Universidad de La Laguna

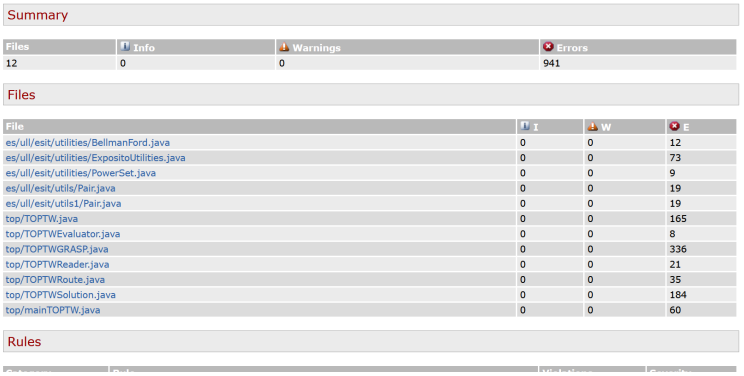
Informe 3. Práctica: Sonarqube, Maven & Doxygen

La calidad del software es un factor crítico en el desarrollo y mantenimiento de aplicaciones, ya que determina su confiabilidad, eficiencia y facilidad de mantenimiento. A medida que los proyectos crecen en complejidad, se hace necesario contar con herramientas que permitan evaluar y gestionar métricas de calidad del producto de manera efectiva. En este contexto, SonarQube y Maven destacan como dos soluciones esenciales. SonarQube facilita el análisis estático del código fuente, detectando problemas de seguridad, errores y áreas de mejora en cuanto a estándares de calidad. Maven, por su parte, proporciona un sistema robusto para la gestión y construcción de proyectos Java, permitiendo la integración con otros sistemas de control de calidad y facilitando el proceso de generación de entregables de software.

Apartado 6: Plugins de Maven

Estos son los plugins que he usado para analizar el código de los diferentes archivos java presentes en el proyecto:

- 1. Checkstyle: El plugin **Checkstyle** es una herramienta de análisis de código estático que ayuda a mantener un estilo de código consistente y mejorar su legibilidad en proyectos Java



Summary

Files	Info	Warnings	Errors
12	0	0	941

Files

File	I	W	E
es/ull/esit/utilities/BellmanFord.java	0	0	12
es/ull/esit/utilities/ExpositoUtilities.java	0	0	73
es/ull/esit/utilities/PowerSet.java	0	0	9
es/ull/esit/utils/Pair.java	0	0	19
es/ull/esit/utils1/Pair.java	0	0	19
top/TOPTW.java	0	0	165
top/TOPTWEvaluator.java	0	0	8
top/TOPTWGRASP.java	0	0	336
top/TOPTWReader.java	0	0	21
top/TOPTWRoute.java	0	0	35
top/TOPTWSolution.java	0	0	184
top/mainTOPTW.java	0	0	60

Rules

Como se puede observar en la imagen, vemos una pila de errores en prácticamente todas las clases. Si entramos a analizar cada una de ellas, vemos que muchos errores son de tamaño de carácteres en

lineas o de tipo misceláneo, es decir, que los parámetros están mal definidos.

- 2. PMD: El plugin PMD es una herramienta de análisis de código estático que se utiliza para identificar problemas de calidad en el código fuente Java, como errores comunes de programación, código duplicado y prácticas que afectan el rendimiento o la legibilidad

PMD Results

The following document contains the results of PMD 7.3.0.

Violations By Priority

Priority 3

es/ull/esit/utilities/ExpositoUtilities.java

Rule	Violation	Line
UnusedPrivateMethod	Avoid unused private methods such as 'getFirstAppearance(int[], int)'.	23
CollapsibleIfStatements	This if statement could be combined with its parent	90–93
EmptyCatchBlock	Avoid empty catch blocks	217–218
EmptyCatchBlock	Avoid empty catch blocks	226–227

es/ull/esit/utilities/PowerSet.java

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

Por ejemplo, como se puede ver en la foto, el método Collection.toArray() va a lanzar una excepción.

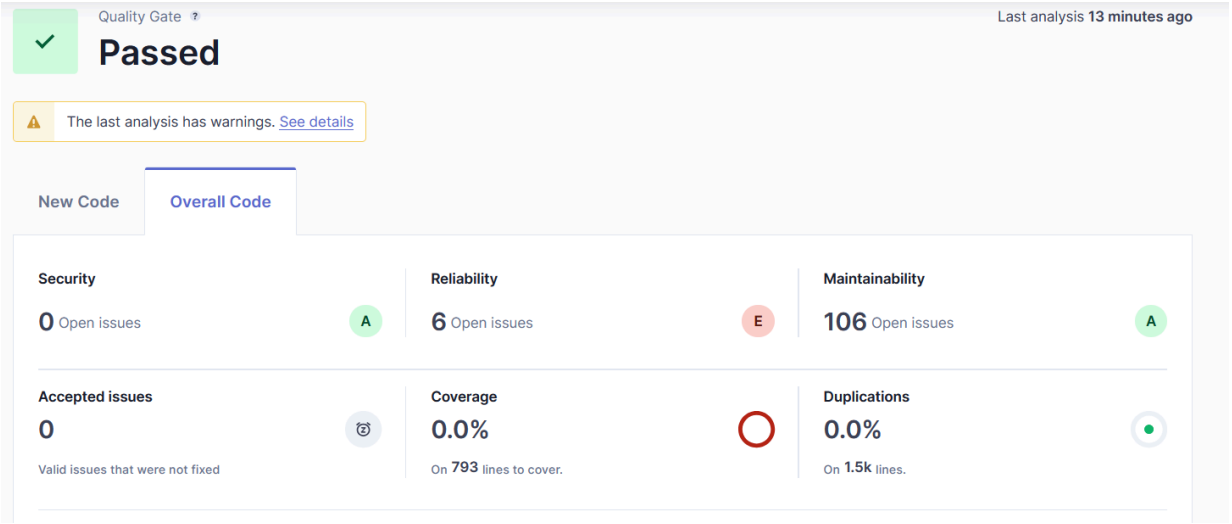
Apartado 7 y 8: SonarQube y Maven

Para poder usar el analizador de código estático sonarQube en tu máquina desde Maven, debes tener corriendo un servidor SonarQube en tu ordenador. Así se ve el pom.xml para que pueda ejecutar sonar scanner:

```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <sonar.projectKey>ExpositoTOP</sonar.projectKey>
    <sonar.projectName>ExpositoTOP</sonar.projectName>
    <sonar.projectVersion>1.0</sonar.projectVersion>
    <sonar.sources>src/main/java</sonar.sources>
    <sonar.language>java</sonar.language>
    <sonar.host.url>http://localhost:9000</sonar.host.url>
    <sonar.token>sqp_0015b5e08eaf3fa15b0a41a8fbc2c859c389a49c</sonar.token>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

Después se ejecuta este comando en la linea de comandos:
mvn clean install sonar:sonar

Una vez hecho esto, y si se ejecuta exitosamente, la página de sonarqube se va actualizar con el código analizado.



Un problema de seguridad que he solucionado en el programa es la de criptografía débil, la cual surge cuando generamos un numero pseudoaleatorio sin seguridad. Para ello he usado una instancia de Random en una aplicación de single-thread creando un atributo privado estático en la clase:

```
private static final Random random = new Random();
```