

# SpotBugs

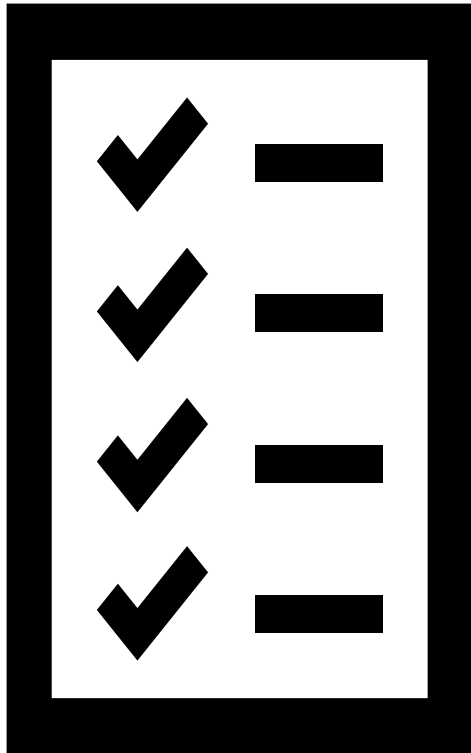


## Grupo 02:

- **Diego Fernández**
- **Óscar Gallardo**
- **Miguel García**

- **Celso Gimeno**
- **Héctor G. Iglesias**
- **Mario Martín**

# ÍNDICE



## 1. Herramienta

1. Funcionalidad
2. Bugs analizables

## 2. Opciones de uso:

*Requisitos mínimos + Instalación + Ejercicios*

1. GUI
2. Eclipse (Plug-in)
3. Maven
4. Gradle (Solo explicación teórica)

## 3. Conexión con Sonar

Puedes ver el repositorio aquí:

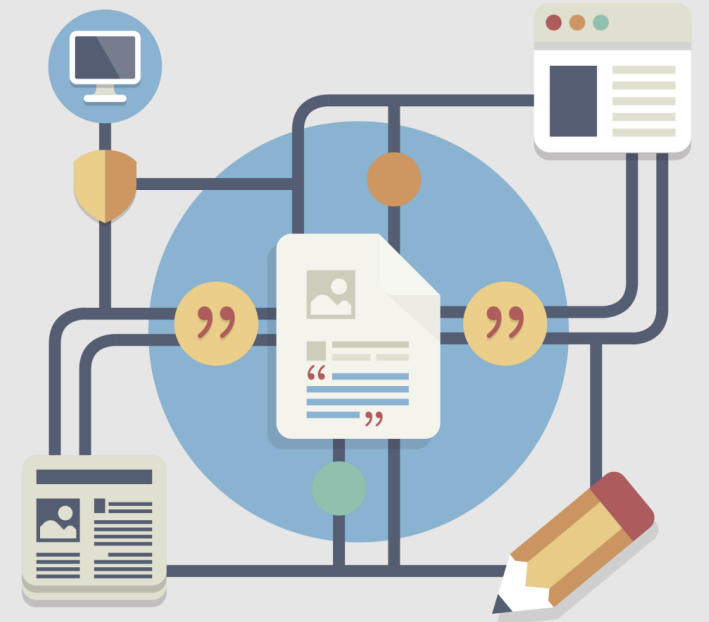
<https://github.com/mmp819/Tutorial-Spotbugs>

Puedes descargar el repositorio aquí:

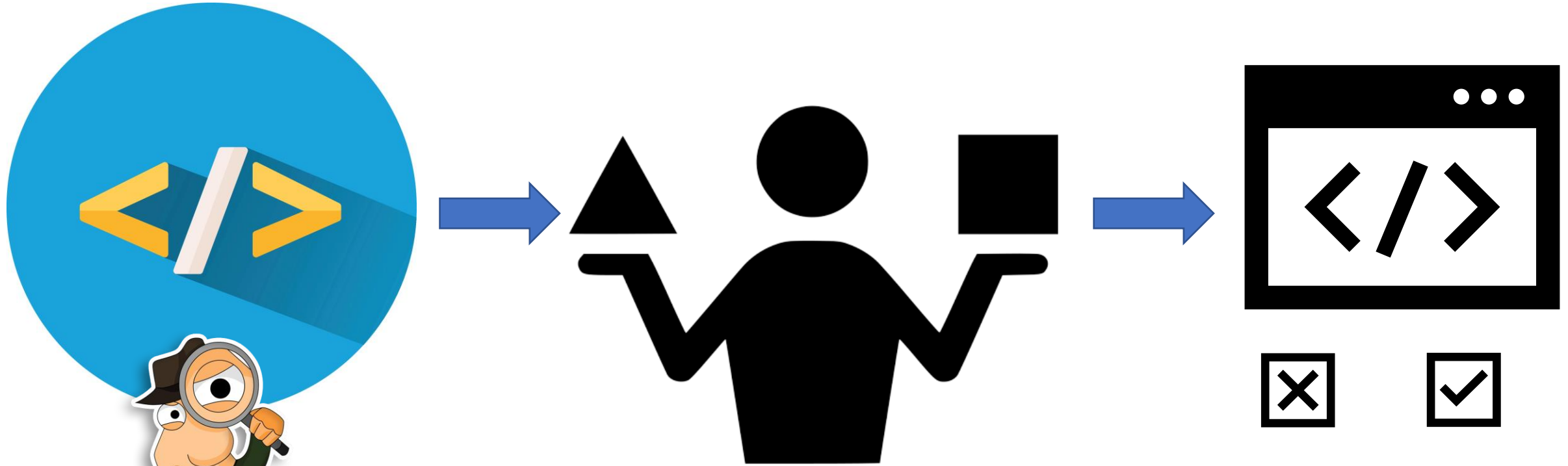
[shorturl.at/bLORY](https://shorturl.at/bLORY)

