INGENIERÍA DE SOFTWARE II

SonarQube

Docente:

Jairo Cortes

Estudiantes:

Lady Johana Mancera Bejarano

Andres Felipe Gonzalez Merchán

Universidad Cooperativa de Colombia

Facultad de Ingeniería

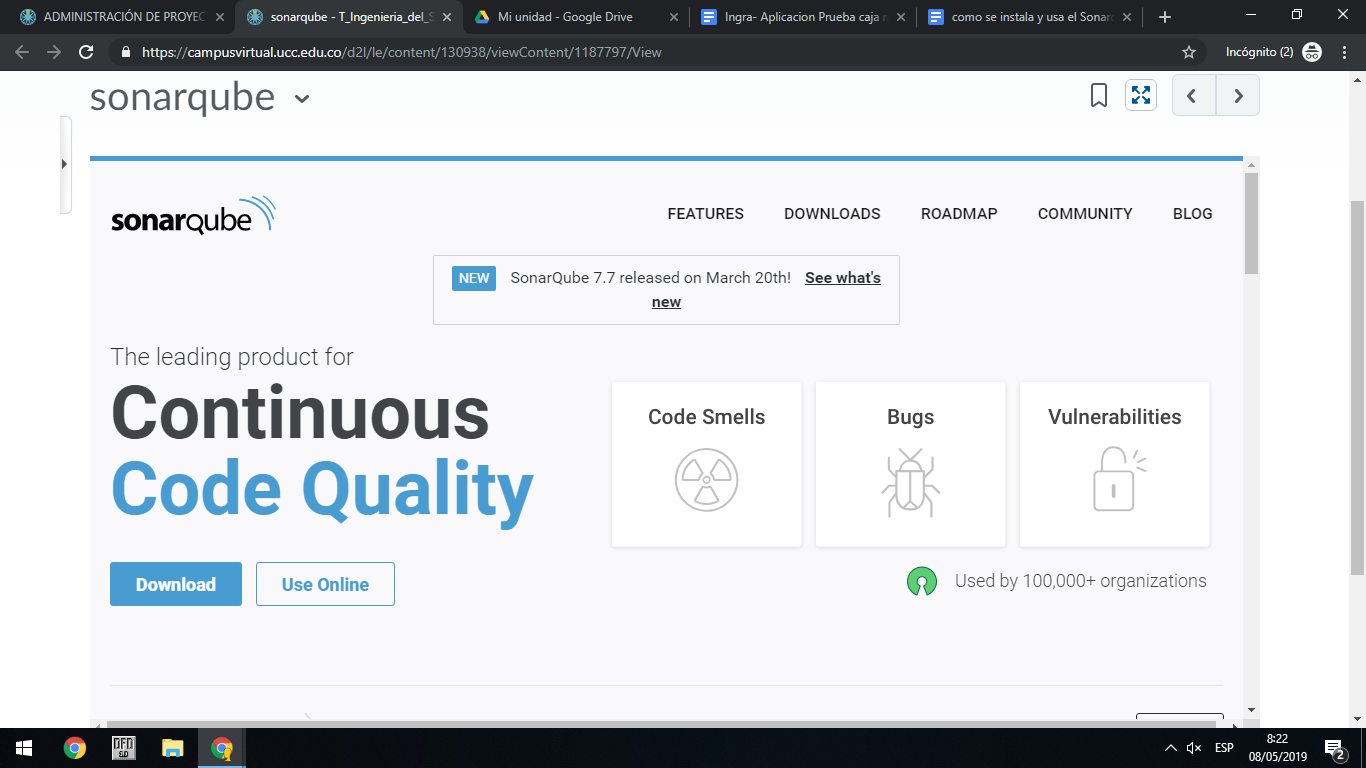
Ingeniería de Sistemas

Bogotá

2019

**Pasos para la descarga e instalación de SonarQube**

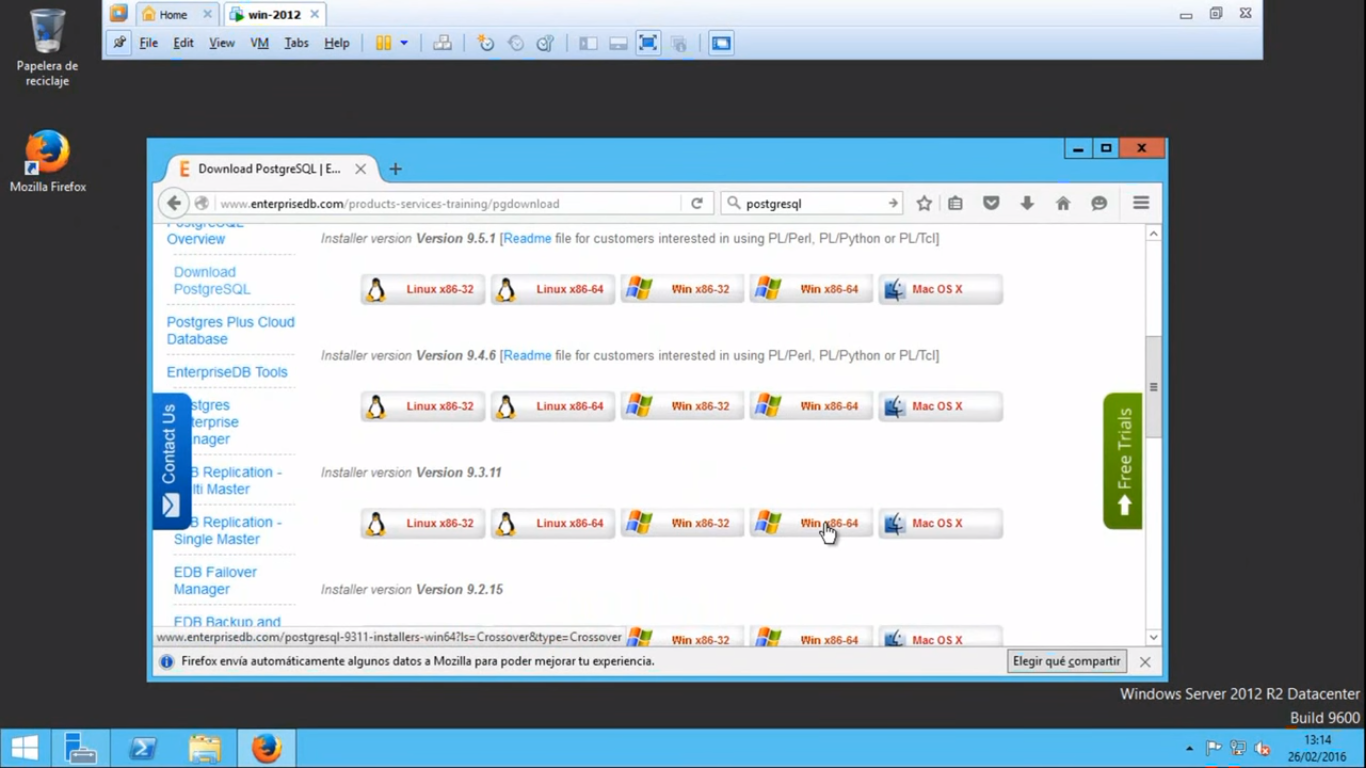
Entramos a la pagina principal de SonarQube para buscar el instalador:



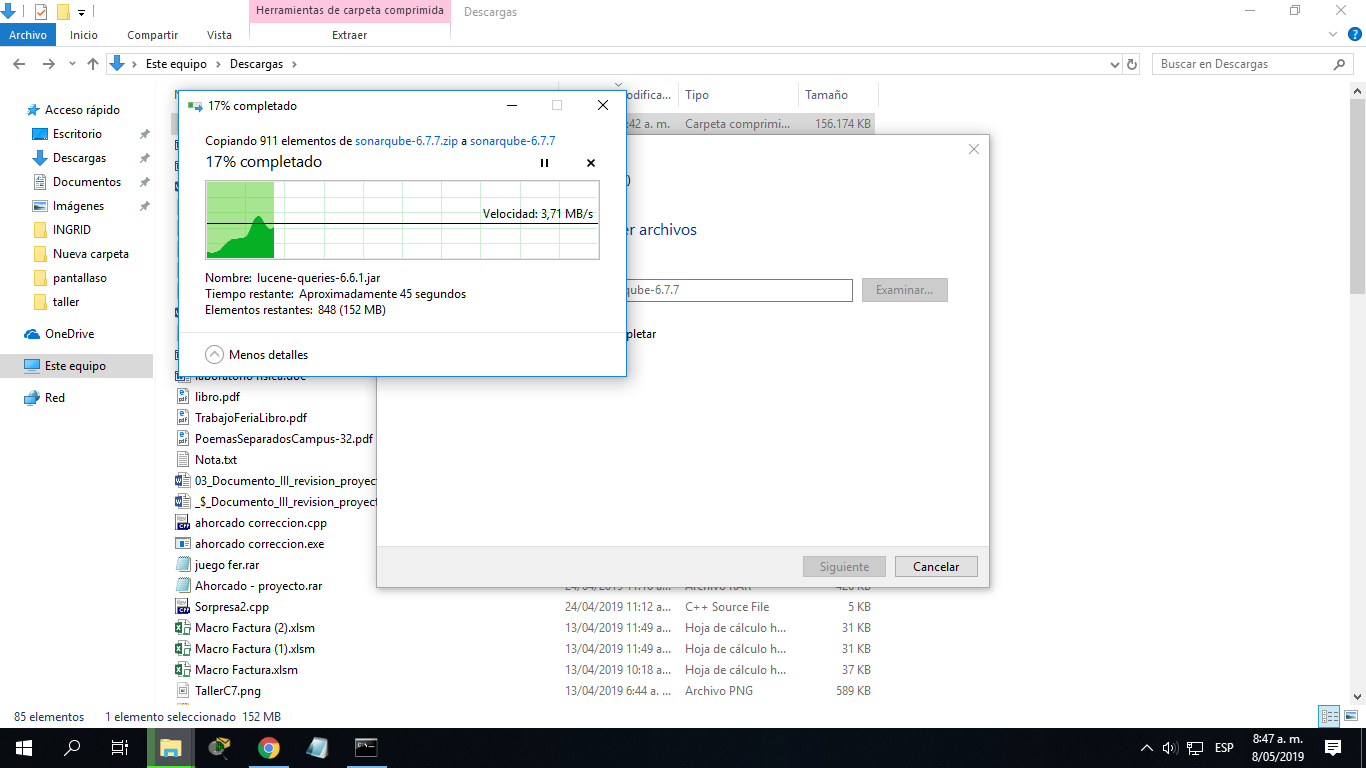
Para que el programa corra necesita un Motor de base de datos como por ejemplo postgresSQL

