Informe de Práctica 3

Herramientas de Calidad del Producto Software y Documentación

SonarQube, Maven y Doxygen

Adnan Hawari Capa
27/10/2023

Laboratorio y Desarrollo de Herramientas

1. Introducción

SonarQube es una herramienta de código abierto diseñada para la inspección continua de la calidad del código, permitiendo la detección automática de bugs, code smells y vulnerabilidades de seguridad en el código fuente. Al integrarse con sistemas de integración y despliegue continuo (CI/CD), ofrece evaluaciones basadas en reglas para múltiples lenguajes de programación, proporcionando métricas y estimaciones como la "deuda técnica" para ayudar a los equipos a mejorar la calidad y seguridad de sus aplicaciones.

Maven es una herramienta de gestión y comprensión de proyectos de software. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar la construcción, la notificación y la documentación de un proyecto desde una pieza central de información. Además de la gestión de las dependencias y la automatización del proceso de compilación, facilita la integración con otras herramientas, estableciendo estándares uniformes de construcción. Su amplio ecosistema de plugins lo hace adaptable a diversas necesidades, simplificando y estandarizando tareas complejas en el ciclo de vida del desarrollo de software.

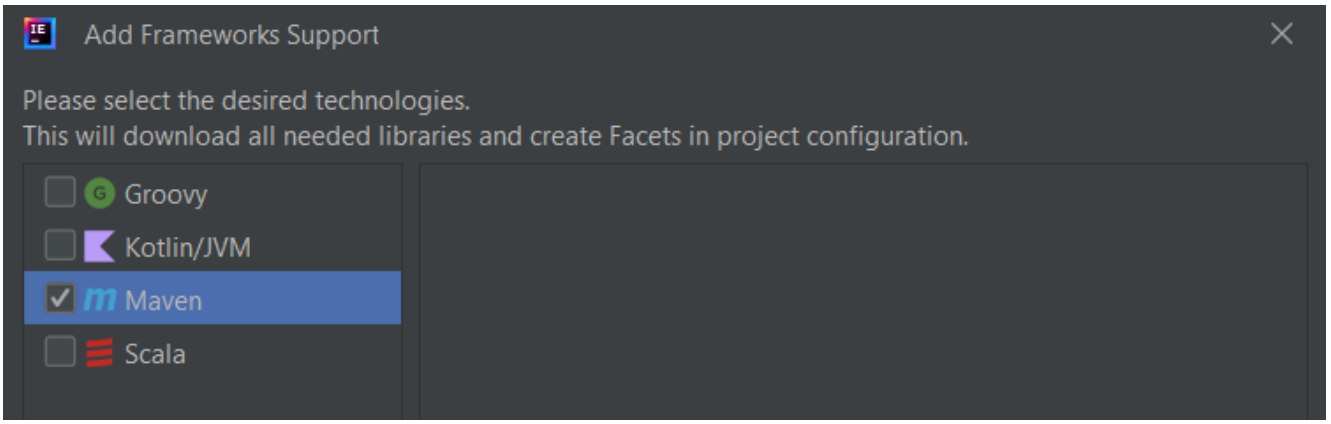
Doxygen es una herramienta de documentación que genera documentación de referencia a partir del código fuente de programas escritos en diversos lenguajes de programación. A través de comentarios especiales insertados dentro del código, Doxygen extrae información y crea documentación en varios formatos, como HTML y LaTeX. Además de la documentación de la API, Doxygen también visualiza las relaciones entre los diferentes elementos del código a través de gráficos y diagramas, lo que facilita la comprensión de la estructura y relaciones dentro de un proyecto. Es ampliamente utilizado en proyectos de software de código abierto y cerrado para garantizar que el código esté adecuadamente documentado y sea accesible para desarrolladores y usuarios.