# Herramienta

# Funcionalidad



# Búsqueda de bugs en Java



**Análisis del código**

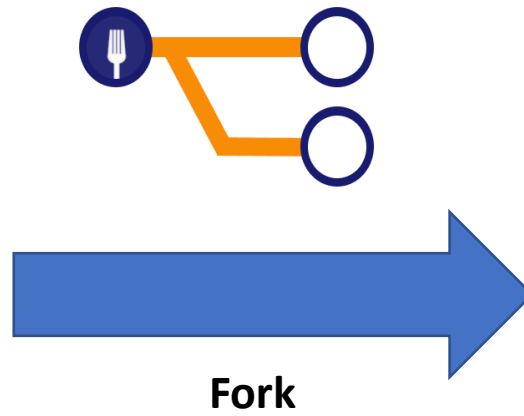
**Comparación con patrones de bugs**

**Mostrar código considerado como erróneo**

# Origen



**FindBugs**  
(Desactualizado)



**SpotBugs**

# Tipos de Bugs Analizados



# Malas prácticas

## Violaciones de prácticas esenciales de codificación.

- Retorno de valores nulos en métodos que deben devolver un valor booleano.
- Definir métodos *clone* sin implementar la interfaz *Cloneable*.
- Implementar la interfaz *Cloneable* sin definir un método *clone*.
- Uso de métodos que cierran la VM en lugar de excepciones. Ej: `System.exit`
- El método *equals* no comprueba si se ha introducido un parámetro nulo.
- Comparar *strings* con `==`.
- Redefinir método *hashCode* sin redefinir método *equals*.
- Nombrar un método comenzando por letras mayúsculas.
- Clase serializable sin especificar un identificador.
- Indicar el lanzamiento de excepciones genéricas, en vez de específicas.



# Exactitud

## Aparentes errores de código que pueden dar lugar a resultados no intencionados.

- Uso de bucles sin puntos de salida.
- Uso de bucles recursivos infinitos.
- Uso de *float* en lugar de *double* para cálculos matemáticos (falta de precisión).
- Método *equals* que siempre retorna *false* (posiblemente mal definido).
- Definir métodos como *toString*, pudiendo sobrescribir otros existentes como *toString*.
- Uso de operaciones en *arrays* donde se excede su número de elementos.
- Uso de un método *close* en un objeto sin inicializar.
- Llamada a *equals* con objetos de diferentes tipos.
- Comparación de un valor de tipo *int* con uno de tipo *long*.
- Uso de valores flotantes para bucles, con su consecuente imprecisión.

# Experimentales

## Patrones de bugs experimentales cuyo funcionamiento no ha sido probado aún

- Clase demasiado grande como para ser analizada.
- Patrón de bug desconocido.
- Posibles fallos y excepciones al cerrar una base de datos, un *stream* o cualquier otro recurso que requiera "limpiar" tras una operación.
- Posible pérdida de cambios en la configuración de un logger al usar OpenJDK.

# Experimentales

## Patrones de bugs experimentales cuyo funcionamiento no ha sido probado aún

- Clase demasiado grande como para ser analizada.
- Patrón de bug desconocido.
- Posibles fallos y excepciones al cerrar una base de datos, un *stream* o cualquier otro recurso que requiera "limpiar" tras una operación.
- Posible pérdida de cambios en la configuración de un logger al usar OpenJDK.

# Internacionalización

## Fallos en el código relativos a Locales

- Conversión de strings a mayúsculas o minúsculas sin tener en cuenta los caracteres internacionales.
- Conversiones de String a Byte y viceversa sin tener en cuenta si la plataforma de codificación es adecuada o no.

# Código vulnerable a ataques

**Código que es vulnerable a ataques de código no confiable.**

- Método invocado que solo debe invocarse dentro de un bloque doPrivileged.
- Método que puede exponer la representación interna devolviendo una referencia a un objeto mutable.
- Método estático público que puede exponer la representación interna al devolver una matriz o array.
- El campo estático y final se debe mover fuera de una interfaz y proteger el paquete.

