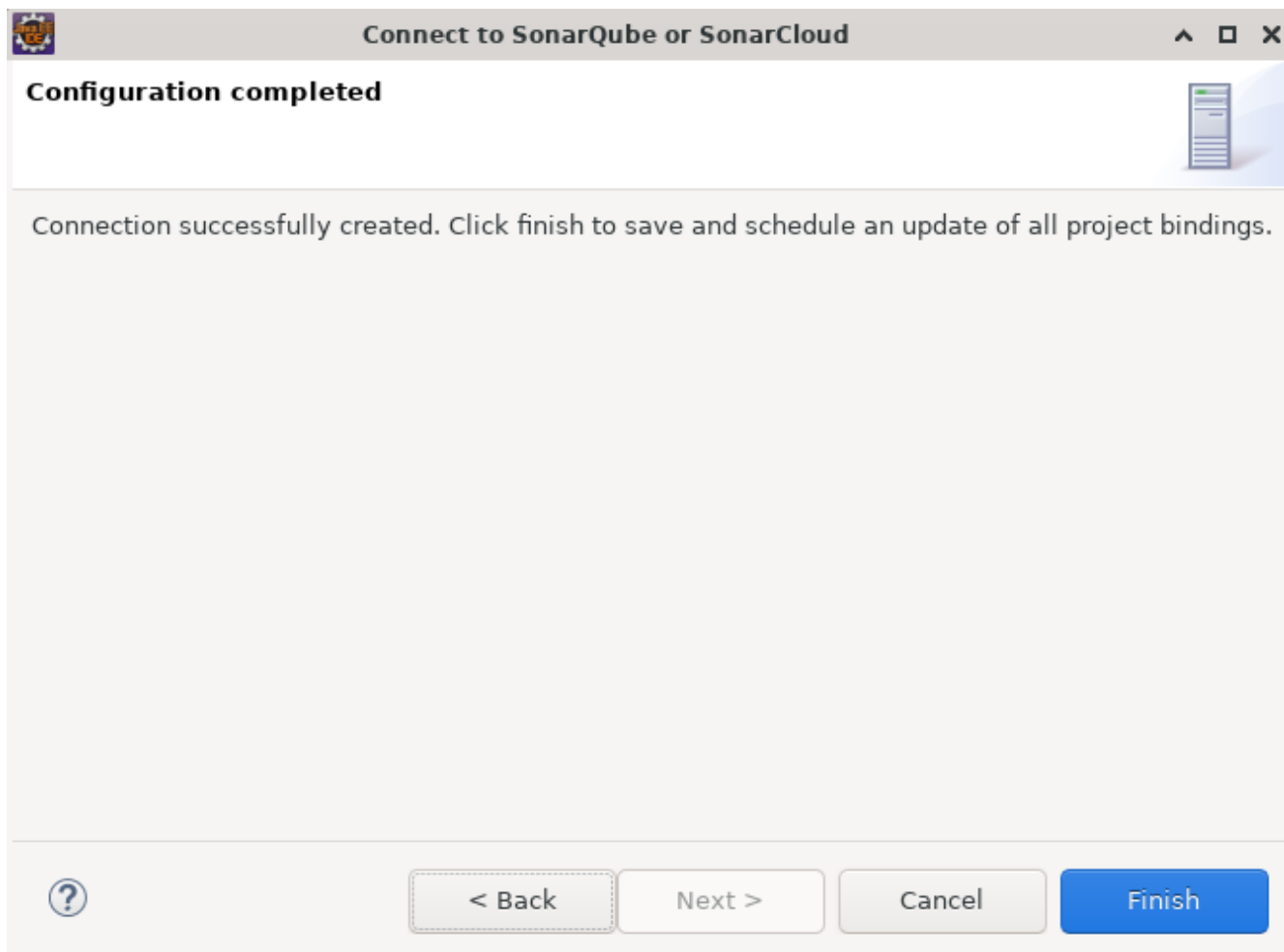
Pulsamos **Next**

En esta pantalla, vamos a indicar que sí queremos recibir notificaciones desde el servidor SonarQube.



Pulsamos **Next**

Si todo ha ido bien, deberíamos de observar una pantalla como esta:



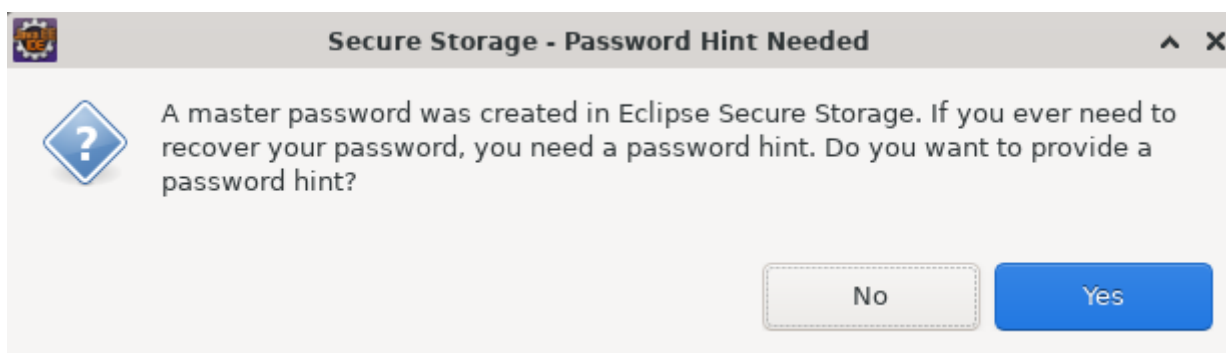
Pulsamos **Finish**

Nos aparecerá una pantalla, específica para el IDE eclipse, donde se nos requiere de una clave de acceso para el Keyring, el lugar de almacenamiento seguro para datos sensibles que tiene Eclipse.

Indicamos como contraseña: **sonarqube**

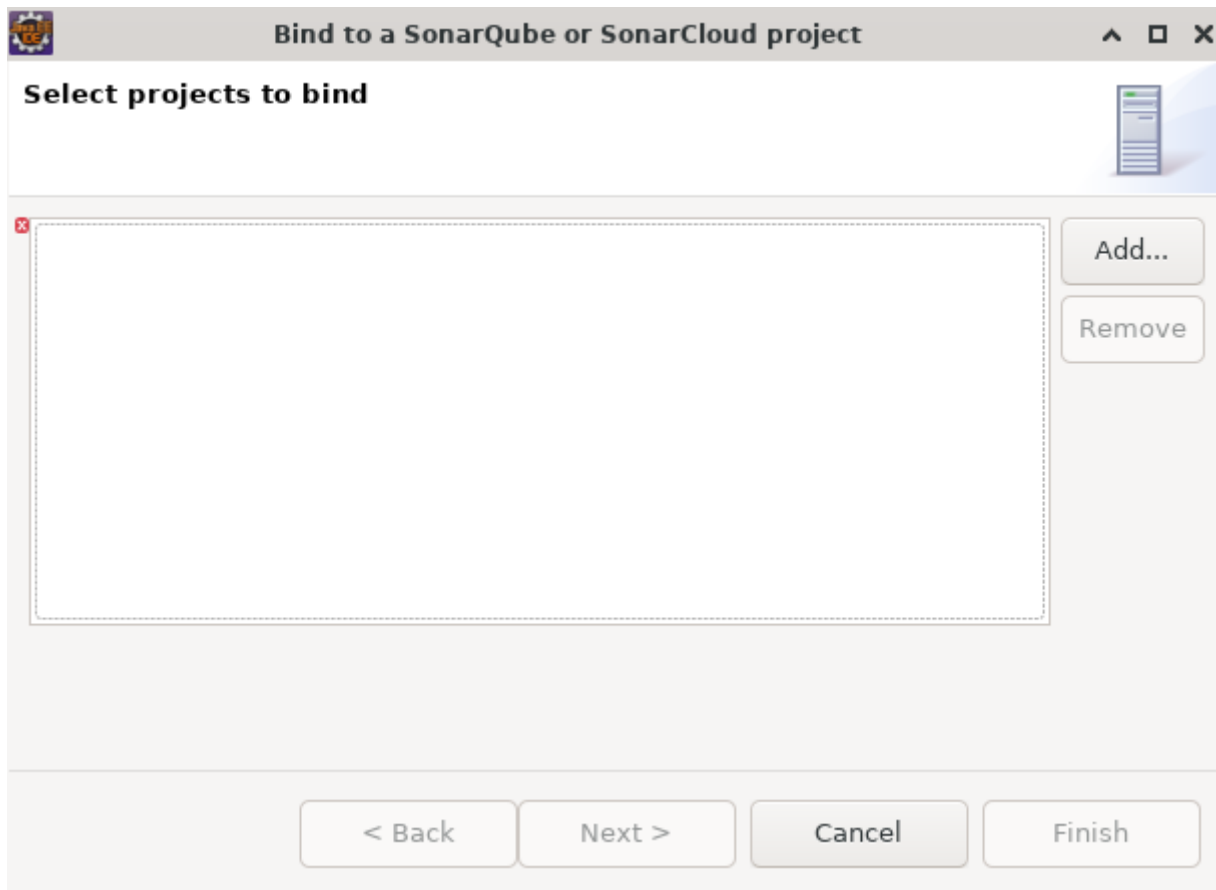
Seguidamente, nos aparecerá otra ventana donde nos preguntará si queremos suministrar alguna pista de sugerencia por si nos olvidamos de la clave.