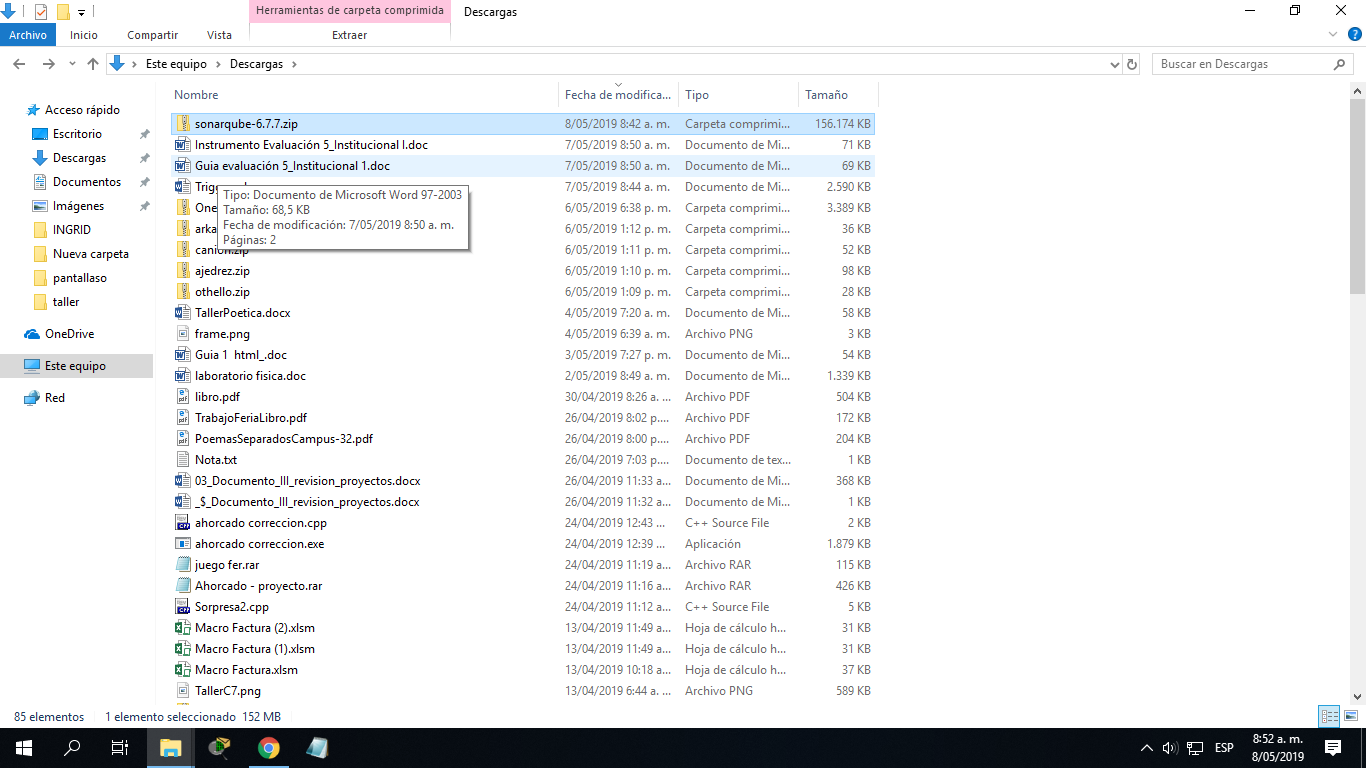




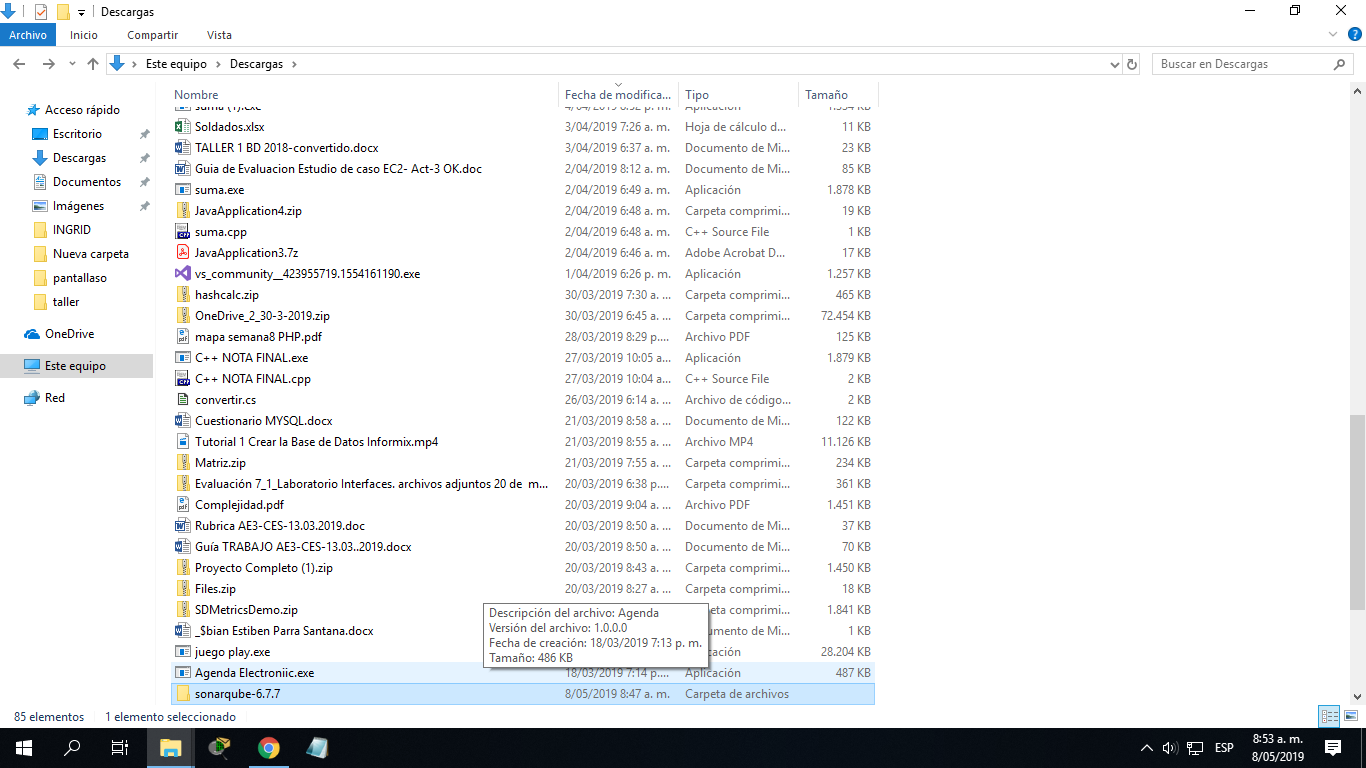


1. [**Descarga**](http://www.sonarqube.org/downloads/) y descomprime SonarQube (por ejemplo en «C:\sonarqube» o «/etc/sonarqube»)