# Uso correcto de varios hilos

## Defectos de código relacionados con subprocesos, bloqueos y volátiles.

- La secuencia de llamadas a la abstracción concurrente puede no ser atómica (ej. HashMap).
- Campo calendario estático.
- Sincronizar y comprobar nulo en el mismo campo.
- Notificación sin un bloqueo.
- Una referencia volátil a una matriz no trata los elementos de la matriz como volátiles.
- Se creó un subproceso utilizando el método de ejecución vacío predeterminado.
- Sincronización en String interno.

```
private static String LOCK = "LOCK";  
...  
synchronized(LOCK) {  
    ...  
}  
...
```

# ***Bogus random noise***

**Está destinado a ser útil como control en experimentos de minería de datos, no para encontrar errores reales en el software.**

- Advertencia falsa sobre una referencia de puntero nulo.
- Advertencia falsa sobre una referencia de campo.
- Advertencia falsa sobre una llamada de método.
- Advertencia falsa sobre una operación.

# Rendimiento

## **Código que no es necesariamente incorrecto, pero puede ser ineficiente**

- Los métodos equals y hashCode de URL están bloqueando  
(DMI\_BLOCKING\_METHODS\_ON\_URL)
- Los mapas y los conjuntos de URL pueden consumir mucho rendimiento  
(DMI\_COLLECTION\_OF\_URLS)
- El método invoca el método toString() en una cadena (DM\_STRING\_TOSTRING)
- Recolección de basura explícita; extremadamente dudoso excepto en el código de evaluación comparativa (DM\_GC)
- El método invoca un constructor de números ineficiente; use valor estático de en su lugar (DM\_NUMBER\_CTOR)
- El método invoca un constructor numérico de punto flotante ineficiente; use valor estático de en su lugar (DM\_FP\_NUMBER\_CTOR)

# Seguridad (Input)

**Un uso de entrada no confiable de una manera que podría crear una vulnerabilidad de seguridad explotable de forma remota.**

- Cookie HTTP formada a partir de una entrada que no es de confianza (HRS\_REQUEST\_PARAMETER\_TO\_COOKIE)
- El servlet reflejó una vulnerabilidad de secuencias de comandos entre sitios en la página de error (XSS\_REQUEST\_PARAMETER\_TO\_SEND\_ERROR)
- Recorrido de ruta absoluta en *servlet* (PT\_ABSOLUTE\_PATH\_TRAVERSAL)
- Vulnerabilidad de secuencias de comandos entre sitios reflejada en el *servlet* (XSS\_REQUEST\_PARAMETER\_TO\_SERVLET\_WRITER)



# Código anómalo/confuso

**Código que es confuso, anómalo o escrito de una manera que conduce a errores.**

- Código inútil.  
(UC\_USELESS\_OBJECT)  
(UC\_USELESS\_VOID\_METHOD)

- Estructura try catch sin que se pueda lanzar una excepción.  
(REC\_CATCH\_EXCEPTION)

- Switch sin default.  
(SF\_SWITCH\_NO\_DEFAULT)

- Mismo código en dos ramas distintas.  
(DB\_DUPLICATE\_SWITCH\_CLAUSES)

# Opciones de uso



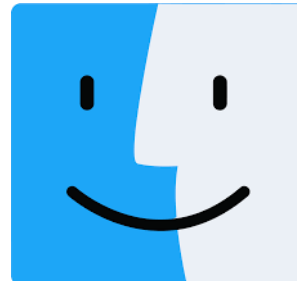
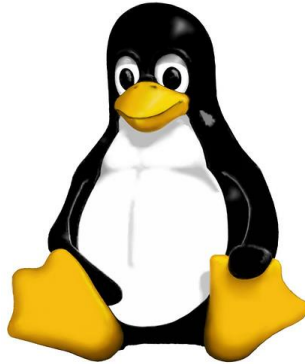
# Requisitos mínimos