Indicamos **No**

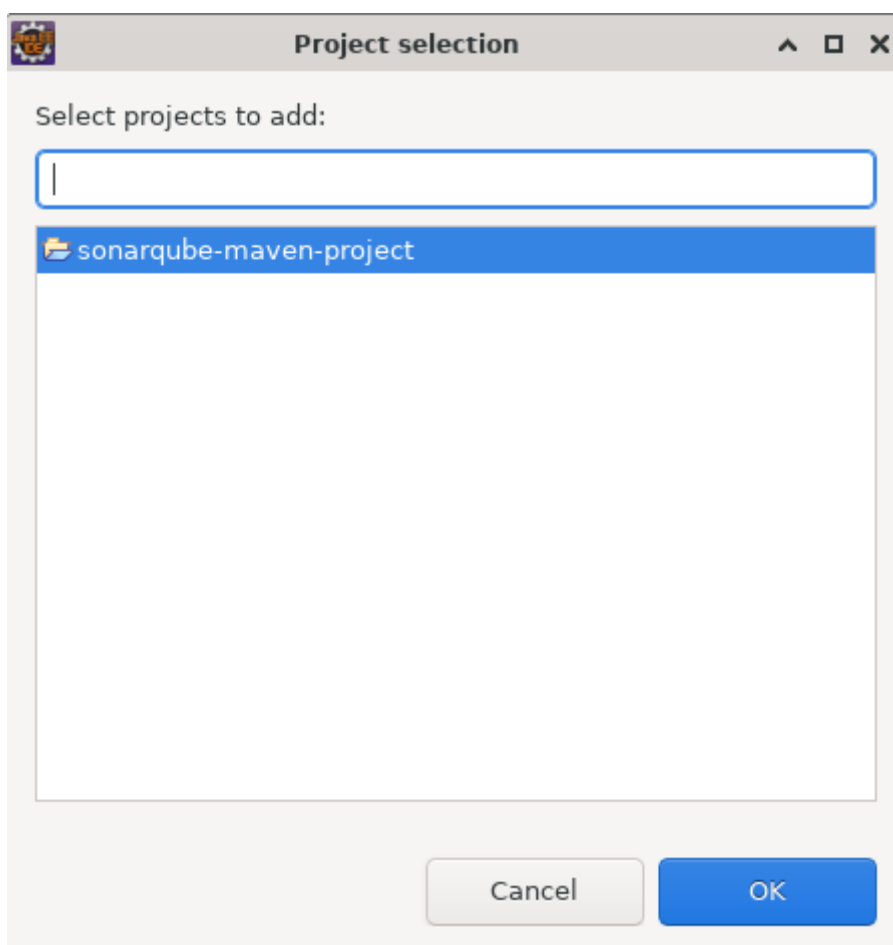


Seguidamente, vamos a llevar a cabo la operación de **Binding**, que consistirá en conectar el proyecto local con SonarQube.