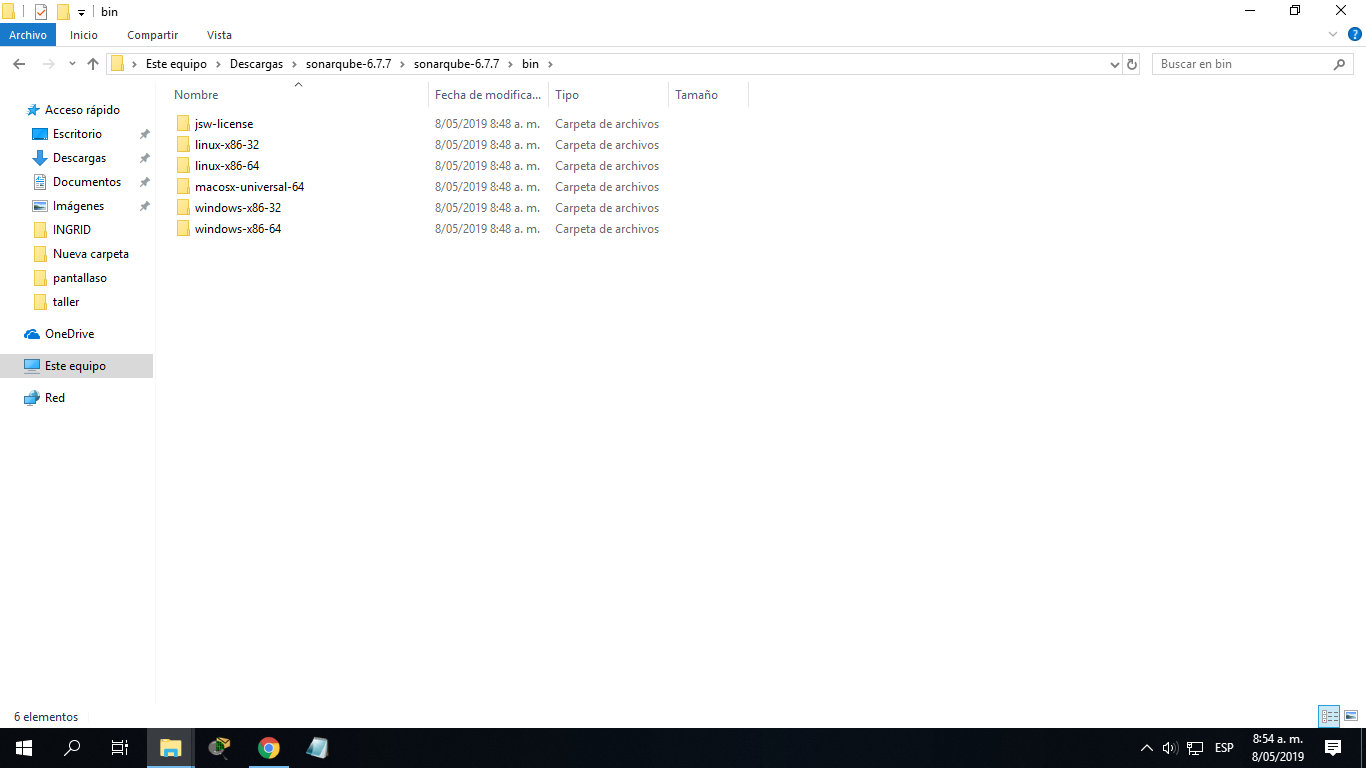


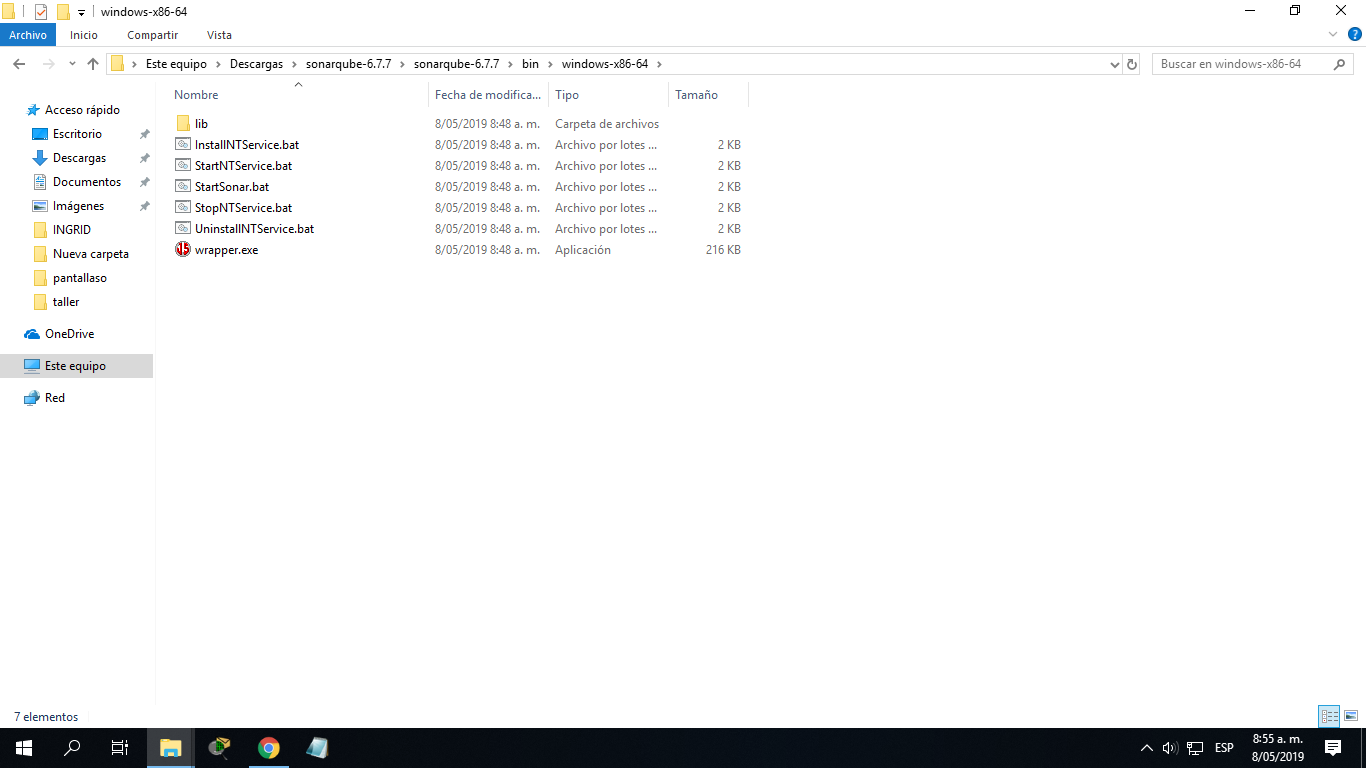


Entramos a la carpeta extraída

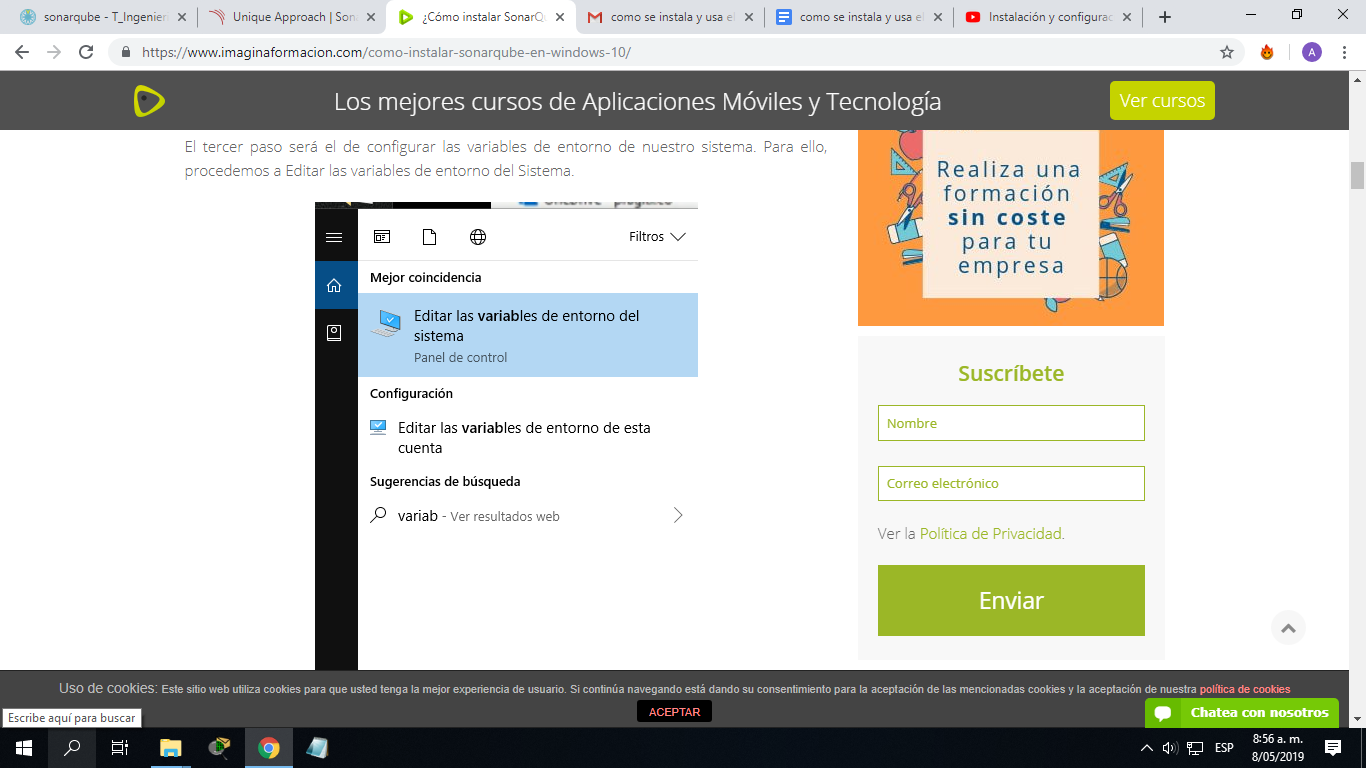


Ingresamos a bin



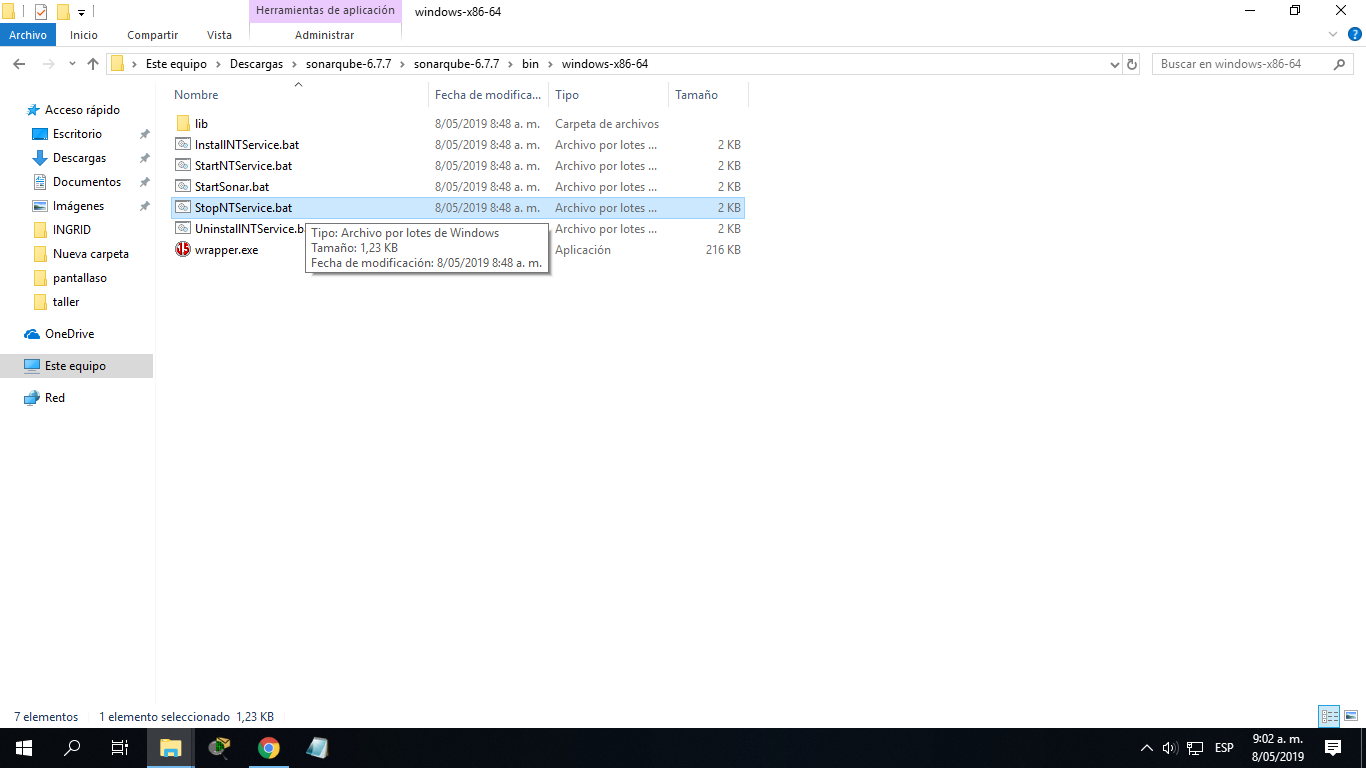