## JRE

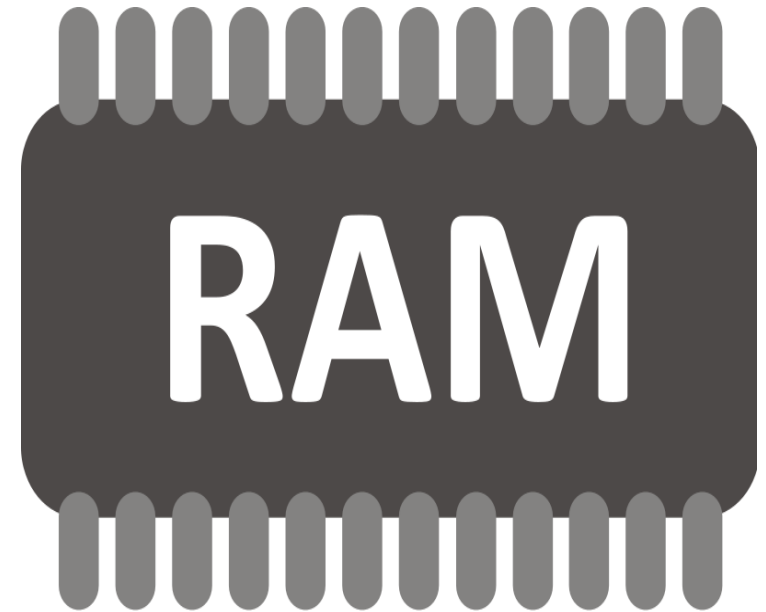


Versión 1.8 o superior

## Sistema operativo



## RAM



> 512 MB

Fuente de Imágenes:

Java: [https://www.flaticon.es/icono-gratis/java\\_226777](https://www.flaticon.es/icono-gratis/java_226777)

Windows: [https://es.m.wikipedia.org/wiki/Archivo:Windows\\_logo\\_-\\_2021.svg](https://es.m.wikipedia.org/wiki/Archivo:Windows_logo_-_2021.svg)

Linux: [https://es.m.wikipedia.org/wiki/Archivo:Linux\\_logo.jpg](https://es.m.wikipedia.org/wiki/Archivo:Linux_logo.jpg)

MAC OS: <https://logos-download.com/50541-apple-mac-os-logo-download.html>

RAM: [https://esferas.org/mt/msqlu/assets\\_c/2013/09/pgb-ram-11.html](https://esferas.org/mt/msqlu/assets_c/2013/09/pgb-ram-11.html)

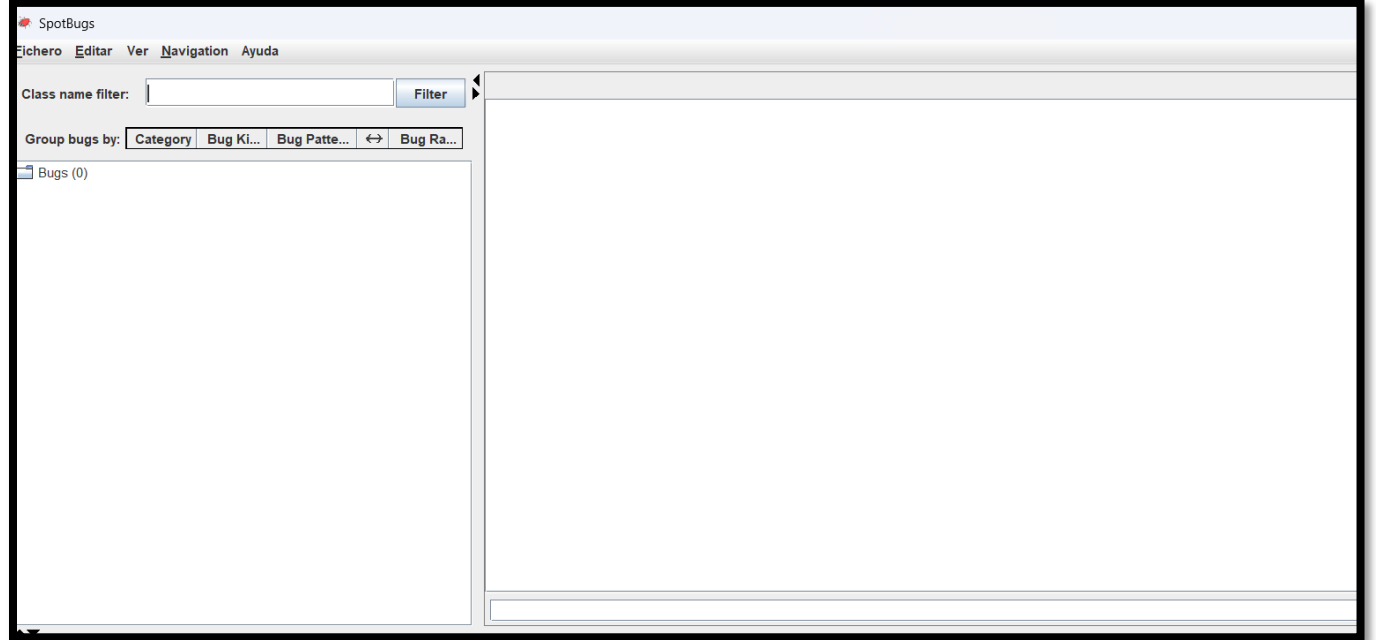
# GUI Spotbugs

1º Descargar versión 4.7.3 de Spotbugs: <https://github.com/spotbugs/spotbugs/releases/download/4.7.3/spotbugs-4.7.3.zip>

2º Descomprimir fichero .zip

3º Acceder a carpeta *lib*.

4º Ejecutar *spotbugs.jar*.



5º Descargar ejercicio (GUI), consistente en un proyecto con errores y descomprimirlo.