Esta práctica se centra en la familiarización y aplicación práctica de la herramienta Maven, con un enfoque adicional en la integración de SonarQube y Doxygen para el análisis y documentación de proyectos. En la primera sección, exploraremos el fundamento y la instalación de Maven, sumergiéndonos en sus comandos esenciales tanto desde la línea de comandos como desde un IDE. A continuación, tomaremos como referencia un proyecto específico, RTVRPTW: “Recomendador Turístico con Ventanas de Tiempo”, para construir y gestionar utilizando Maven. El proceso continuará con la exploración profunda del archivo **pom.xml** y la configuración de plugins en Maven. Posteriormente, emplearemos Maven junto con SonarQube para analizar aspectos del proyecto, enfocándonos especialmente en cuestiones de seguridad. Para concluir, abordaremos la documentación del proyecto con la ayuda de Doxygen, destacando elementos clave en el código y aprovechando las funcionalidades que Doxygen ofrece para facilitar esta tarea. El objetivo global es proporcionar una experiencia integral en la gestión, análisis y documentación de proyectos de software utilizando herramientas modernas y eficientes.

2. Configuración Inicial del Proyecto

Una vez Instalado y configurado Maven, procedemos a descargar el proyecto a nuestro equipo. En nuestro caso utilizamos el IDE Intellij, y el proyecto RTVRPTW es un proyecto de tipo .project (Eclipse), por lo que debemos realizar ciertos cambios. Primero importamos el proyecto con la opción "***proyecto nuevo de fuentes existentes***" seleccionamos el tipo de proyecto de origen, en este caso Eclipse, y procedemos al cambio.

Después de esto tendremos un proyecto en formato Intellij, pero aún no será un proyecto de Maven, ni tendrá su estructura (directorios, y pom.xml). Por lo que procedemos a añadir soporte de Framework Maven.



Ya con una estructura de tipo Maven y con el archivo pom.xml disponible, podemos proceder a configurar nuestro proyecto. Utilizamos los comandos *validate*, *compile*, y *package* para compilar y empaquetar el proyecto. Seguidamente pasamos a instalar los Plugin.

3. Plugins Instalados en Maven

3.1 Checkstyle

Checkstyle es un plugin utilizado en Maven para evaluar y asegurar que el código Java cumpla con una codificación y formato estandarizado. Se basa en configuraciones que definen un conjunto de reglas y patrones de codificación a los que debe adherirse el código. Cuando se ejecuta, Checkstyle produce informes que señalan las violaciones de estas reglas en el código fuente.

A traves del enlace : https://maven.apache.org/plugins/maven-checkstyle-plugin/usage.html obtenemos el código para añadir a nuestro pom.xml.

```xml
<project>
  ...
   <reporting>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-checkstyle-plugin</artifactId>
          <version>3.3.1</version>
          <reportSets>
            <reportSet>
              <reports>
                <report>checkstyle</report>
              </reports>
            </reportSet>
          </reportSets>
        </plugin>
      </plugins>
    </reporting>
  ...
</project>
```

Una vez añadido a nuestro pom.xml, ejecutamos el comando **mvn site** para generar el report junto con los demás reports del proyecto o **mvn checkstyle : checkstyle** para generarlo de manera independiente.

```
C:\Users\Adn\IdeaProjects\ExpositoTOP>mvn checkstyle:checkstyle
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< org.example:ExpositoTOP >-----------------------
[INFO] Building ExpositoTOP 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- checkstyle:3.3.1:checkstyle (default-cli) @ ExpositoTOP ---
[INFO] Rendering content with org.apache.maven.skins:maven-default-skin:jar:1.3 skin.
[INFO] There are 922 errors reported by Checkstyle 9.3 with sun_checks.xml ruleset.
[WARNING] Unable to locate Source XRef to link to - DISABLED
[WARNING] Unable to locate Test Source XRef to link to - DISABLED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.931 s
[INFO] Finished at: 2023-10-26T21:10:59+01:00
[INFO] ------------------------------------------------------------------------
```

3.2 Apache Maven PMD Plugin

El Apache Maven PMD Plugin es una herramienta que permite integrar el análisis de código fuente de PMD con el proceso de construcción de Maven. PMD es un popular analizador estático de código para Java (y otros lenguajes) que identifica y reporta posibles problemas, malos patrones de codificación y errores potenciales en el código fuente. A su vez tiene compatibilidad con CPD (Copy/Paste Detector) que es una herramienta que detecta bloques de código duplicados y permite ejecutarlo.