3. Tenemos que configurar las variables de entorno de nuestro sistema. Para ello, procedemos a Editar las variables de entorno del Sistema.

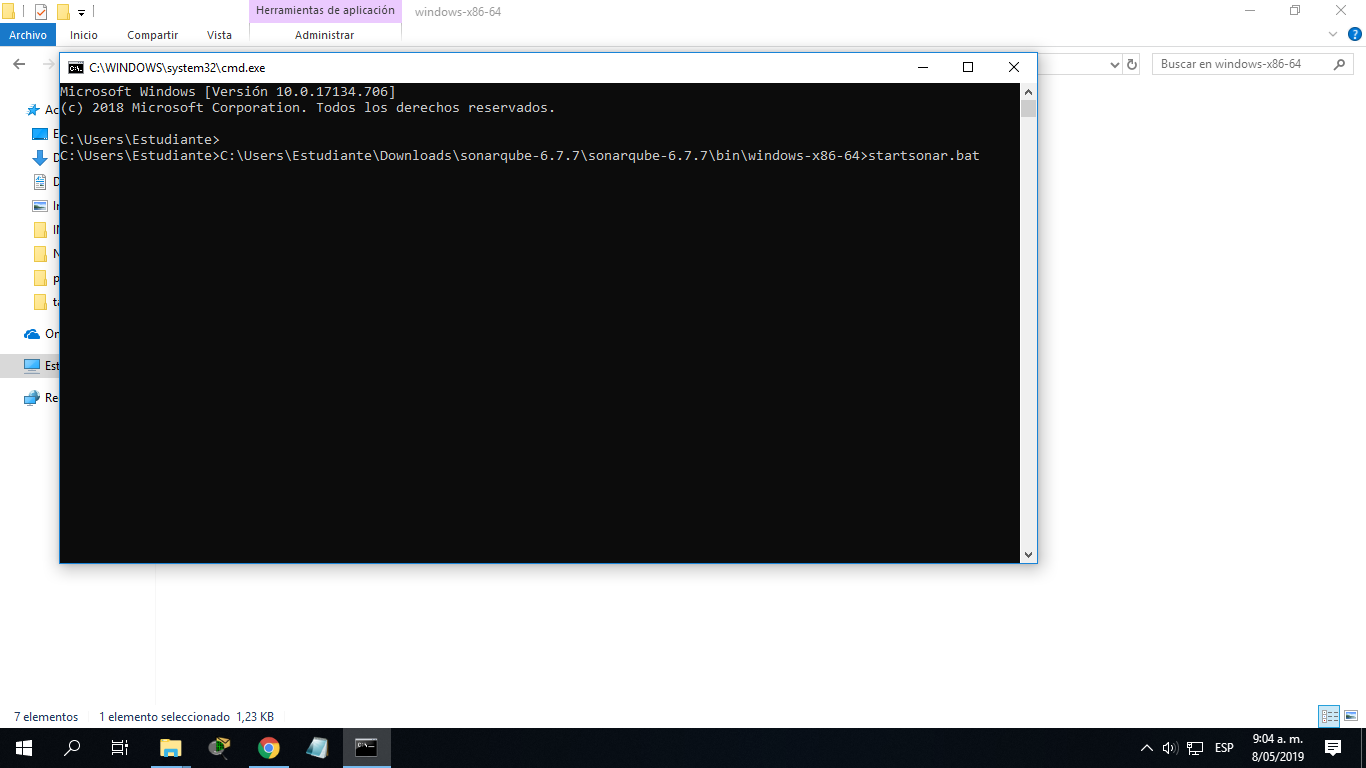


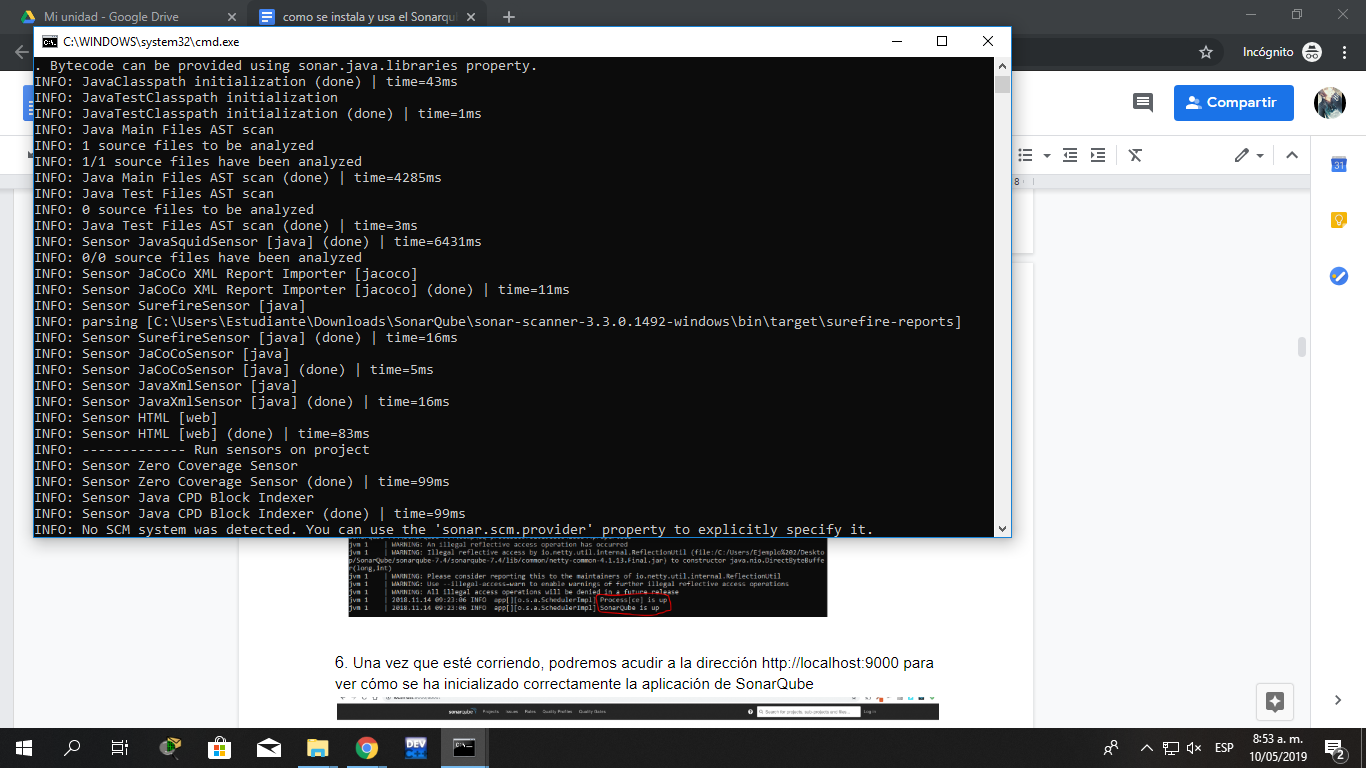
4. Verificamos que el SonaQube esta hubicado en la direccion correcta