Hacemos clic en el área blanca y pulsamos sobre el botón **Add**

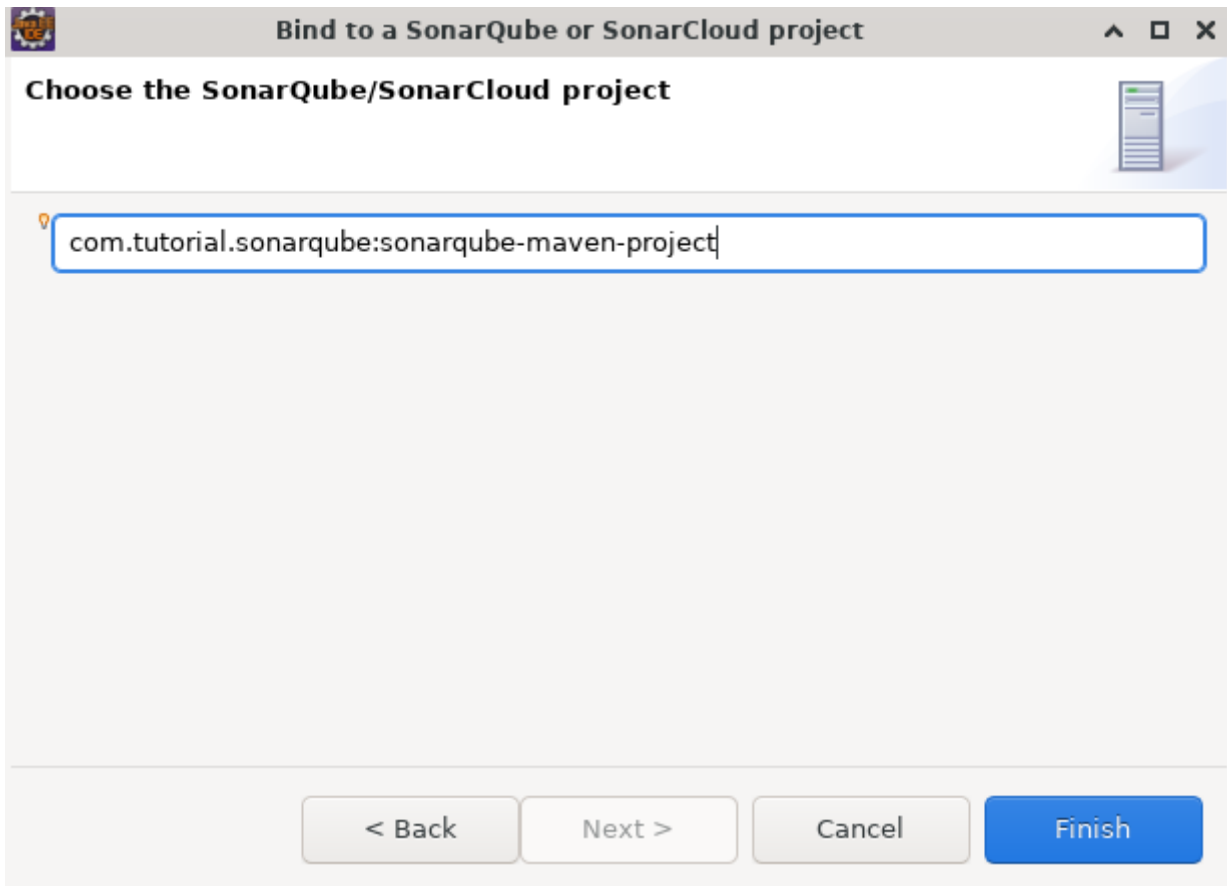


Seleccionamos el proyecto y pulsamos el botón de **OK**



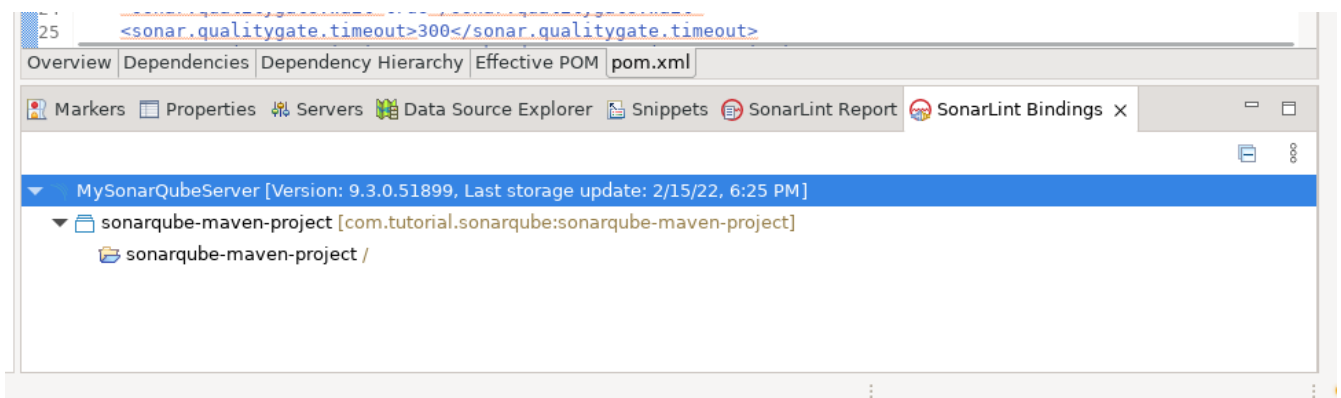
Seguidamente, seleccionando el proyecto, pulsamos el botón de **Next**

Y por último, pulsamos sobre el botón de **Finish**



Pulsamos el botón de **Finish**

Estando dentro del IDE, deberíamos de observar como efectivamente, disponemos de conexión con el servidor de SonarQube.



### 5.1.5. Creando un nuevo proyecto en SonarQube

Vamos a crear un nuevo proyecto en SonarQube con el nombre **java-project-eclipse-sonarqube-integration**.

Accedemos en SonarQube a la opción de **Administration > Projects > Management > Create Project**

Indicamos la siguiente configuración:



- **Name**
  - java-project-eclipse-sonarqube-integration
- **Key**
  - java-project-eclipse-sonarqube-integration
- **Visibility**
  - Private

## Create Project

All fields marked with \* are required

**Name \***

**Key \***

**Visibility**

☐

Public

☒

Private

Create

Cancel

Pulsamos sobre el botón **Create**

A continuación, vamos a vincular el Grupo **development** con este proyecto.

Accedemos en SonarQube a la opción de **Administration > Projects > Management > (Rueda dentada) > Edit Permissions**

<input type="checkbox"/>	<a href="#">java-project-eclipse-sonarqube-integration</a>	<b>PRIVATE</b>	java-project-eclipse-sonarqube-integration	—
<input type="checkbox"/>	<a href="#">sonarqube-maven-project</a>		com.tutorial.sonarqube:sonarqube-maven-project	F  Edit Permissions Apply Permission Template
<input type="checkbox"/>	<a href="#">sonarqube-maven-project-by-properties</a>		sonarqube-maven-project-by-properties	Feb 15, 2022

Marcamos todos los permisos para del grupo **development** con el proyecto:

java-project-eclipse-sonarqube-integration ☆ master

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

### Permissions

Grant and revoke project-level permissions. Permissions can be granted to groups or individual users. This project is private. Only authorized users can browse and see the source code.

☐ Public ☒ Private [Apply Permission Template](#)

	All	Users	Groups	Search for users or groups	Browse	See Source Code	Administer Issues	Administer Security Hotspots	Administer	Execute Analysis
<b>sonar-administrators</b> System administrators					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<b>Administrator</b> admin					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Anyone</b>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>User1</b> user1 user1@user1.com					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>User2</b> user2 user2@user2.com					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>User3</b> user3 user3@user3.com					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>User4</b> user4 user4@user4.com					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>development</b> Development group for SonarQube					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<b>sonar-users</b>										

### 5.1.6. Ajuste el pom.xml (Sonar Properties)

Para este nuevo escenario, vamos a indicar la siguiente configuración en nuestro archivo **pom.xml** de nuestro proyecto.



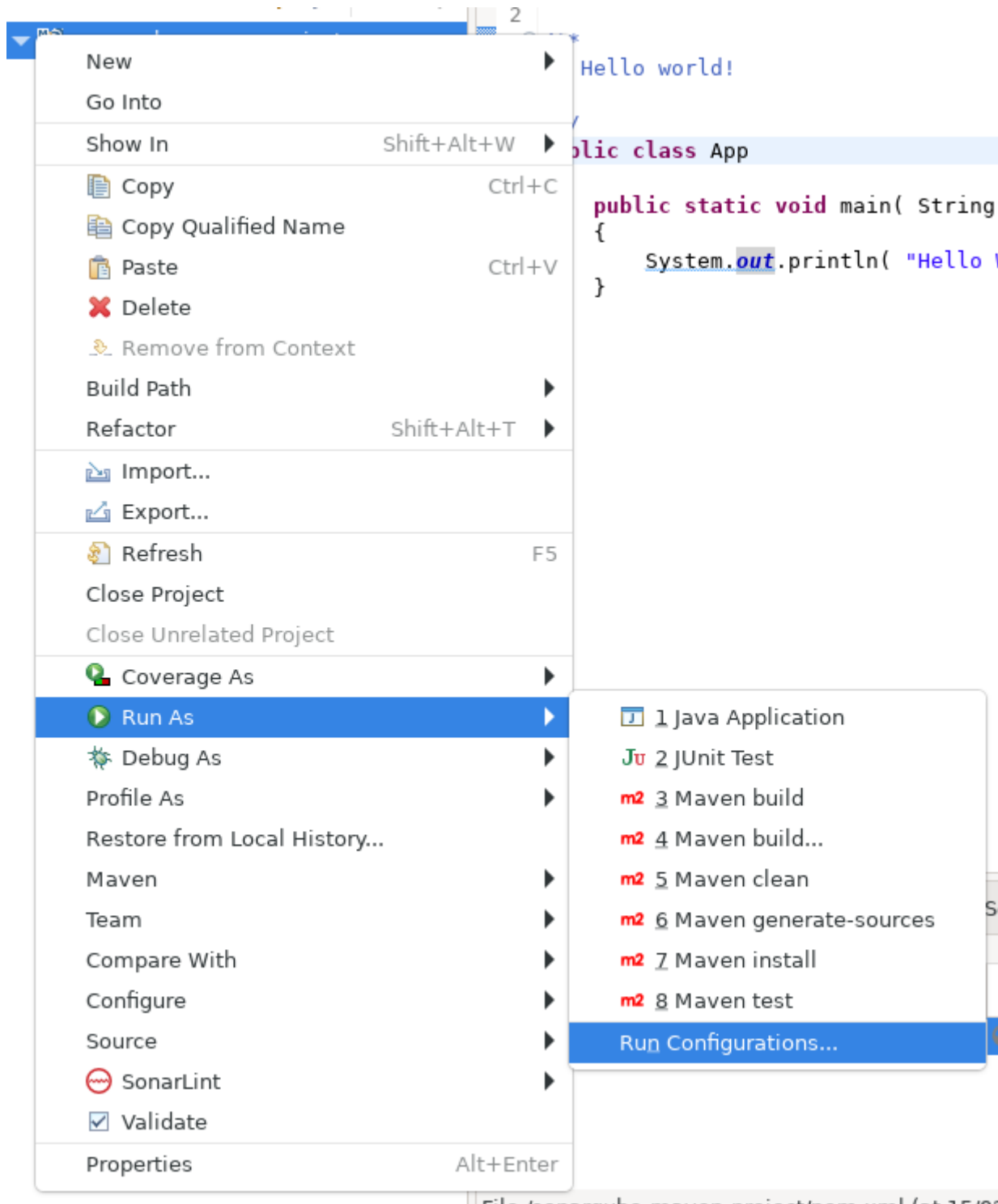
Indicamos el token del usuario **user1**.

```
<properties>
...
<sonar.projectKey>java-project-eclipse-sonarqube-integration</sonar.projectKey>
<sonar.projectName>java-project-eclipse-sonarqube-integration</sonar.projectName>
<sonar.login>d46ab3fe979b1d5c226836f10f0829e93917cd6a</sonar.login>
<sonar.log.level>INFO</sonar.log.level>
<sonar.qualitygate.wait>true</sonar.qualitygate.wait>
<sonar.qualitygate.timeout>300</sonar.qualitygate.timeout>
<sonar.projectDescription>Java Project Eclipse Integration</sonar.projectDescription>
<sonar.host.url>http://localhost:9002</sonar.host.url>
...
</properties>
```