6º Importar el ejercicio en Eclipse (Projects from Folder or Archive).

Puedes encontrar el ejercicio para el uso de la GUI en el repositorio:  
[shorturl.at/bLORY](https://shorturl.at/bLORY)

# GUI Spotbugs

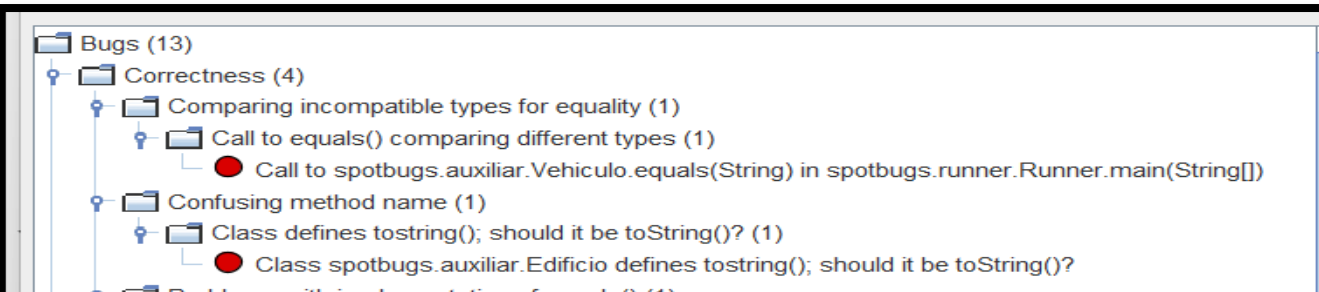
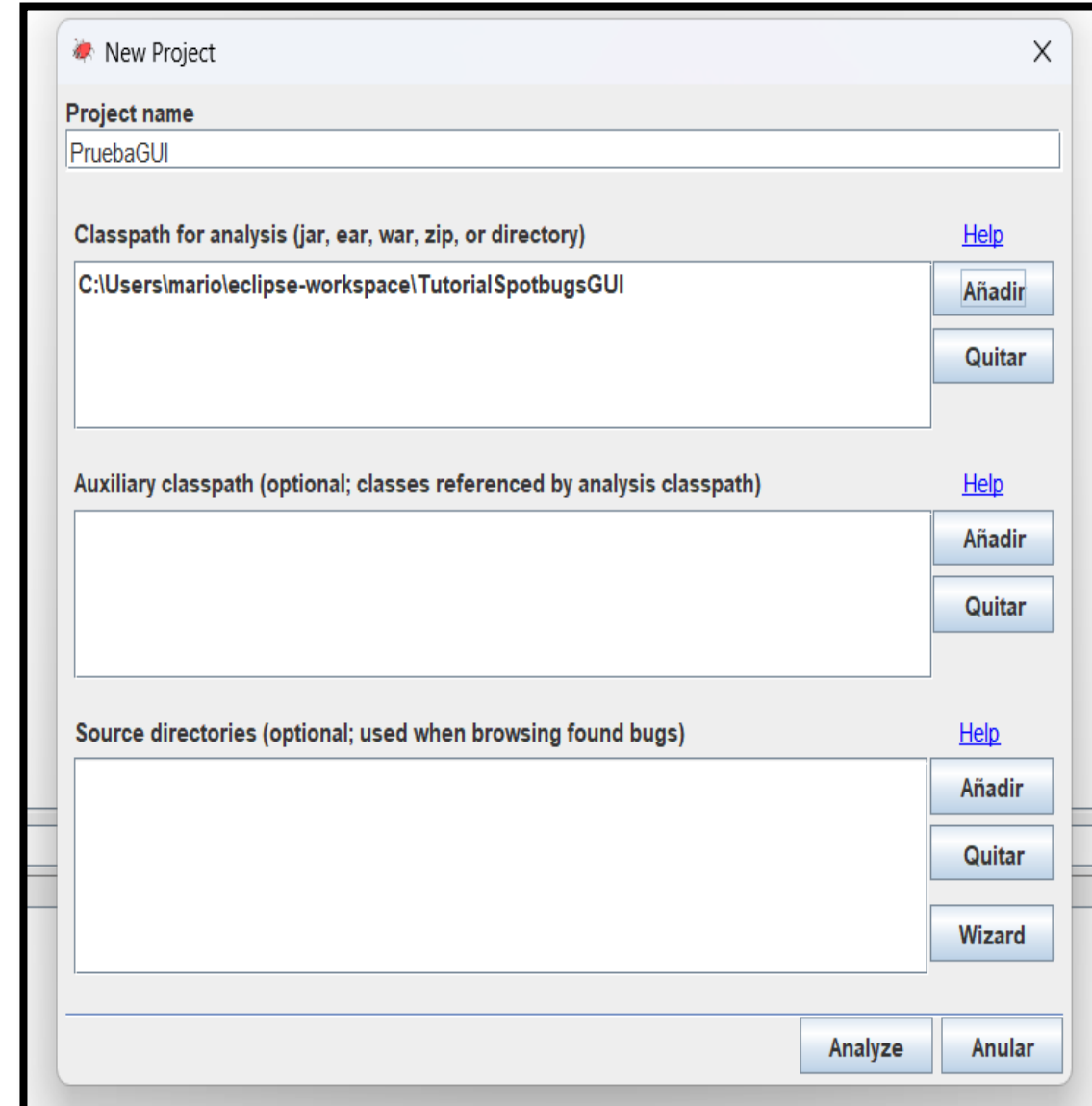
7º Clickar en Fichero > Nuevo Proyecto.