5. Para comprobar que en efecto, SonarQube se inicia correctamente. Para ello, se navega con la consola hasta la carpeta donde se aloja el archivo startsonar.bat y se ejecuta de la siguiente manera:

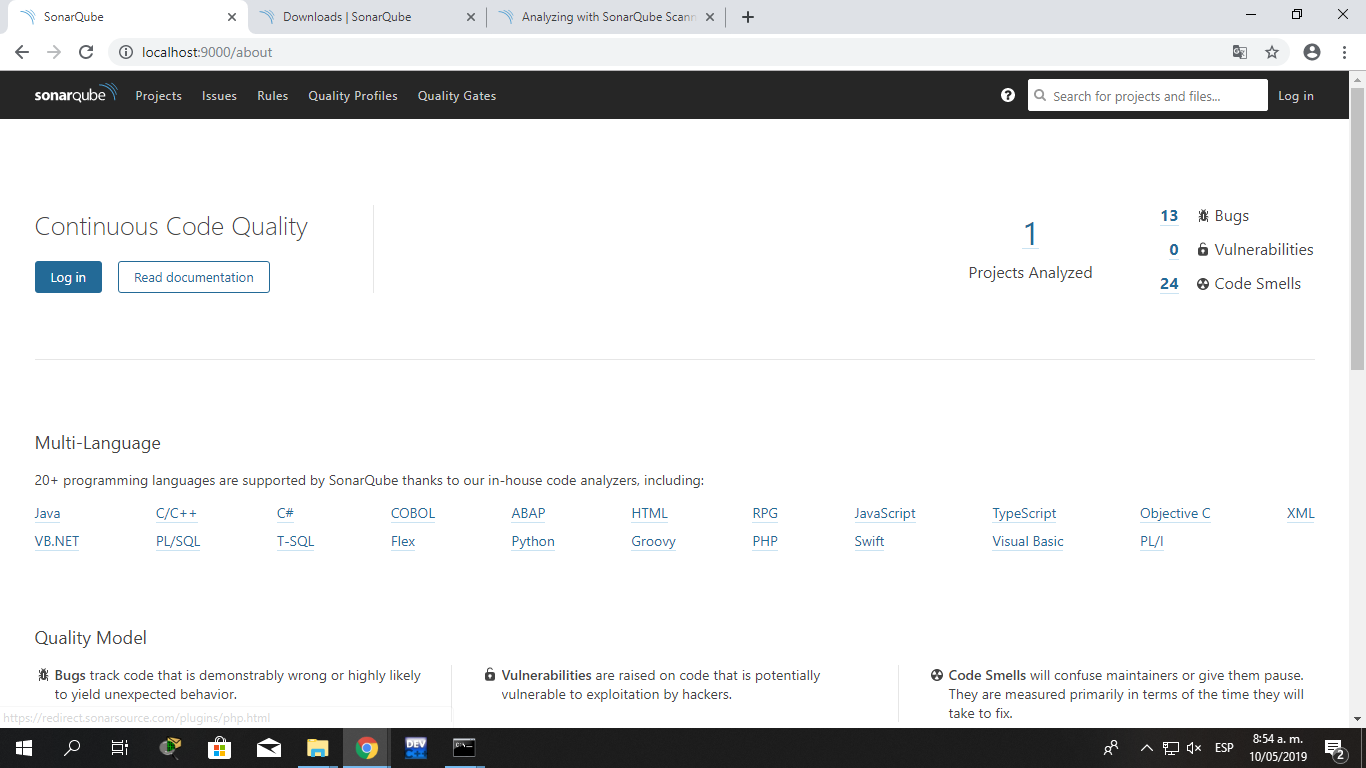


abrimos el ejemplo con cmd

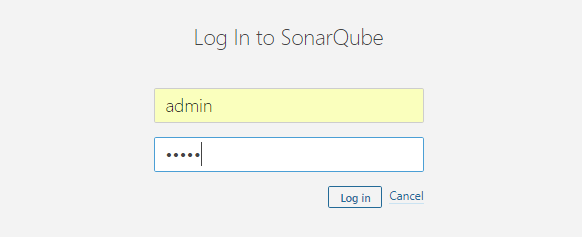




6. Una vez que esté corriendo, podremos acudir a la dirección http://localhost:9000 para ver cómo se ha inicializado correctamente la aplicación de SonarQube



7. Aquí podremos realizar login con el usuario y contraseña predefinidos: usuario: admin y contraseña: admin.



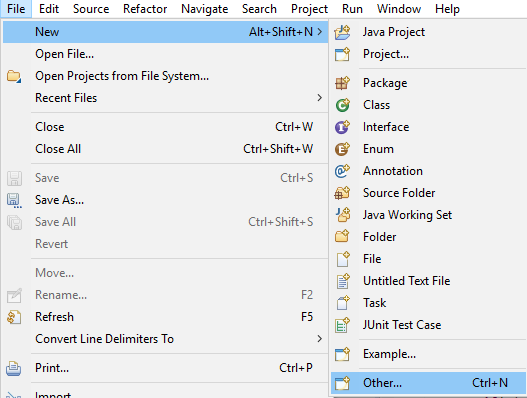
8. En este punto ya tendríamos SonarQube inicializado y vamos a ver cómo integrarlo con un proyecto Maven de ejemplo creado con Eclipse.

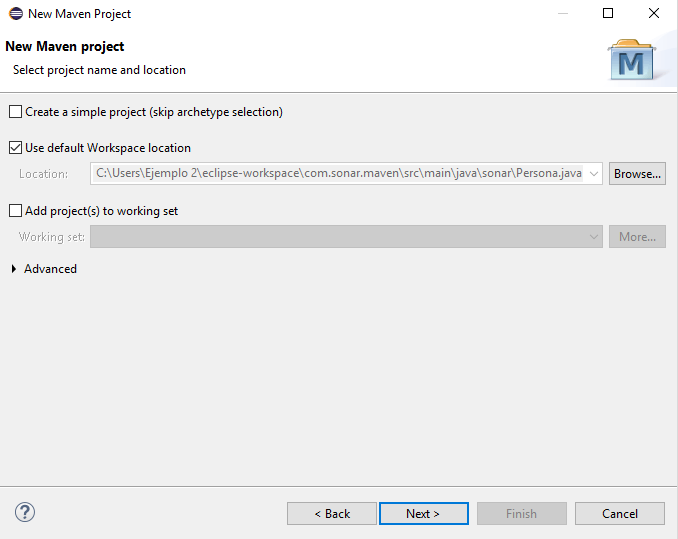
9.Instalar Eclipse

El primer paso será el de instalar la última versión de Eclipse Photon a través del [instalador de Eclipse](https://www.eclipse.org/downloads/).