A traves del enlace : https://maven.apache.org/plugins/maven-pmd-plugin/usage.html
obtenemos el código para añadir a nuestro pom.xml.

```xml
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>3.21.0</version>
      </plugin>
    </plugins>
  </reporting>
  ...
</project>
```

Una vez añadido a nuestro pom.xml, ejecutamos el comando **mvn site** para generar el report junto con los demás report del proyecto **mvn pmd:pmd** para hacerlo de manera independiente.

```
C:\Users\Adn\IdeaProjects\ExpositoTOP>mvn pmd:pmd
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< org.example:ExpositoTOP >-----------------------
[INFO] Building ExpositoTOP 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- pmd:3.21.0:pmd (default-cli) @ ExpositoTOP ---
[WARNING] Unable to locate Source XRef to link to - DISABLED
[INFO] PMD version: 6.55.0
[INFO] Rendering content with org.apache.maven.skins:maven-default-skin:jar:1.3 skin.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.861 s
[INFO] Finished at: 2023-10-26T21:52:22+01:00
[INFO] ------------------------------------------------------------------------
```

3.3 OWASP. Dependency Check

Este Plugin identifica componentes con vulnerabilidades conocidas en proyectos, usando bases de datos como la National Vulnerability Database (NVD). Genera informes detallados de dependencias vulnerables, sugiere versiones seguras, se integra con el proceso de construcción de Maven y permite la supresión de falsos positivos. Es una herramienta esencial para mejorar la seguridad de las aplicaciones, asegurándose de que las bibliotecas usadas no contengan vulnerabilidades publicadas.

En : https://jeremylong.github.io/DependencyCheck/dependency-check-maven/index.html obtenemos el código para añadir a nuestro pom.xml.

```
<project>
    ...
    <reporting>
        ...
        <plugins>
            ...
            <plugin>
                <groupId>org.owasp</groupId>
                <artifactId>dependency-check-maven</artifactId>
                <version>8.4.2</version>
                <reportSets>
                    <reportSet>
                        <reports>
                            <report>aggregate</report>
                        </reports>
                    </reportSet>
                </reportSets>
            </plugin>
            ...
        </plugins>
        ...
    </reporting>
    ...
</project>
```

Una vez añadido a nuestro pom.xml, ejecutamos el comando **mvn site** para generar el report junto con los demás report del proyecto.

3. Análisis con Maven

3.1 Resultados Checkstyle

Checkstyle Results

The following document contains the results of Checkstyle 9.3 with sun_checks.xml ruleset.

Summary

Files	Info	Warnings	Errors
11	0	0	922

Files

File	I	W	E
main/java/es/ull/esit/utilities/BellmanFord.java	0	0	12
main/java/es/ull/esit/utilities/ExpositoUtilities.java	0	0	73
main/java/es/ull/esit/utilities/PowerSet.java	0	0	9
main/java/es/ull/esit/utils/Pair.java	0	0	19
main/java/top/TOPTW.java	0	0	165
main/java/top/TOPTWEvaluator.java	0	0	8
main/java/top/TOPTWGRASP.java	0	0	336
main/java/top/TOPTWReader.java	0	0	21
main/java/top/TOPTWRoute.java	0	0	35
main/java/top/TOPTWSolution.java	0	0	184
main/java/top/mainTOPTW.java	0	0	60

Tras analizar los 11 archivos se encontraron 922 errores en total, sin advertencias ni informaciones. Los archivos, entre los que se incluyen "BellmanFord.java" y "TOPTWGRASP.java", presentan violaciones en múltiples categorías, como diseño, nomenclatura y espacios en blanco. Las infracciones más comunes se relacionan con **"WhitespaceAround"** (143 errores), **"MissingJavadocMethod"** (102 errores) y **"FinalParameters"** (122 errores). Además, se detectaron problemas de longitud de línea y 80 instancias de líneas con espacios finales innecesarios.