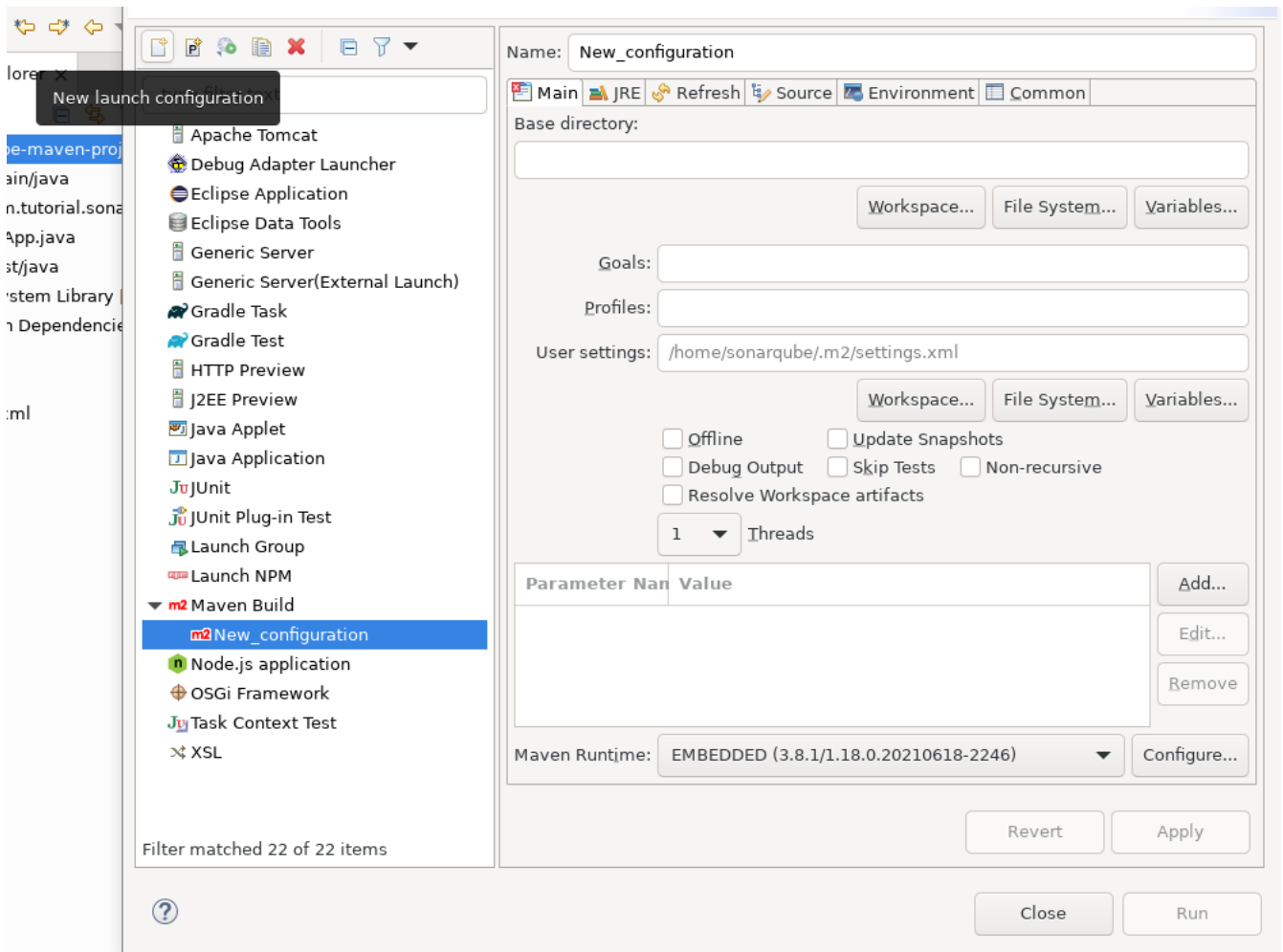
### 5.1.7. Configuración de la fase de Maven para Sonar

Ahora, vamos a crear en el IDE una configuración directa para ejecutar análisis contra SonarQube directamente desde el propio IDE, a través de Maven.

En primer lugar, en nuestro proyecto, pulsamos botón derecho del ratón **Run As > Run Configurations**



Seleccionamos el tipo de configuración que vamos a crear de tipo **Maven Build**, y pulsamos sobre el botón superior izquierdo **New launch configuration**



Indicamos los siguientes datos:

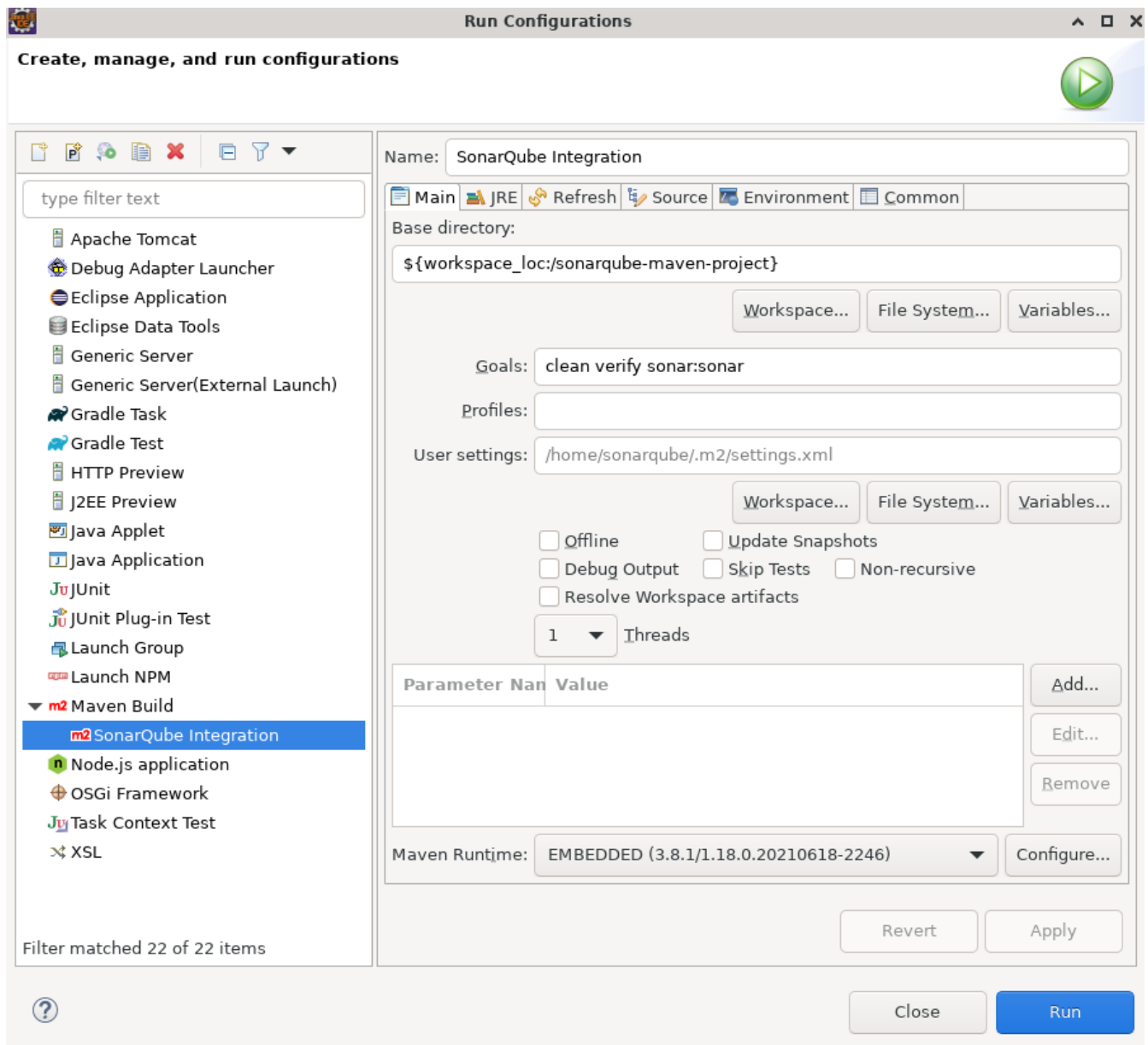
- **Name**
  - SonarQube Integration
- **Goals**
  - clean verify sonar:sonar
- **Base directory**
  - \${workspace\_loc:/sonarqube-maven-project}

