Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Práctica 3.

Herramientas de Calidad del Producto Software y Documentación. Sonarqube, Maven y Doxygen.

Álvaro González Rodríguez
(alu0101202556@ull.edu.es)

Índice:

1. Plugins del proyecto en Maven.

Se han configurado varios plugins de reporte y de construcción, siendo estos los siguientes:

Plugins de reporte:

- Checkstyle: Este plugin se encarga de ayudar a los programadores a escribir código en Java que cumpla las guías de estilo, por lo que automáticamente chequea todo el código. Como resultado de este análisis se tendrá un informe completo de todo lo que se podría cambiar para que se cumpla un estándar de codificación.

- PMD: Permite analizar el código de un proyecto en Java para encontrar fallos de programación comunes como son variables sin usar, bloques catch vacíos, creaciones de objetos innecesarios entre otras funcionalidades.

- Dependency-Check: Se trata de un plugin desarrollado por OWASP, el cual, como su nombre indica, detecta vulnerabilidades contenidas dentro de las dependencias del proyecto.

- Javadoc: Utiliza la herramienta Javadoc para generar la documentación del proyecto.

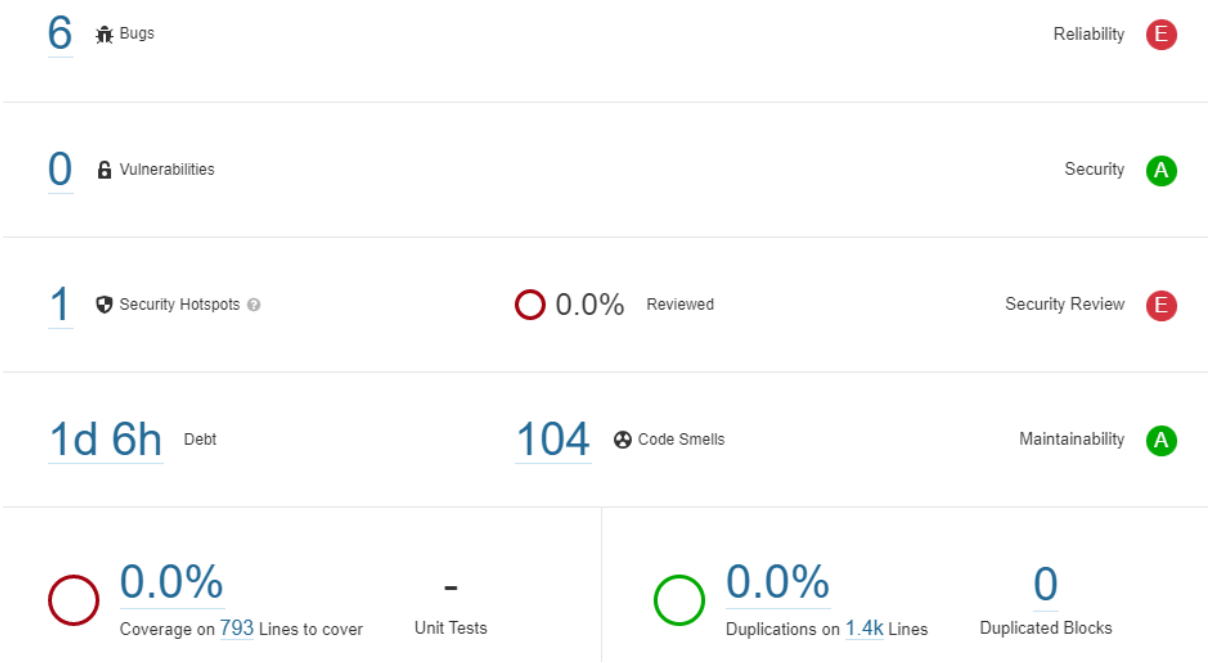
Plugins de construcción:

- Clean, site, deploy, compiler, ...: Plugins que corresponden a las distintas fases del ciclo de vida de Maven, son prácticamente necesarios para el funcionamiento de la herramienta de automatización.