8º Otorgar un nombre al proyecto.

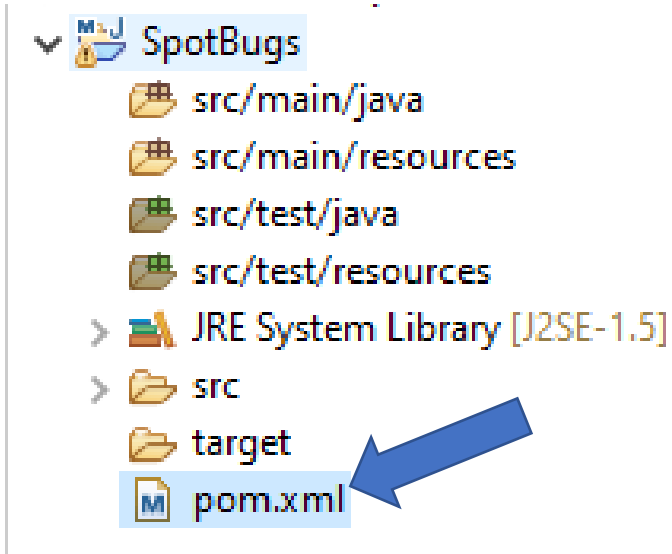
9º En la sección *Classpath*, indicar la ruta del proyecto.

10º Realizar análisis y observar los bugs descritos.

11º Resolver los bugs y volver a ejecutar el análisis.  
Puedes hacer esto con *Fichero > Redo Analysis*.



# Maven



```
<build>
  <plugins>
    <plugin>
      <groupId>com.github.spotbugs</groupId>
      <artifactId>spotbugs-maven-plugin</artifactId>
      <version>4.7.3.0</version>
      <dependencies>
        <dependency>
          <groupId>com.github.spotbugs</groupId>
          <artifactId>spotbugs</artifactId>
          <version>4.7.3</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
```

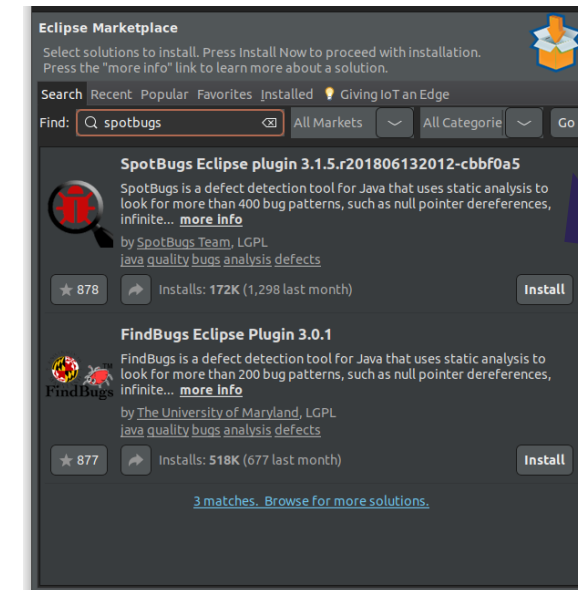
**mvn spotbugs:check**

# Eclipse (Plug-in)

**Versión mínima: Neon 4.6  
(30/05/2016)**



**Disponible en  
Marketplace**



Enlace al código del taller de SpotBugs: [shorturl.at/bLORY](https://shorturl.at/bLORY)

# Gradle

```
// Top-level build file where you can add configuration options common to all sub-projects/modules
buildscript {
    repositories {
        google()
        mavenCentral()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.2.2'
        classpath "org.jacoco:org.jacoco.core:0.8.7"
        // If you're using gradle 6.x, add this to use SpotBugs app version 4.0.2
        classpath "gradle.plugin.com.github.spotbugs.snom:spotbugs-gradle-plugin:4.3.0"
        // If you're using gradle 4.x or 5.x, add this to use SpotBugs app version 3.1.2
        classpath "com.github.spotbugs:spotbugs-gradle-plugin:2.0.1"
    }
}

plugins {
    id 'com.android.application' version '7.2.1' apply false
    id 'com.android.library' version '7.2.1' apply false
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

```
plugins {
    id 'com.android.application'
    id "org.sonarqube" version "3.0"
    id "com.github.spotbugs"
}