Pulsamos sobre el botón **Apply** y luego **Close**

### 5.1.8. Enviando análisis a SonarQube

Seleccionamos nuestro proyecto, botón derecho **Run As > Run Configurations...**

Elegimos la configuración que hemos creado y pulsamos sobre el botón **Run**



Si todo ha ido bien, en la consola de Eclipse deberíamos de observar una traza de este estilo:

```
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=0ms
[INFO] Analysis report generated in 372ms, dir size=119.7 kB
[INFO] Analysis report compressed in 113ms, zip size=18.8 kB
[INFO] Analysis report uploaded in 298ms
[INFO] ----- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] QUALITY GATE STATUS: PASSED - View details on http://localhost:9002/dashboard?id=java-project-eclipse-sonarqube-integration
[INFO] Analysis total time: 1:18.149 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:38 min
[INFO] Finished at: 20XX-0X-15T19:13:16+01:00
[INFO] -----
```

Finalmente si en Eclipse consultamos el proyecto, observaremos que efectivamente ahora contiene toda la información escaneada del proyecto:

To benefit from more of SonarQube's features, [set up analysis in your favorite CI.](#)

## QUALITY GATE STATUS

**Passed**

All conditions passed.

## MEASURES

### New Code

### Overall Code

0

Bugs

Reliability **A**

0

Vulnerabilities

Security **A**

1

Security Hotspots

0.0% Reviewed

Security Review **E**

10min

Debt

2

Code Smells

Maintainability **A**

0.0%

Coverage on 2 Lines to cover

1

Unit Tests

0.0%

Duplications on 81 Lines

0

Duplicated Blocks

# Capítulo 6. Quality Gates

Mediante el concepto de Quality Gates, podemos hacer cumplir una política de calidad en nuestra empresa respondiendo a una pregunta clave:

¿Está mi proyecto listo para su lanzamiento/puesta en producción?

Para poder responder a esta pregunta, definimos un conjunto de condiciones contra las cuales se miden los proyectos.

Por ejemplo:

- No hay nuevos problemas de bloqueo
- Cobertura de código en código nuevo superior al 80%

Idealmente, todos los proyectos utilizarán la misma puerta de calidad, pero eso no siempre es práctico.

Por ejemplo, podemos encontrar que:

- La implementación tecnológica difiere de una aplicación a otra (es posible que no necesitemos la misma cobertura de código en código nuevo para aplicaciones Web o aplicaciones de escritorio).
- Deseamos garantizar requisitos más estrictos en algunas de nuestras aplicaciones.

Es por eso que se pueden definir tantas puertas de calidad como necesitemos.

Puede acceder a la página de Quality Gates desde el menú superior. Desde aquí puede definir y gestionar sus Quality Gates.

## 6.1. Utilizando la mejor configuración de Quality Gate

La Quality Gate **Sonar way** lo proporciona SonarSource, está activado de forma predeterminada y se considera integrado y de solo lectura.

Este Quality Gate representa la mejor manera de implementar el concepto de la filosofía **Clean Code** centrándose en el código nuevo que va apareciendo a lo largo del tiempo.

Con cada lanzamiento de SonarQube, se ajusta automáticamente esta puerta de calidad predeterminada de acuerdo con las capacidades de SonarQube configuradas.

Con Quality Gate, podemos aplicar indicadores en función de las métricas del código general y del código nuevo como:

- Confiabilidad
- Seguridad
- Revisión de seguridad y capacidad de mantenimiento

Estas métricas son parte de la puerta de calidad predeterminada.

Tenemos que tener en cuenta que, si bien la calidad del código de test afecta nuestra Quality Gate, solo se mide en función de las métricas de mantenimiento y confiabilidad.

Los problemas de duplicidad de código y seguridad no se miden en el código de test.

Debe ajustar sus controles de calidad para que brinden comentarios claros a los desarrolladores que miran la página de su proyecto.

Debemos de recordar que las condiciones de Quality Gate deben usar valores diferenciales.

Por ejemplo, no tiene sentido verificar un valor absoluto como: El número de líneas de código es mayor que 1000.

Una buena referencia de valor referencial para nuestro código, podría ser por ejemplo el hecho de que el porcentaje de cobertura fuera superior a un 80%.

## 6.2. Quality Gate recomendada

El equipo de SonarQube recomienda la puerta de calidad de vía Sonar incorporada de serie para la mayoría de los proyectos.

Esta puerta de calidad se enfoca en mantener limpio el código nuevo, en lugar de gastar mucho esfuerzo en remediar el código antiguo.

El uso de esta Quality Gate ya está configurada como el perfil predeterminado.

## 6.3. Posibilidad de notificaciones

SonarQube dispone de un mecanismo de notificación de fallos, los usuarios pueden ser notificados cuando falla una Quality Gate.

## 6.4. Sobre la seguridad de acceso

Las Quality Gates pueden ser accedidas por cualquier usuario, incluso por usuarios anónimos.

Todos los usuarios pueden ver cada aspecto relacionado con la Quality Gate.

Para que los usuarios de SonarQube puedan llevar a cabo operaciones como la creación, edición o eliminación de Quality Gates, deben de disponer del permiso **Administer Quality Profiles and Gates**

## 6.5. Lab: Quality Gates

Mediante este laboratorio llevaremos a cabo la creación y ajuste de una Quality Gate personalizada para nuestro proyecto.

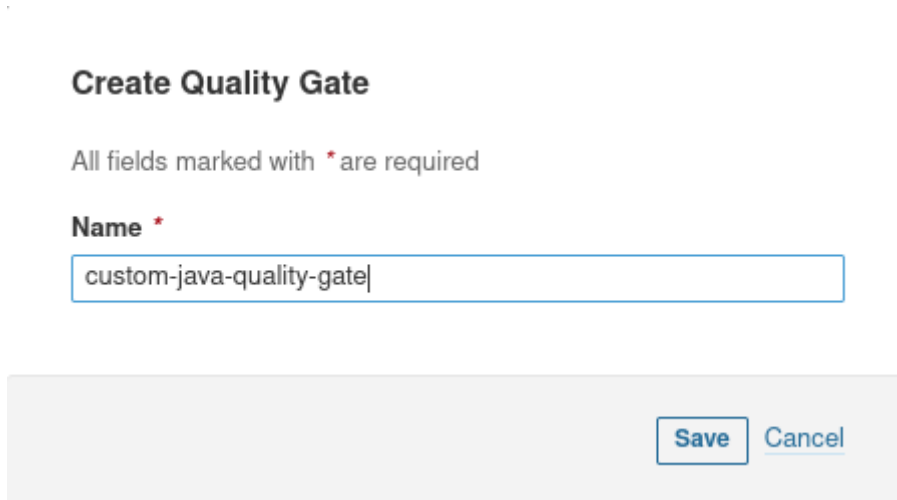


### 6.5.1. Creando una nueva QualityGate

Desde el menú principal hacemos clic en la opción **Quality Gates > Create**

Indicamos la siguiente propiedad:

- **Name**
  - custom-java-quality-gate



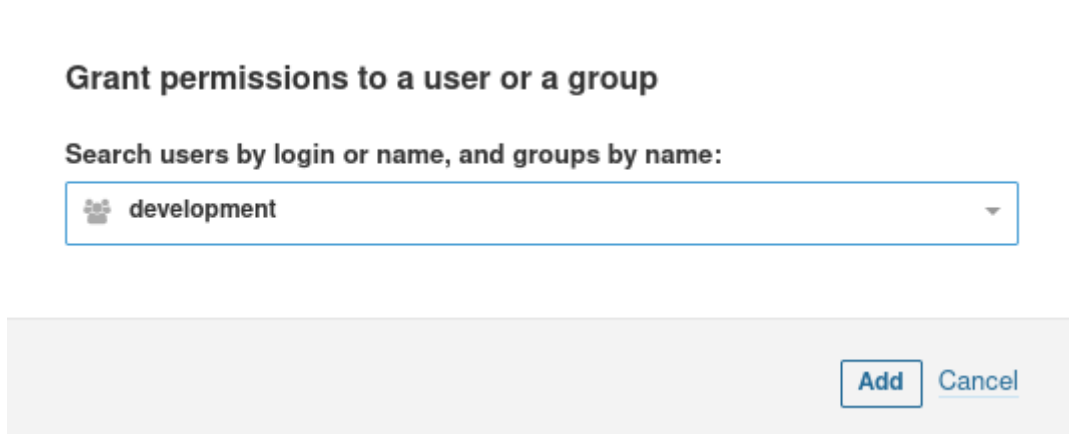
Pulsamos el botón de **Save**

### 6.5.2. Asignando permisos

Una vez dentro de detalle de la QualityGate, pulsamos sobre la opción **Grant permissions to a user or a group**

La idea es otorgarle permisos al grupo **development**.

Buscamos el grupo en la ventana del buscador y pulsamos el botón **Add**



### 6.5.3. Asignando condiciones

Nuestro QualityGate va a tener una serie de condiciones que lo harán característico.

Dentro del detalle del QualityGate, pulsamos sobre la opción **Add Condition**

Nos aparecerá una ventana que nos permitirá elegir si la condición de análisis se va a tener en cuenta únicamente para el código nuevo que aparezca en el proyecto desde el momento en que el QualityGate entra en operación, o bien, si queremos que se tenga en cuenta para todo el código existente del proyecto.

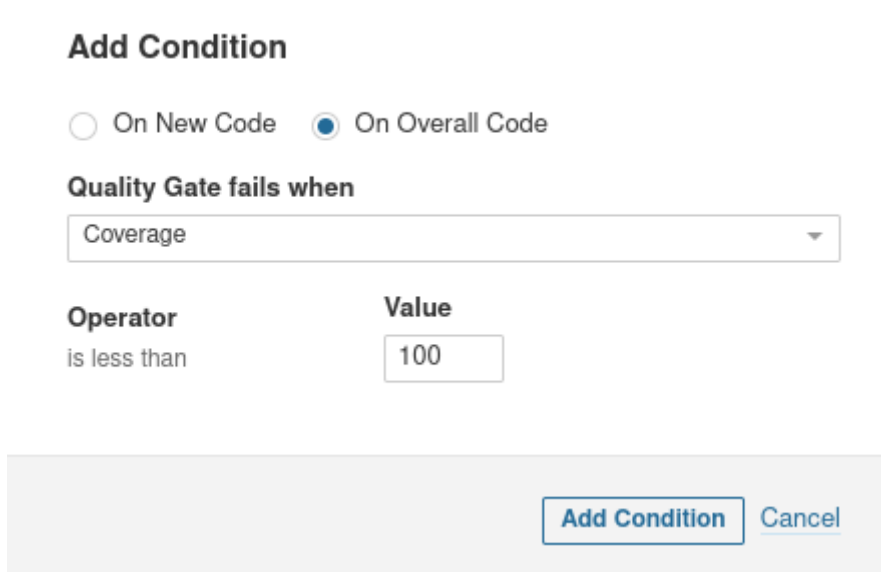
Indicamos la opción de **On Overall Code**

Adicionalmente, aparecerá un selector con título **Quality Gate fails when**.

En ese selector podremos elegir una métrica de las distintas categorías que nos ofrece SonarQube.

De la categoría **Coverage**, seleccionamos la opción **Coverage**

Vamos a indicar un valor un tanto exagerado, ya que vamos a indicar que si el porcentaje de cobertura es menor a 100, la QualityGate falle.



**Add Condition**

☐ On New Code ☒ On Overall Code

**Quality Gate fails when**

Coverage

**Operator**  
is less than

**Value**  
100

Add Condition Cancel

Pulsamos sobre **Add Condition**

#### 6.5.4. Asignando la QualityGate al proyecto

Dentro del detalle de la QualityGate, en el selector **Projects**, buscamos el proyecto que comience por **java-project**.

Una vez nos aparezca en el listado, hacemos clic y lo seleccionamos, nuestra QualityGate debería de tener un aspecto como este:

Quality Gates ⓘ

Create

custom-java-quality-gate

Sonar way

DEFAULT

BUILT-IN

custom-java-quality-gate

Rename

Copy

Set as Default

Delete

Conditions ⓘ

Add Condition

Conditions on Overall Code

Metric	Operator	Value	Edit	Delete
Coverage	is less than	100%		

Projects ⓘ

With Without All

Q java-project

☒ java-project-eclipse-sonarqube-integration  
 java-project-eclipse-sonarqube-integration

1 of 1 shown

Permissions

Users with the global "Administer Quality Gates" permission and those listed below can manage this Quality Gate.

development

Grant permissions to a user or a group

### 6.5.5. Probando la QualityGate

Nuestro proyecto actualmente dispone de un 0% de cobertura, con lo cual, debería de fallar en análisis de Sonar debido a que la QualityGate vinculada con el proyecto no cumple los requisitos.

Ejecutamos el análisis de nuestro código contra SonarQube desde el IDE Eclipse, y obtenemos la siguiente traza:

```

[INFO] Analysis report uploaded in 99ms
[INFO] ----- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 37.672 s
[INFO] Finished at: 2022-02-16T12:07:13+01:00
[INFO] -----
[ERROR] Failed to execute goal org.sonarsource.scanner.maven:sonar-maven-plugin:3.9.1.2184:sonar (default-cli) on project sonarqube-maven-project: QUALITY GATE STATUS: FAILED - View details on http://localhost:9002/dashboard?id=java-project-eclipse-sonarqube-integration -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException

```

Adicionalmente, si visualizamos el proyecto en SonarQube, observaremos que efectivamente tras el último análisis, la QualityGate ha fallado:

To benefit from more of SonarQube's features, [set up analysis in your favorite CI.](#)

## QUALITY GATE STATUS

**Failed**

1 conditions failed

### On Overall Code

0.0% Coverage is less than 100%

## MEASURES

### New Code

Since February 15, 2...  
Started 17 hours ago

### Overall Code

0

New Bugs

Reliability **A**

0

New Vulnerabilities

Security **A**

0

New Security Hotspots

Reviewed

Security Review **A**

0

Added Debt

0

New Code Smells

Maintainability **A**

Coverage on 0 New Lines to cover

Duplications on 0 New Lines

## ACTIVITY

# Capítulo 7. Quality Profiles

Los perfiles de calidad son un componente central de SonarQube donde se definen conjuntos de reglas que, cuando se violan, generan problemas en nuestra base de código.

Por ejemplo: Los métodos no deben tener una complejidad cognitiva superior a 15 líneas.

Cada lenguaje de programación tiene su propio perfil de calidad.

Para administrar perfiles de calidad, desde la página principal del SonarQube, accedemos a la opción de **Quality Profiles**.

Aquí encontraremos los perfiles de calidad agrupados por idioma.

Idealmente, todos nuestros proyectos se medirán con el mismo perfil de calidad, pero eso no siempre es práctico.

Por ejemplo, podríamos encontrarnos con las siguientes situaciones:

- Tenemos diferentes requisitos técnicos entre distintos proyectos (por ejemplo, se pueden aplicar diferentes reglas a una aplicación backend Java cuyo cometido sea el de un microservicio que únicamente realiza operaciones de cálculo sin acceder a base de datos, y otra aplicación que sí que acceda a bases de datos).
- Tenemos la necesidad por ejemplo, de requisitos más estrictos para algunos proyectos como algún framework interno que esté desarrollando la empresa

Podemos definir tantos perfiles de calidad como necesitemos para satisfacer necesidades específicas.