Rules

Category	Rule	Violations	Severity
blocks	LeftCurly	26	Error
	NeedBraces	3	Error
	RightCurly	2	Error
coding	HiddenField	23	Error
	MagicNumber	60	Error
	MultipleVariableDeclarations	8	Error
	SimplifyBooleanExpression	1	Error
design	DesignForExtension	80	Error
	HideUtilityClassConstructor	3	Error
	VisibilityModifier	7	Error
javadoc	InvalidJavadocPosition	1	Error
	JavadocMethod	2	Error
	JavadocPackage	3	Error
	JavadocVariable	35	Error
	MissingJavadocMethod	102	Error
misc	ArrayTypeStyle	6	Error
	FinalParameters	122	Error
	NewlineAtEndOfFile	1	Error
naming	LocalFinalVariableName	2	Error
	LocalVariableName	1	Error
	MethodName	1	Error
	StaticVariableName	2	Error
	TypeName	1	Error
regexp	RegexpSingleline • format: "\s+$" • maximum: "0" • message: "Line has trailing spaces." • minimum: "0"	80	Error
sizes	LineLength • fileExtensions: "java"	89	Error
whitespace	GenericWhitespace	35	Error
	WhitespaceAfter	83	Error
	WhitespaceAround	143	Error

3.2 Resultados pmd

El informe generado por PMD muestra violaciones de código en varios archivos.
Las violaciones se categorizaron en prioridades, siendo las de prioridad 3 las más críticas, incluyendo temas como bloques catch vacíos y condicionales anidados, mientras que las de prioridad 4 se centraron principalmente en el uso innecesario de paréntesis.

PMD Results

The following document contains the results of PMD 6.55.0.

Violations By Priority

Priority 3

main/java/es/ull/esit/utilities/ExpositoUtilities.java

Rule	Violation	Line
UnusedPrivateMethod	Avoid unused private methods such as 'getFirstAppearance(int,int)'.	23
CollapsibleIfStatements	These nested if statements could be combined	90-93
EmptyCatchBlock	Avoid empty catch blocks	217-218
EmptyCatchBlock	Avoid empty catch blocks	226-227

main/java/es/ull/esit/utilities/PowerSet.java

Rule	Violation	Line
ClassCastExceptionWithToArray	This usage of the Collection.toArray() method will throw a ClassCastException.	16

main/java/top/TOPTWGRASP.java

Rule	Violation	Line
UnusedLocalVariable	Avoid unused local variables such as 'newDepot'.	176
CollapsibleIfStatements	These nested if statements could be combined	287-289

Priority 4

main/java/top/TOPTWGRASP.java

Rule	Violation	Line
UselessParentheses	Useless parentheses.	221
UselessParentheses	Useless parentheses.	282

3.3 Resultados CPD

El análisis CPD no encontró problemas en el código.

CPD Results

The following document contains the results of PMD's CPD 6.55.0.

CPD found no problems in your source code.

3.4 Resultados Dependency Check

De las 2 dependencias analizadas, se encontró que"log4j-1.2.14.jar" presenta un nivel de severidad "**CRÍTICO**" con 7 vulnerabilidades.

DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance f
event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project: ExpositoTOP

org.example:ExpositoTOP:1.0-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 8.4.2
- *Report Generated On*: Sat, 28 Oct 2023 14:24:14 +0100
- *Dependencies Scanned*: 2 (2 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 7
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing All Dependencies (click to show less)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
jxl-2.6.12.jar		pkg:maven/net.sourceforge.jexcelapi/jxl@2.6.12		0		26
log4j-1.2.14.jar	cpe:2.3:a:apache:log4j:1.2.14:*:*:*:*:*:*:*	pkg:maven/log4j/log4j@1.2.14	CRITICAL	7	Highest	24