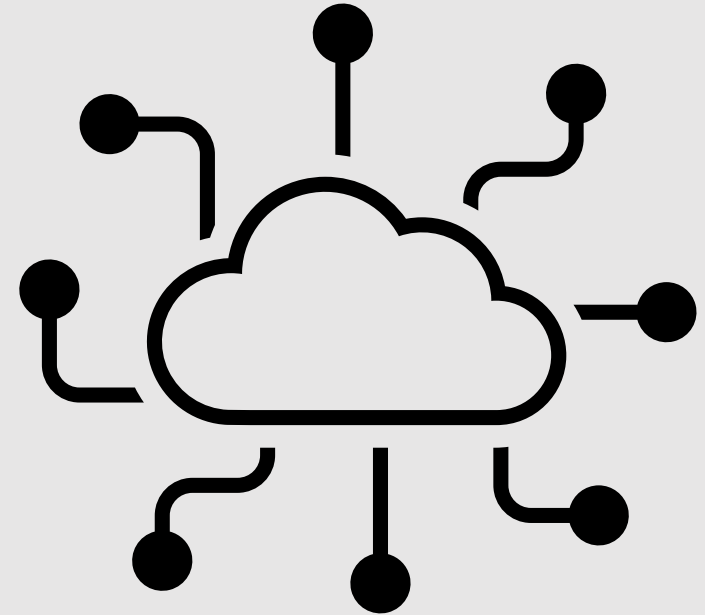
apply plugin: "jacoco"
jacoco {toolVersion = '0.8.7'}
tasks.withType(Test) {...}
task jacocoTestReport(type: JacocoReport, dependsOn: ['testDebugUnitTest']) {...}

spotbugs {
    ignoreFailures = true
    reportsDir = file("$project.buildDir/SpotBugsReports")
    effort = "max"
    reportLevel = "high"
}

// Note: gradle 4/5 should use "com.github.spotbugs.SpotBugsTask"
tasks.withType(com.github.spotbugs.snom.SpotBugsTask) {
    dependsOn 'assembleDebug'
    classes = files("$project.buildDir/intermediates/javac") // Important to use this path
    reports {
        // Enable HTML report only
        html.enabled = true
        xml.enabled = false
    }
}
```



# Conexión a Sonar

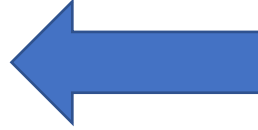


# Crear proyecto Maven

```
<dependencies>
  <dependency>
    <groupId>com.github.spotbugs</groupId>
    <artifactId>sonar-findbugs-plugin</artifactId>
    <version>4.2.2</version>
  </dependency>

  <dependency>
    <groupId>org.sonarsource.sonarqube</groupId>
    <artifactId>sonar-plugin-api</artifactId>
    <version>8.9.10.61524</version>
    <scope>provided</scope>
  </dependency>
</dependencies>