## 7.1. Perfiles de calidad por defecto

Cada lenguaje de programación debe tener un perfil de calidad predeterminado (marcado con la etiqueta predeterminada).

Los proyectos que no están asignados explícitamente a perfiles de calidad específicos se analizan utilizando los perfiles de calidad predeterminados.

También hay al menos un perfil de calidad incorporado para cada lenguaje de programación.

SonarSource crea estos perfiles de calidad con reglas que generalmente se aplican a la mayoría de los proyectos.

Los perfiles de calidad de **Sonar Way** son un buen punto de partida para comenzar a analizar el código, y comienzan siendo los perfiles de calidad predeterminados para cada lenguaje de programación.

Sin embargo, es recomendable que copiemos este perfil por defecto para comenzar a ajustar las reglas. ¿Por qué?

- Los perfiles de calidad predeterminados no se pueden editar, por lo que no podremos

personalizar el **Sonar Way** a nuestro gusto.

- La hoja de ruta **Sonar Way** por defecto pasa a ser una línea de base general a partir de la cual llevar un seguimiento y mantenimiento de nuestras Quality Gates.
- Nuestra propia hoja de ruta **Sonar Way** puede ser actualizada con el tiempo para ajustar qué reglas se incluyen y ajustar la gravedad de las mismas.

## 7.2. Permisos en los perfiles de calidad

Por defecto, únicamente los usuarios con el permiso global **Administer Quality Profiles** pueden editar los perfiles de calidad.

SonarQube también permite a los usuarios con el permiso global **Administer Quality Profiles** otorgar a un experto o un grupo de expertos permisos para llevar a cabo la administración de dichos perfiles.

Los expertos designados únicamente dispondrán de permisos para ese perfil de calidad en concreto, no para todos los perfiles de calidad.

## 7.3. Lab: Quality Profiles

Mediante este laboratorio, realizaremos algunas operaciones de gestión con los perfiles de calidad.

### 7.3.1. Creando un nuevo QualityProfile

Desde el menú principal hacemos clic en la opción **Quality Profiles > Create**

Indicamos las siguientes propiedades:

- **Name**
  - custom-java-quality-profile
- **Language**
  - Java
- **Parent**
  - None

## New Profile

All fields marked with \* are required

Name \*

Language \*

Parent:

[Create](#)[Cancel](#)

Pulsamos el botón de **Create**





### 7.3.2. Asignando reglas

Como hemos indicado que no queremos heredar reglas del QualityProfile por defecto, debemos de indicar al menos alguna antes de proceder a vincular el QualityProfile con nuestro proyecto.

Hacemos clic en el indicador de las reglas inactivas de **Bugs**:

Quality Profiles / Java

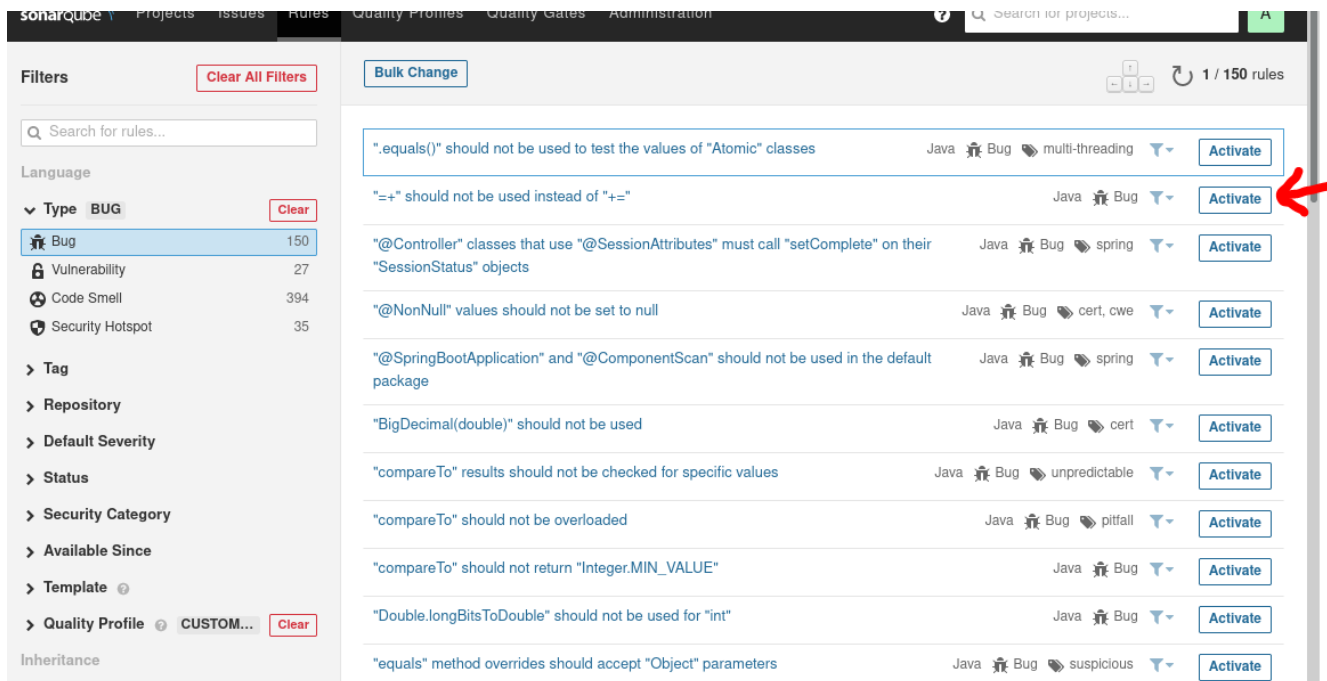
custom-java-quality-profile

Rules	Active	Inactive
<b>Total</b>	<b>0</b>	<b>606</b>
 Bugs	0	150
 Vulnerabilities	0	27
 Code Smells	0	394
 Security Hotspots	0	35

Activate More

Sonar way rules not included ? 460

Ahora, vamos a activar la regla que activa la detección de escritura anómala de los operadores =+



Indicamos un tipo de severidad que consideremos.

En nuestro caso, indicaremos la severidad de mayor gravedad: **Blocker**

### Activate In Quality Profile

#### Quality Profile

custom-java-quality-profile

#### Severity

! Blocker

Activate Cancel

Pulsamos sobre el botón **Activate**

### 7.3.3. Vinculando el QualityProfile con el proyecto

A continuación, desde el menú principal hacemos clic en la opción **Quality Profiles > Filter profiles by(Java) > custom-java-quality-profile**

Una vez en el detalle del QualityProfile, hacemos clic en la opción **Change Projects**

En el buscador que nos aparecerá, comenzamos a escribir **java-project**, seleccionamos el proyecto:



## Projects

WithWithoutAll

Q java-project

X

☒ Bulk Change

☒ java-project-eclipse-sonarqube-integration

java-project-eclipse-sonarqube-integration

1 of 1 shown

[Close](#)

Finalmente, el aspecto de configuración del QualityProfile debería de ser algo así:

custom-java-quality-profile

Updated: 6 minutes agoUsed: NeverChangelog⚙

Rules

	Active	Inactive
Total	1	605
Bugs	1	149
Vulnerabilities	0	27
Code Smells	0	394
Security Hotspots	0	35

Activate More

Sonar way rules not included459

Inheritance

Change Parent

custom-java-quality-profile1 active rules0 overridden rules

Projects

Change Projects

java-project-eclipse-sonarqube-integration1 of 1 shown

Permissions

Users with the global "Manage Quality Profile" permission can manage this Quality Profile.

Grant permissions to more users

### 7.3.4. Realizando ajustes en nuestro proyecto

Ahora, vamos a realizar algunos ajustes en nuestro proyecto, con la idea de:

- Cumplir el 100% de cobertura de código para aprobar la QualityGate
- Incluir una sentencia en el método del código fuente que vamos a probar, para que salte la detección del Bug de nuestro QualityProfile.