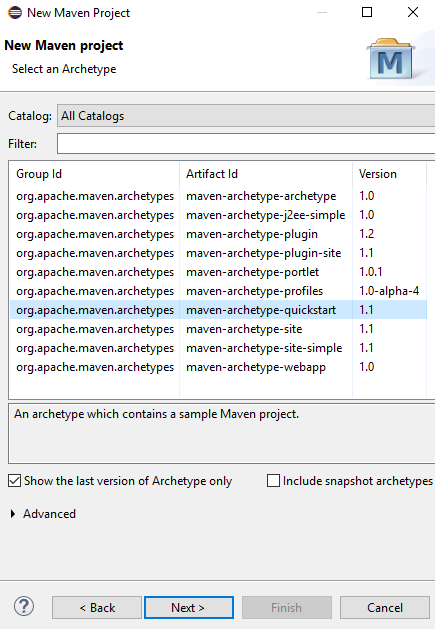


10. Una vez instalado, iniciaremos Eclipse y crearemos un proyecto Maven que configuraremos para que SonarQube pueda analizarlo y ofrecernos información acerca de la calidad de su código, debilidades, duplicidades y riesgos.

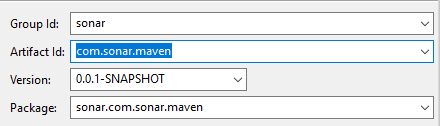




Para facilitar la creación de un proyecto con una clase predeterminada y un archivo de test unitario, elegiremos la opción de maven-archetype-quickstart:



Le daremos un ID de Grupo (Group ID) y un ID de Artefacto (Artifact ID) para identificar nuestro nuevo proyecto Maven. Por ejemplo, podríamos definir sonar como Group ID ycom.sonar.maven como Artifact ID:

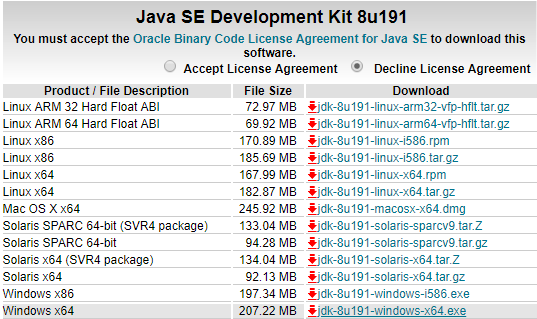


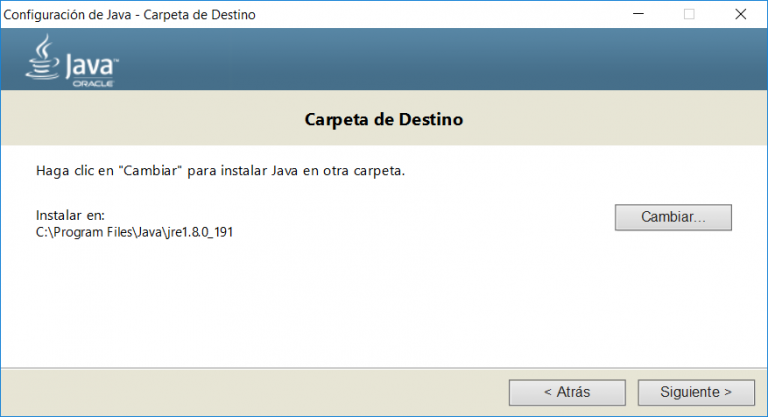
### Configurar Eclipse y un proyecto Maven

Una vez se haya creado el proyecto, debemos asegurarnos de un par de cosas:

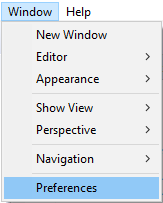
* Configurar correctamente el JDK en las preferencias de Eclipse
* Configurar correctamente el archivo Pom.xml del proyecto para poder hacer uso de Junitpara las pruebas y las configuraciones necesarias para conectar correctamente con SonarQube.

Para poder configurar correctamente el JDK, deberemos [descargarnos la última versión de JDK que se nos ofrece en la web oficial](https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).





Una vez lo descarguemos e instalemos, deberemos asegurarnos de que tenemos bien configurado nuestro Eclipse y que usamos la JDK que hemos descargado, para ello acudimos a window > preferences y desplegamos la opción de Java, para seguido seleccionar la opción de Installed JREs:





Asegúrate también de que la variable de entorno de Java es la correcta (JAVA\_HOME)

#### **Configurar correctamente el archivo pom.xml del proyecto Maven**

Ahora vamos a proceder a configurar el archivo pom.xml para utilizar algunos plugins que nos ayudarán a visualizar información en SonarQube y crear nuestros archivos de tests unitarios en el proyecto.

Deberemos asegurarnos de que el bloque Properties contiene la configuración de SonarQubenecesaria para poder hacer uso de todas sus funcionalidades y definir el lenguaje que va a analizar.

##### **properties**

1. <!-- Propiedades de Sonar y asignación de Plugin Jacoco para medir Coverage del código -->
2. <properties>
3. <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
4. <sonar.core.codeCoveragePlugin>jacoco</sonar.core.codeCoveragePlugin>
5. <sonar.jacoco.reportPath>${project.basedir}/../target/jacoco.exec</sonar.jacoco.reportPath>
6. <sonar.language>java</sonar.language>
7. </properties>

Como se puede apreciar, hemos definido dos propiedades de Sonar que hacen r

eferencia a Jacoco. La primera de todas es codeCoveragePlugin que sirve para indicar que queremos que el Coverage (Porcentaje de código cubierto con tests unitarios) va a ser jacoco. La segunda es la propiedad jacoco.reportPath donde se define el directorio donde se aloja el ejecutable de jacoco.exec que se creará cuando Maven se descargue todas las dependencias y plugins. Además, hemos incluido la propiedad languaje como Java, ya que el proyecto estará escrito en este lenguaje.**dependencies**

Ahora debemos asegurarnos de que tenemos una última versión estable de JUnit:

1. <!-- Dependencia JUnit para realizar Tests Unitarios -->
2. <dependencies>
3. <dependency>
4. <groupId>junit</groupId>
5. <artifactId>junit</artifactId>
6. <version>4.12</version>
7. <scope>test</scope>
8. </dependency>
9. </dependencies>

##### **build**

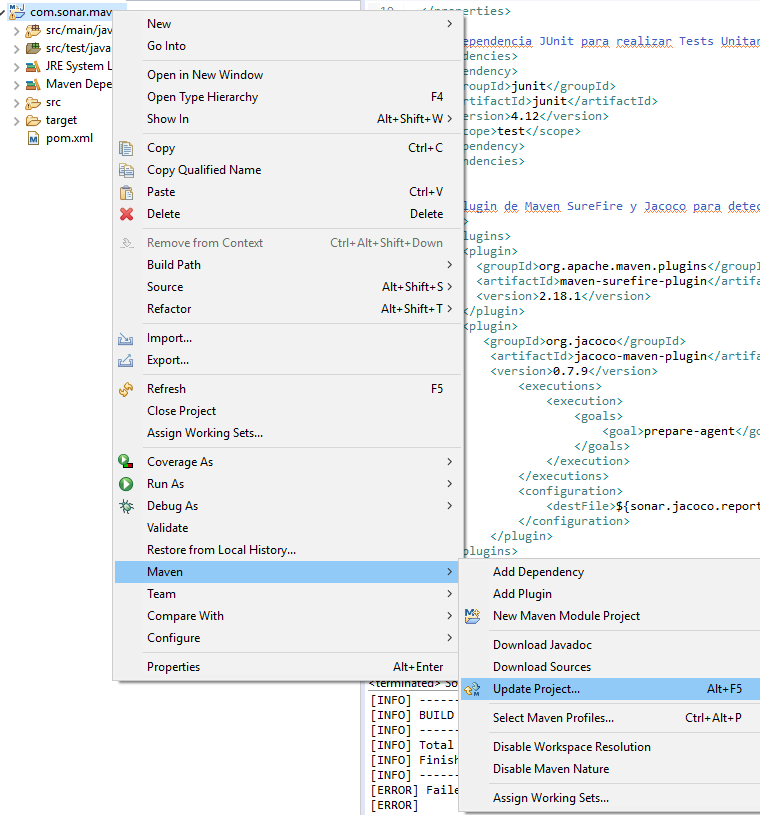
1. <!-- Plugin de Maven SureFire y Jacoco para detección de Coverage por parte de SonarQube -->
2. <build>
3. <plugins>
4. <plugin>
5. <groupId>org.apache.maven.plugins</groupId>
6. <artifactId>maven-surefire-plugin</artifactId>
7. <version>2.18.1</version>
8. </plugin>
9. <plugin>
10. <groupId>org.jacoco</groupId>
11. <artifactId>jacoco-maven-plugin</artifactId>
12. <version>0.7.9</version>
13. <executions>
14. <execution>
15. <goals>
16. <goal>prepare-agent</goal>
17. </goals>
18. </execution>
19. </executions>
20. <configuration>
21. <destFile>${sonar.jacoco.reportPath}</destFile>
22. </configuration>
23. </plugin>
24. </plugins>
25. </build>