- AntRun: Como este proyecto ya tiene un build.xml de Ant es bastante util tener preparado en Maven un plugin que permite ejecutar tareas de Ant desde Maven.

2. Maven y Sonarqube.

Una vez se ha creado y desarrollado el fichero pom.xml ya se puede ejecutar la herramienta Sonarqube con soporte para Maven. El resultado del análisis con el proyecto ExpositoTOP es el siguiente:



Se puede observar que en 1400 lineas de código, aproximadamente, se han encontrado 6 bugs, estando 4 en el directorio main/java/es/ull/esit/utilities, concretamente repartidos 2 en ExpositoUtilities.java, siendo uno de un nivel mayor y otro bloqueador, y los otros dos en PowerSet.java, siendo uno de mayor nivel y otro de menor nivel. Luego, los otros 2 bugs restantes se encuentran en main/java/top/, uno en el fichero TOPTWGRASP.java, siendo un bug crítico, y el otro en TOPTWSolution.java, siendo un bug de mayor nivel.

El único problema grave de seguridad que ha encontrado Sonarqube se encuentra en src/main/java/top/TOPTWGRASP.java. Se trata del uso del *java.util.Random,* que, como funciona con un número generado de forma pseudoaleatoria, puede comprometer a la seguridad del código.

Además, se han detectado 0 problemas de vulnerabilidad y solamente 1 día y 4 horas de deuda técnica con 104 code smells, que relativamente son pocos en comparación con la cantidad de líneas de código que tiene el proyecto, asi que tiene una buena mantenibilidad. Por último, la complejidad ciclomática es de 257 mientras que la cognitiva es de 293.

3. Solución a los problemas del proyecto.

- Security Hotspot:

Como ya se comentó anteriormente, para solucionar este problema hay que quitar del código todas las referencias a java.util.Random y sustituirlas por una función más segura, como, por ejemplo, java.security.SecureRandom. La siguiente sentencia sería un buen generador de un número aleatorio:

```
SecureRandom random = new SecureRandom();

byte bytes[] = new byte[20];

random.nextBytes(bytes);
```

Una vez realizado los cambios el security hotspot debería de estar solucionado.

- Bugs:

El primer bug que encontramos en ExpositoUtilities.java es en la línea 44 y es el siguiente:

A "NullPointerException" could be thrown; "reader" is nullable here.

Básicamente se trata de que hay un objeto que es probable que sea nulo y este realiza una llamada a una función. Por lo que la solución más practica es asegurar que este objeto no sea nulo cuando vaya a realizar la llamada a la función, esto se consigue añadiendo la siguiente linea antes de dicha llamada:

```
assert reader != null;
```

El segundo bug del fichero está en la línea 82 y es el siguiente:

Use try-with-resources or close this "BufferedWriter" in a "finally" clause.

Todos los objetos o streams que implementan una interfaz AutoCloseable necesitan ser cerrados una vez usados, cosa que el código no realiza, por lo que tenemos que implementar un bloque finally solucionar el bug, por lo que el código quedaría de la siguiente manera:

```
BufferedWriter writer = new BufferedWriter(new FileWriter(file));
try {
    writer.write(text);
    writer.flush();
} catch (Exception ex) {
    Logger.getLogger(ExpositoUtilities.class.getName()).log(Level.SEVERE, msg: null, ex);
} finally {
    writer.close();
}
```

Otros dos bugs se encuentran en el fichero PowerSet.java en las líneas 26 y 51. El primero se trata de un bug de poca importancia, basicamente comenta que toda implementación del java.util.Iterator.next() debe de lanzar una excepción.

Add a "NoSuchElementException" for iteration beyond the end of the collection.