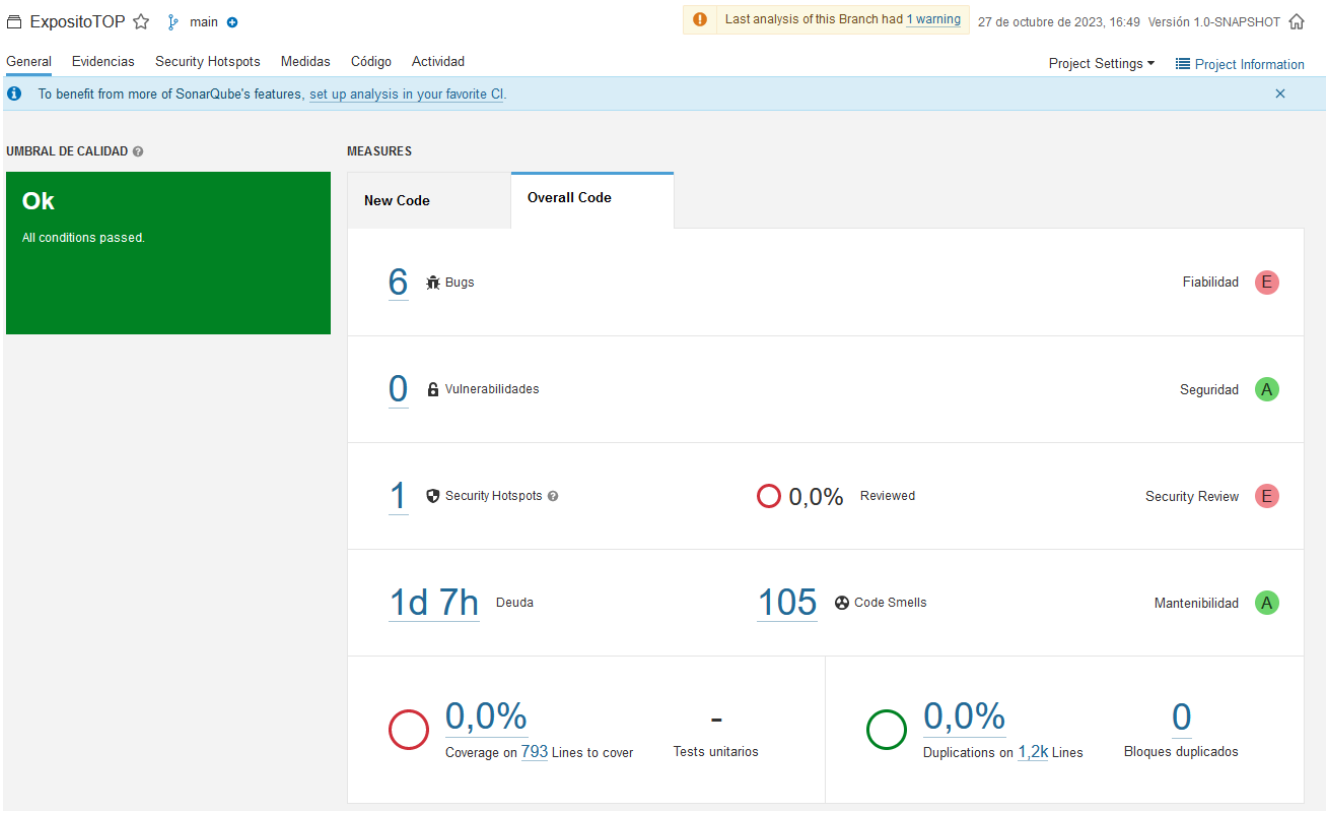
4. Análisis SonarQube

Para realizar un análisis con SonarQube de nuestro proyecto de Maven, creamos un nuevo proyecto en este y generamos un Token y ejecutamos la línea de comando que se nos facilita dentro del directorio de nuestro proyecto (en una misma línea sin /)