Como se puede apreciar, definimos el plugin de jacoco y el de maven-surefire-plugin.

### Update del proyecto Maven

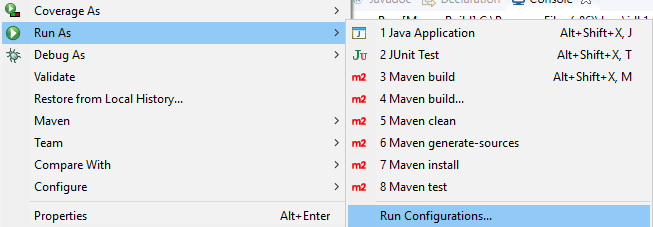
Una vez tengamos el archivo pom.xml deberemos actualizar el proyecto Maven para asegurarnos de que todas los cambios son reflejados correctamente.

Ahora hacemos click sobre el proyecto con el botón derecho.

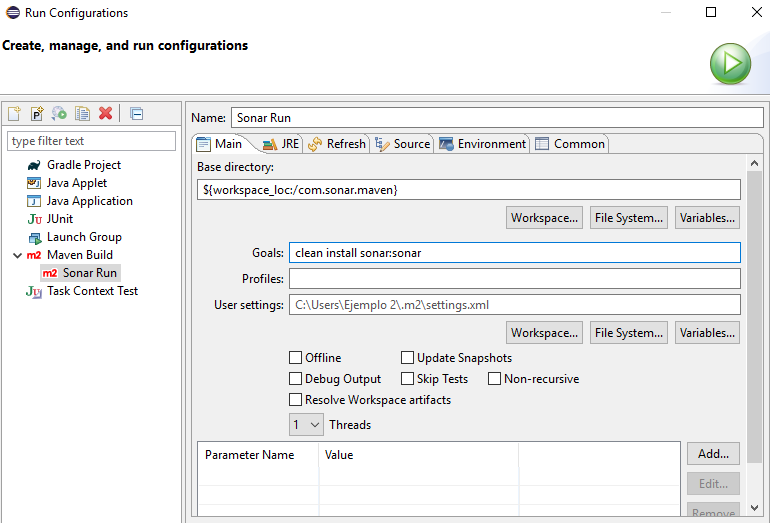


### Configuración de ejecución de Maven

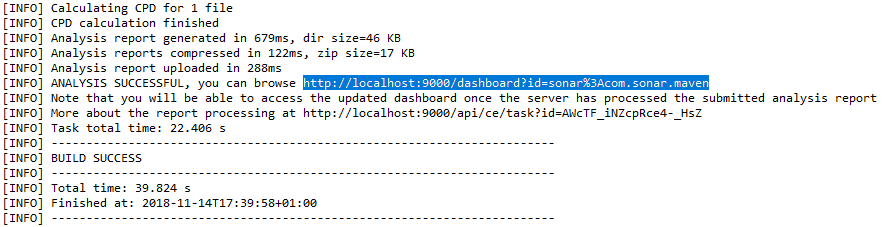
Vamos a crear una configuración personalizada de ejecución del proyecto. Ésta contendrá un clean de Maven, para a continuación, realizar un install y lanzar sonar:sonar para desplegar la aplicación en SonarQube.

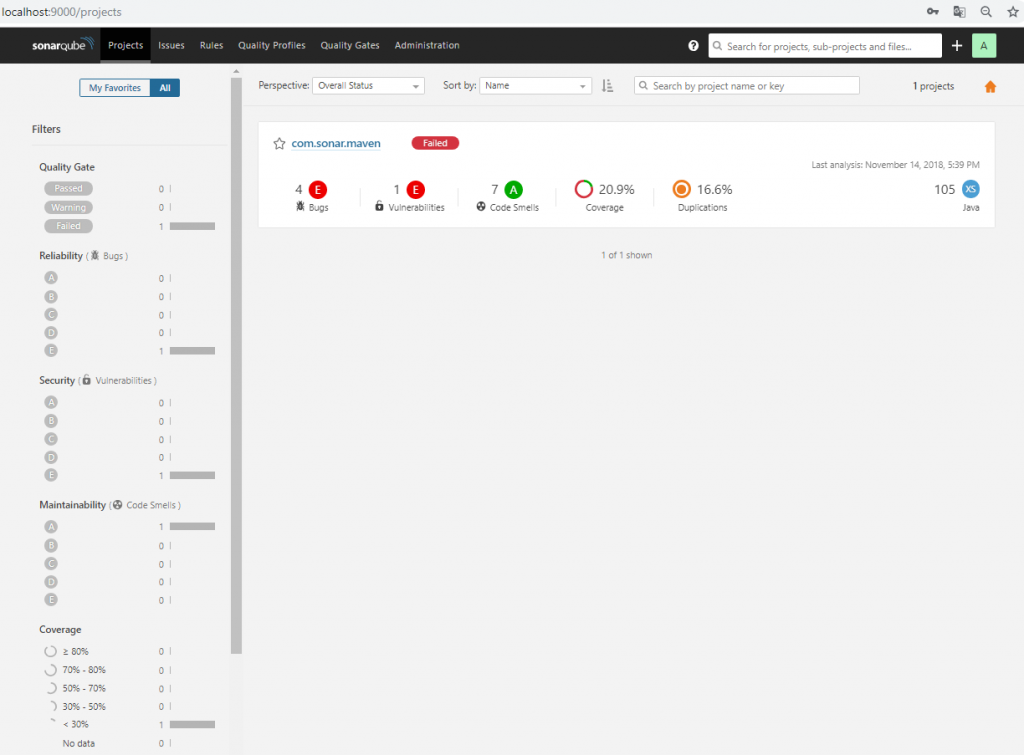


Crearemos una ejecución llamada Sonar Run definiendo el Workspace sobre el que estamos trabajando como Base Directory y cómo Goals incluiremos los comandos clean, install y sonar:sonar.



Le daremos a Apply y a Run para desplegar la aplicación en el servidor Sonar que deberá estar corriendo en localhost:9000.





Esta captura de pantalla se ha realizado sobre un proyecto ya en desarrollo. Si has ejecutado SonarQube en un proyecto creado desde cero, no tendrás vulnerabilidades, bugs o código duplicado.