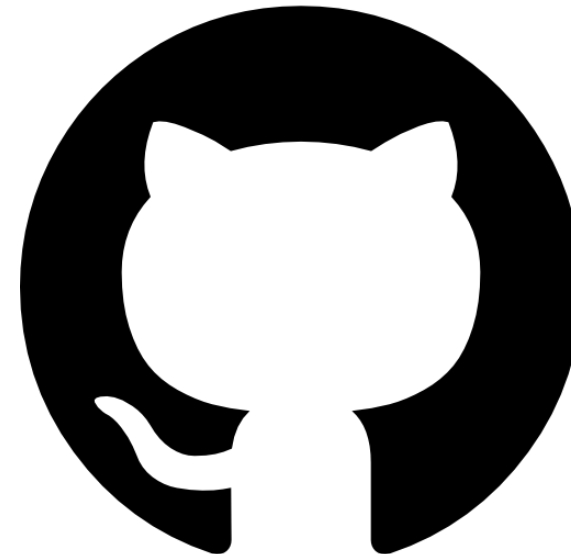
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.sonarsource.scanner.maven</groupId>
        <artifactId>sonar-maven-plugin</artifactId>
        <version>3.8.0.2131</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
  <sonar.host.url>http://localhost:9000</sonar.host.url>
  <sonar.login>admin</sonar.login>
  <sonar.password>1234</sonar.password>
</properties>
```



Dependencias y configuración a añadir  
en el fichero POM.



También disponible para  
copiarlo desde fichero  
.txt.



*Haz click aquí*

# Lanzar Sonar

1º Descargar versión 8.9.10 LTS: <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.10.61524.zip>

2º Descomprimir fichero .zip

3º Ejecutar Sonar, dentro de la carpeta bin:

- Windows: Ejecutar archivo *StartSonar.bat*.
- Linux: Ejecutar archivo *sonar.sh*.
- Mac: Ejecutar archivo *sonar.sh*. (Puede ser necesario cambiar ajustes de seguridad).

4º Esperar a que se levante el servidor. (Puede haber errores en versiones de Java distintas Java-11).

5º Acceso al servidor levantado en: <http://localhost:9000>.

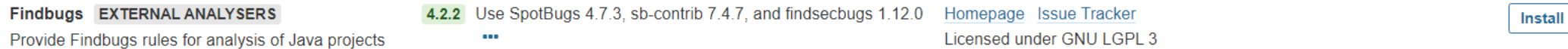
6º Cambiar la contraseña por una de vuestro gusto para el primer acceso.

→ *Se recomienda establecer como contraseña: **1234**, de forma que no sea necesario modificar el POM entregado.*

# Configurar plug-in en Sonar

1º Acceder a *Administration > Marketplace*.

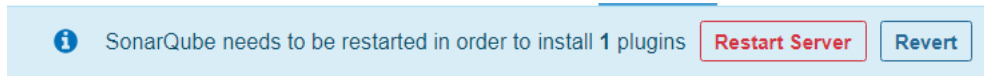
2º Buscar plug-in: Findbugs.




The screenshot shows the 'Findbugs' plugin page in the Sonar Marketplace. It is categorized under 'EXTERNAL ANALYSERS'. The version '4.2.2' is highlighted in green. The description states: 'Use SpotBugs 4.7.3, sb-contrib 7.4.7, and findsecbugs 1.12.0'. There are links for 'Homepage' and 'Issue Tracker', and a note 'Licensed under GNU LGPL 3'. An 'Install' button is visible on the right.

Findbugs **EXTERNAL ANALYSERS** **4.2.2** Use SpotBugs 4.7.3, sb-contrib 7.4.7, and findsecbugs 1.12.0 [Homepage](#) [Issue Tracker](#) [Install](#)  
Provide Findbugs rules for analysis of Java projects ... Licensed under GNU LGPL 3

3º Instalar el plug-in. Una vez aparezca como pendiente de instalación, reiniciar con el botón que aparecerá arriba (*restart server*) y volver a lanzar el servidor ejecutando el script.



The screenshot shows a light blue notification bar with an information icon on the left. The text reads: 'SonarQube needs to be restarted in order to install 1 plugins'. To the right of the text are two buttons: 'Restart Server' (highlighted with a red border) and 'Revert'.

 SonarQube needs to be restarted in order to install 1 plugins [Restart Server](#) [Revert](#)

4º Acceder a *Quality Profiles* y filtrar perfiles por *Java*.

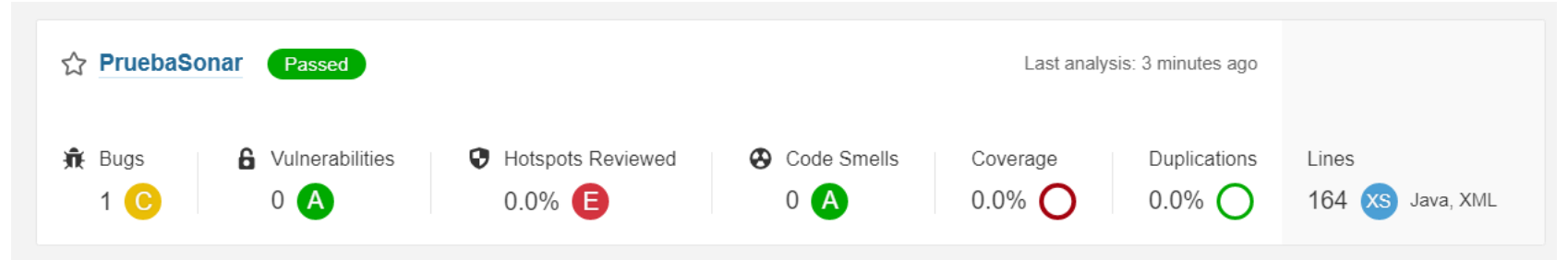
5º Clickar en *Findbugs*. Pulsar el engranaje en la esquina superior derecha y marcar como predeterminado.

6º A partir de ahora, el análisis se realizará en base a sus regla, en lugar de las de *Sonar*.

# Ejecutar Análisis

1º Lanzar: *mvn clean install*.

2º Lanzar: *mvn sonar:sonar*.



3º Refrescar ventana del navegador en la que se visualiza el servidor.

4º El análisis se ha realizado teniendo en cuenta las reglas de *Findbugs*.

Correctness - Call to equals() comparing different types

findbugs:EC\_UNRELATED\_TYPES

Bug Major correctness Available Since Dec 08, 2022 FindBugs (Java) Constant/issue: 1h

This method calls equals(Object) on two references of different class types and analysis suggests they will be to objects of different classes at runtime. Further, examination of the equals methods that would be invoked suggest that either this call will always return false, or else the equals method is not be symmetric (which is a property required by the contract for equals in class Object).

# ¿Dudas?

**Quedamos a vuestra disposición.**