mvn clean verify sonar:sonar
-Dsonar.projectKey=ExpositoTOP-Dsonar.host.url=http://localhost:9000
-Dsonar.login=sqp_61c8f109838c5e8c553dc3981c52bf906e3b191d

```
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- sonar:3.10.0.2594:sonar (default-cli) @ ExpositoTOP ---
[INFO] User cache: C:\Users\Adn\.sonar\cache
[INFO] SonarQube version: 9.9.2.77730
[INFO] Default locale: "es_ES", source code encoding: "UTF-8"
[INFO] Load global settings
[INFO] Load global settings (done) | time=73ms
[INFO] Server id: 147B411E-AYsEcv1ap1iGdiHdqA2x
[INFO] User cache: C:\Users\Adn\.sonar\cache
[INFO] Load/download plugins
[INFO] Load plugins index
[INFO] Load plugins index (done) | time=42ms
[INFO] Load/download plugins (done) | time=130ms
[INFO] Process project properties
[INFO] Process project properties (done) | time=10ms
[INFO] Execute project builders
[INFO] Execute project builders (done) | time=1ms
[INFO] Project key: ExpositoTOP
[INFO] Base dir: C:\Users\Adn\IdeaProjects\ExpositoTOP
[INFO] Working dir: C:\Users\Adn\IdeaProjects\ExpositoTOP\target\sonar
[INFO] Load project settings for component key: 'ExpositoTOP'
```

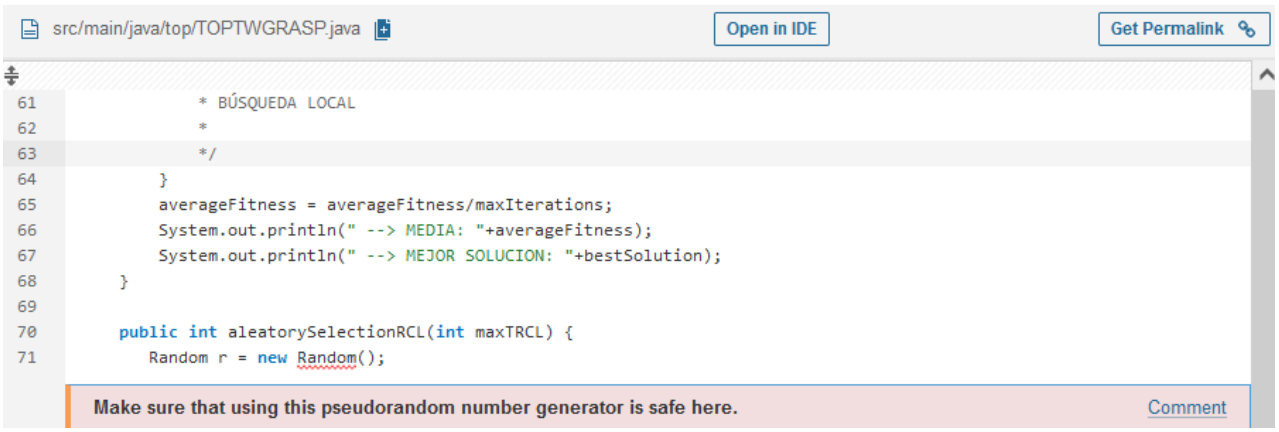
Obteniendo los siguientes resultados:



El informe de SonarQube indica que todos los umbrales de calidad se cumplieron satisfactoriamente. Se detectaron 6 errores de software (bugs), 0 vulnerabilidades, 1 Security Hotspot, y se identificaron 105 code smells. Esto nos da unas calificaciones de E en Fiabilidad, A en Seguridad, E en Revisión de Seguridad y una A en Mantenibilidad.

5. Problemas de Seguridad

El security Hotspot detectado es el siguiente:



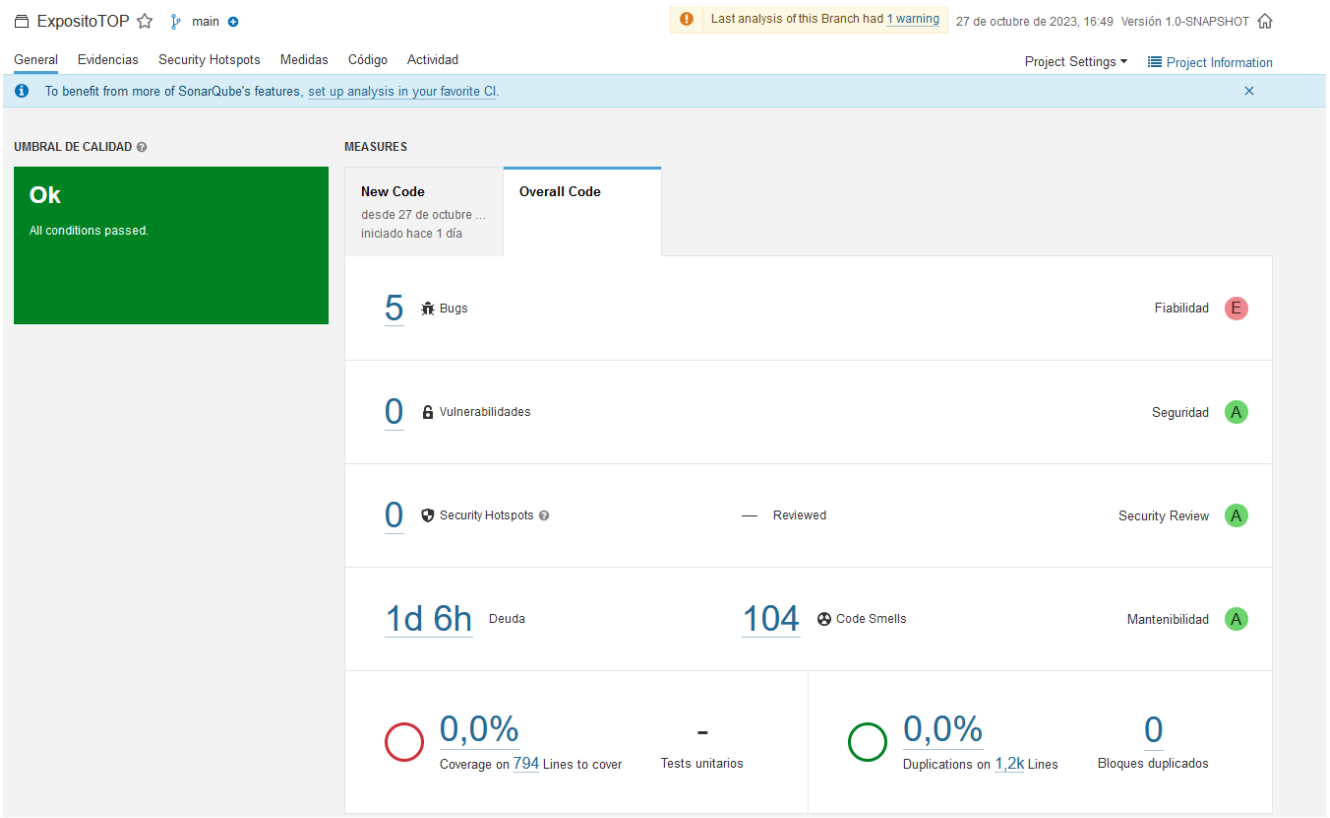
Se nos indica que en la línea 71 de *TOPTWGRASP.java,* se crea un objeto de la clase **Random**. El uso de la clase Random para generar números aleatorios en contextos donde la seguridad es importante no es recomendado, ya que los números generados no son criptográficamente seguros. Esto significa que un atacante podría predecir los números generados y explotar esto en ciertos escenarios. Se nos recomienda cambiar a una fuente de números aleatorios criptográficamente seguros, como **SecureRandom**.

```
public int aleatorySelectionRCL(int maxTRCL) {
    Random r = new Random();
    int low = 0;
    int high = maxTRCL;
    int posSelected = r.nextInt( bound: high-low) + low;
    return posSelected;
}
```

Una vez realizado el cambio, el código queda de la siguiente forma :

```
4 usages
public int aleatorySelectionRCL(int maxTRCL) {
    SecureRandom random = new SecureRandom();
    byte bytes[] = new byte[20];
    random.nextBytes(bytes);
    int low = 0;
    int high = maxTRCL;
    int posSelected = random.nextInt( bound: high-low) + low;
    return posSelected;
}
```

Volvemos a pasar el SonarQube y obtenemos los siguientes resultados:



Como se puede observar hemos solucionado el security hotspot, por lo que pasamos de una Calificación de E a una A.

6. Link del Repositorio

https://github.com/Adn086/LDH