En primer lugar, vamos a cambiar el código de la clase **App.java** dejándolo con el siguiente contenido:

```
package com.tutorial.sonarqube;

public class App
{
    public static void main( String[] args )
    {
        App mainApp = new App();
        Integer sumResult = null;
        Integer incrementResult = null;

        sumResult = mainApp.sum(2, 2);
        incrementResult = mainApp.incrementUnits(1, sumResult);

        System.out.println("SUM RESULT: " + sumResult);
        System.out.println("INCREMENT RESULT: " + incrementResult);
    }

    private Integer sum(Integer number1, Integer number2) {
        return number1 + number2;
    }

    private Integer incrementUnits(Integer unitsToIncrement, Integer singleNumber) {
        return singleNumber += unitsToIncrement;
    }
}
```

Seguidamente, vamos a cambiar el código de la clase **AppTest.java** para que se lleven a cabo test en el 100% del código fuente:

```

package com.tutorial.sonarqube;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

public class AppTest
{
    @Test
    public void shouldBeReturn2SumOneAndOne()
    {
        App mainApp = new App();
        Integer sumResult = mainApp.sum(1, 1);

        assertTrue(sumResult.equals(2));
    }

    @Test
    public void shouldBeIncrementOneUnit() {
        App mainApp = new App();
        Integer incrementResult = null;

        incrementResult = mainApp.incrementUnits(1, 1);

        assertTrue(incrementResult.equals(1));
    }
}

```

Y ahora nos quedaría llevar a cabo el ajuste en el archivo **pom.xml** para incorporar el plugin **jacoco**, de forma que pueda generar información de cobertura y con esa información, informar a SonarQube.

Modificamos la sección **<build></build>** dejándola así:

```

<build>
  <plugins>
    <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
    <plugin>
      <artifactId>maven-clean-plugin</artifactId>
      <version>3.1.0</version>
    </plugin>
    <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-
bindings.html#Plugin_bindings_for_jar_packaging -->
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.1</version>

```

```

</plugin>
<plugin>
  <artifactId>maven-jar-plugin</artifactId>
  <version>3.0.2</version>
</plugin>
<plugin>
  <artifactId>maven-install-plugin</artifactId>
  <version>2.5.2</version>
</plugin>
<plugin>
  <artifactId>maven-deploy-plugin</artifactId>
  <version>2.8.2</version>
</plugin>
<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
<plugin>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.7.1</version>
</plugin>
<plugin>
  <artifactId>maven-project-info-reports-plugin</artifactId>
  <version>3.0.0</version>
</plugin>
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.7</version>
  <executions>
    <execution>
      <id>default-prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <!-- Si sólo queremos que se genere el informe de pruebas al generar mvn site, en lugar de al hacer mvn
test, indicamos esta execution únicamente
  <execution>
    <id>jacoco-site</id>
    <phase>post-integration-test</phase>
    <goals>
      <goal>report</goal>
    </goals>
  </execution>-->
    <execution>
      <id>jacoco-report</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
</plugins>
</build>

```

Agregamos la sección **<reporting></reporting>**, dentro de la sección **<project></project>**:

```

<project>
...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.7</version>
        <reportSets>
          <reportSet>
            <reports>
              <report>report</report>
            </reports>
          </reportSet>
        </reportSets>
      </plugin>
    </plugins>
  </reporting>
...
</project>

```

### 7.3.5. Ejecutando análisis contra SonarQube

Ahora, ejecutamos el siguiente comando para lanzar el análisis contra SonarQube para verificar que la QualityGate es cumplida:

```

$ mvn clean verify sonar:sonar

[INFO] Quality profile for java: custom-java-quality-profile
[INFO] Quality profile for xml: Sonar way
[INFO] ----- Run sensors on module java-project-eclipse-sonarqube-integration
[INFO] Load metrics repository
[INFO] Load metrics repository (done) | time=319ms
[INFO] Sensor JavaSensor [java]
[INFO] Configured Java source version (sonar.java.source): 7
[INFO] JavaClasspath initialization
[INFO] JavaClasspath initialization (done) | time=157ms
[INFO] JavaTestClasspath initialization
[INFO] JavaTestClasspath initialization (done) | time=11ms
[INFO] Java "Main" source files AST scan
[INFO] 1 source file to be analyzed
[INFO] Load project repositories
[INFO] Load project repositories (done) | time=172ms
[INFO] 0/1 files analyzed, current file: src/main/java/com/tutorial/sonarqube/App.java
[INFO] 1/1 source file has been analyzed
[INFO] Java "Main" source files AST scan (done) | time=12957ms
[INFO] Java "Test" source files AST scan
[INFO] 1 source file to be analyzed
[INFO] 1/1 source file has been analyzed
[INFO] Java "Test" source files AST scan (done) | time=1140ms
[INFO] No "Generated" source files to scan.
[INFO] Sensor JavaSensor [java] (done) | time=20506ms
[INFO] Sensor JaCoCo XML Report Importer [jacoco]
[INFO] 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations:
target/site/jacoco/jacoco.xml,target/site/jacoco-it/jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
[INFO] Importing 1 report(s). Turn your logs in debug mode in order to see the exhaustive list.

```

```

[INFO] Sensor JaCoCo XML Report Importer [jacoco] (done) | time=447ms
[INFO] Sensor CSS Rules [javascript]
[INFO] No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
[INFO] Sensor CSS Rules [javascript] (done) | time=23ms
[INFO] Sensor C# Project Type Information [csharp]
[INFO] Sensor C# Project Type Information [csharp] (done) | time=7ms
[INFO] Sensor C# Analysis Log [csharp]
[INFO] Sensor C# Analysis Log [csharp] (done) | time=367ms
[INFO] Sensor C# Properties [csharp]
[INFO] Sensor C# Properties [csharp] (done) | time=3ms
[INFO] Sensor SurefireSensor [java]
[INFO] parsing [/home/sonarqube/eclipse-workspace/app-backend/sonarqube-maven-project/target/surefire-reports]
[INFO] Sensor SurefireSensor [java] (done) | time=1548ms
[INFO] Sensor HTML [web]
[INFO] Sensor HTML [web] (done) | time=82ms
[INFO] Sensor XML Sensor [xml]
[INFO] 1 source file to be analyzed
[INFO] 1/1 source file has been analyzed
[INFO] Sensor XML Sensor [xml] (done) | time=3876ms
[INFO] Sensor Text Sensor [text]
[INFO] 3 source files to be analyzed
[INFO] 3/3 source files have been analyzed
[INFO] Sensor Text Sensor [text] (done) | time=242ms
[INFO] Sensor VB.NET Project Type Information [vbnet]
[INFO] Sensor VB.NET Project Type Information [vbnet] (done) | time=10ms
[INFO] Sensor VB.NET Analysis Log [vbnet]
[INFO] Sensor VB.NET Analysis Log [vbnet] (done) | time=455ms
[INFO] Sensor VB.NET Properties [vbnet]
[INFO] Sensor VB.NET Properties [vbnet] (done) | time=8ms
[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=22ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=419ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=1ms
[INFO] Analysis report generated in 980ms, dir size=97.8 kB
[INFO] Analysis report compressed in 405ms, zip size=17.5 kB
[INFO] Analysis report uploaded in 450ms
[INFO] ----- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] QUALITY GATE STATUS: PASSED - View details on http://localhost:9002/dashboard?id=java-project-eclipse-sonarqube-integration
[INFO] Analysis total time: 1:17.911 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 03:01 min
[INFO] Finished at: 202X-0X-16T14:15:16+01:00
[INFO] -----

```

Adicionalmente, observamos desde el dashboard de SonarQube, como nuestro proyecto se ha actualizado después del último análisis y ahora la QualityGate se cumple:

To benefit from more of SonarQube's features, [set up analysis in your favorite CI.](#)

## QUALITY GATE STATUS

**Passed**

All conditions passed.

## MEASURES

### New Code

Since February 15, 2...  
Started 19 hours ago

### Overall Code

1

New Bugs

Reliability **E**

0

New Vulnerabilities

Security **A**

0

New Security Hotspots

Reviewed

Security Review **A**

0

Added Debt

0

New Code Smells

Maintainability **A**

100%

Coverage on 2 New Lines to cover

0.0%

Duplications on 68 New Lines