Por lo que añadiendo las siguientes líneas de código el error debería de estar solucionado:

```
if(!hasNext()){
      throw new NoSuchElementException();
}
```

El siguiente bug es bastante más complejo:

Refactor this code so that the Iterator supports multiple traversal

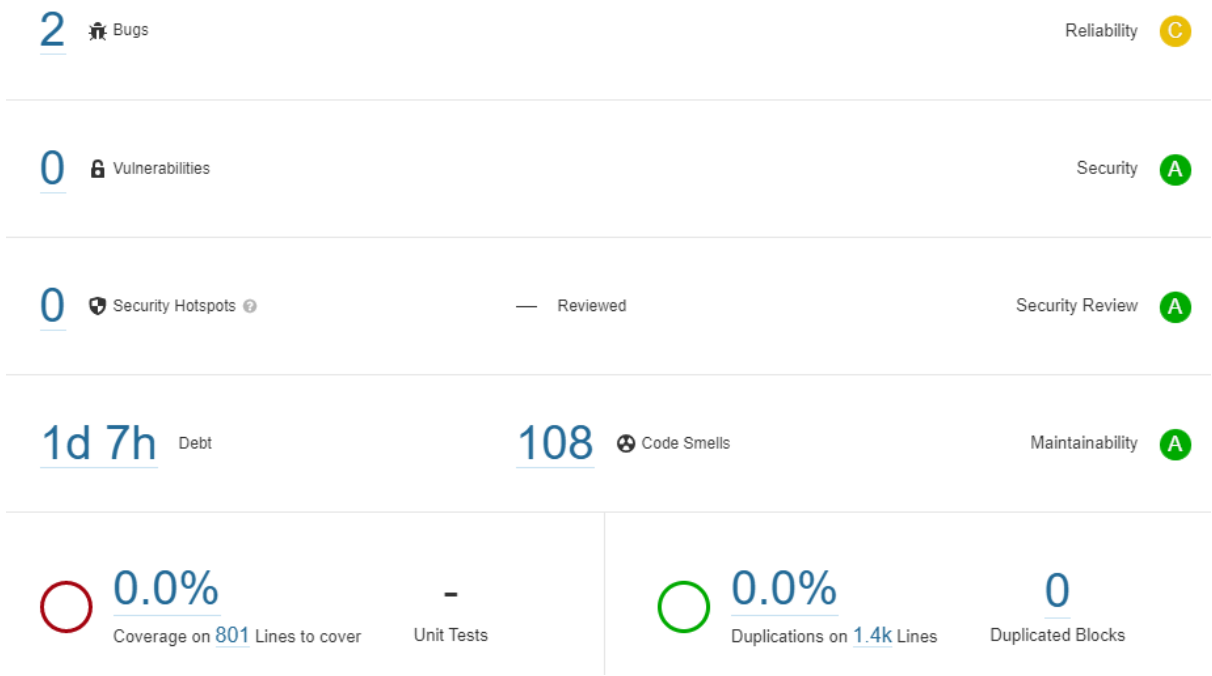
Ocurre cuando tenemos un código que implementa más de un tipo de iterador y ponemos *return this* en el método *iterator()*, por lo que tenemos que refactorizar el código completo para que pueda devolver un tipo u otro.

El siguiente bug se encuentra en TOPTWGRASP.java y está relacionado con el security hotspot que ya ha sido corregido.

Por último tenemos el siguiente error que se encuentra en la línea 56 de TOPTWSolution.java:

Either override Object.equals(Object), or rename the method to prevent any confusion.

Hay que cambiar el nombre del método o añadir un @Override porque puede dar confusión con el equal de Java. Así que, una vez aplicado los arreglos al código (con la excepción de los dos últimos bugs) y después de haber vuelto a lanzar Sonarqube, nos queda el siguiente panel:



Donde se han solucionado 4 bugs y se ha eliminado el Security Hotspot a cambio de añadir 4 nuevos Code Smells al código.

4. Documentación en Doxygen

Respecto a la documentación en Doxygen, solamente se ha documentado en estilo Javadoc los ficheros que se encuentran en el paquete utilities y utils. Esta documentación se encuentra dentro de la carpeta doc y se ha hecho uso de Doxywizard para su elaboración. Para poder generarla en estilo Javadoc se ha tenido que activar la opción JAVADOC_BANNER y JAVADOC_AUTOBRIEF en el asistente de doxygen. La documentación generada en PDF se encuentra en doc/latex/refman.pdf.

5. Enlace al código en GitHub.

https://github.com/AlvaroGonzalezRodriguez/ExpositoTOP-Lab.git