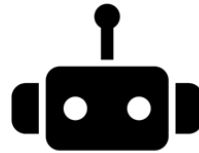




PUCP

SESIÓN DE LABORATORIO 5 SonarQube y JMeter



HORARIO 10M1

Empezaremos a las 8:10 p.m

Gracias!



Calidad de Código

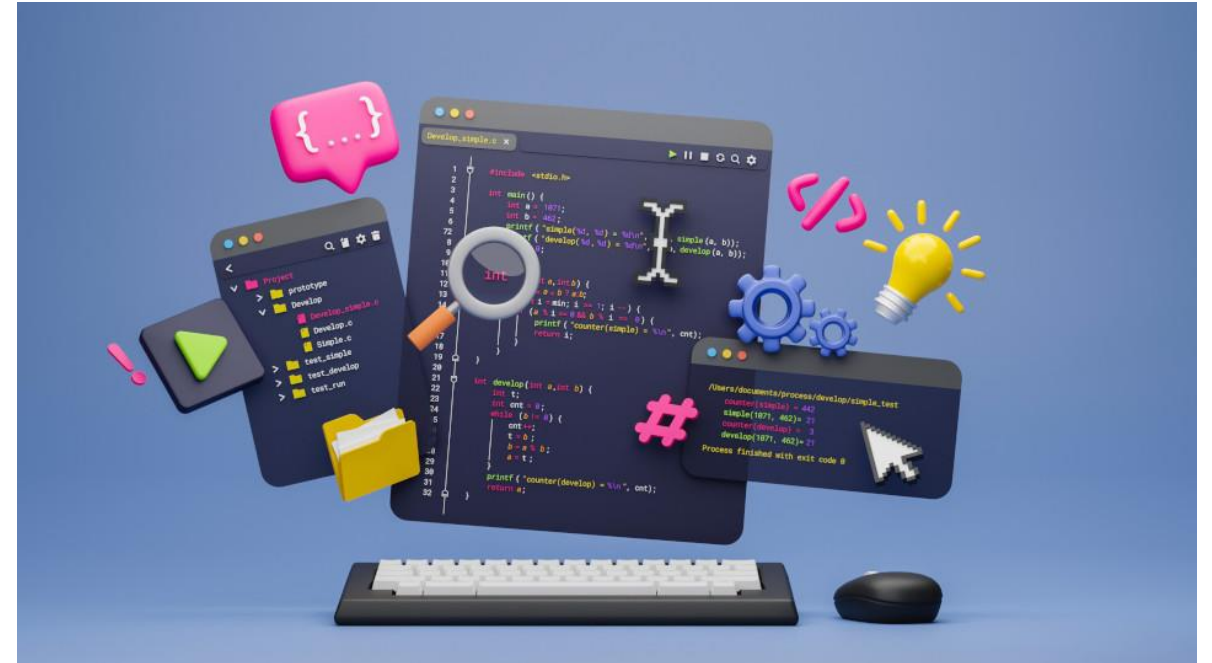
La **calidad del código** se refiere a qué tan bien está escrito y estructurado el código fuente de una aplicación. Un código de calidad es **legible**, **fácil de mantener**, **eficiente** y **seguro**. Esto significa que el código no solo debe cumplir con los requisitos funcionales, sino que también debe ser fácil de entender, modificar y extender en el futuro, sin introducir nuevos problemas o errores.

Claridad: El código debe ser legible y comprensible para otros desarrolladores.

Eficiencia: Debe realizar su función de la manera más óptima posible, sin consumir recursos innecesarios.

Mantenibilidad: Debe permitir cambios y mejoras sin causar errores imprevistos.

Seguridad: El código debe estar libre de vulnerabilidades que puedan ser explotadas.



SonarQube

Es una plataforma de análisis y gestión de la calidad del código que permite identificar problemas en el código fuente, como errores, vulnerabilidades, código duplicado y malas prácticas. Ofrece soporte para múltiples lenguajes de programación y además ayuda a mejorar la mantenibilidad y seguridad del software mediante análisis estáticos y continuos, proporcionando informes detallados y métricas de calidad para garantizar que el código cumpla con los estándares de la organización.



Conceptos Clave en SonarQube

Bugs

Un **bug** es un error en el código que provoca que el software no funcione como debería. Puede manifestarse de diferentes maneras, como resultados incorrectos, fallos en la ejecución, o comportamientos inesperados.

Code Smells

Los **code smells** son indicios de que algo en el diseño o estructura del código no es óptimo, aunque no necesariamente es un bug. Estos problemas no suelen romper el código ni impedir que funcione, pero indican que el código podría ser más difícil de entender, mantener o escalar en el futuro.

Deuda Técnica

La **deuda técnica** es un concepto que se refiere a las consecuencias de tomar atajos en el desarrollo del software para cumplir con fechas de entrega o requisitos inmediatos, pero que a largo plazo resultan en mayor esfuerzo y costos.

Duplicación de Código

La **duplicación de código** ocurre cuando los mismos fragmentos de código o bloques muy similares se repiten en diferentes partes del proyecto. Esto es problemático porque cualquier cambio en una parte del código debe ser replicado manualmente en todas las demás instancias duplicadas, lo que aumenta el riesgo de introducir errores.



Identificación de Bugs

Pruebas unitarias y de integración: Las pruebas automatizadas ayudan a verificar si el código está funcionando como se espera. Si una prueba falla, puede ser un indicio de que hay un bug.

Depuración: Utiliza herramientas de depuración que permiten analizar paso a paso el comportamiento del programa, verificando el flujo de ejecución y los valores de las variables.

Análisis estático de código: Herramientas como **SonarQube** pueden detectar errores comunes antes de que el código sea ejecutado.

Revisión de código por pares: Que otros desarrolladores revisen el código puede ayudar a identificar bugs que tal vez el autor no vio.

Ejemplos de bugs comunes:

- Condiciones incorrectas:** Un error lógico en una condición que hace que el programa tome decisiones incorrectas.
- Errores de referencia nula:** Intentar acceder a un objeto o variable que no ha sido inicializado.
- Errores de concurrencia:** Fallos que ocurren cuando múltiples hilos o procesos acceden simultáneamente a recursos compartidos sin la debida sincronización.



Identificación de Code Smells

Análisis estático de código: Herramientas como SonarQube identifican patrones que son considerados malas prácticas.

Revisiones de código: Los desarrolladores experimentados pueden detectar "code smells" al revisar el código, reconociendo patrones problemáticos basados en su experiencia.

Métricas de código: Utilizar métricas como la **complejidad ciclomática** (que mide cuántos caminos distintos puede tomar el código) puede indicar si una función o clase es demasiado compleja.

Ejemplos de code smells comunes:

Métodos o funciones muy largas: Si una función hace demasiadas cosas, es difícil de entender y mantener.

Nombres confusos o poco descriptivos: Nombres de variables o funciones que no explican claramente su propósito.

Código duplicado: Fragmentos de código repetidos en diferentes lugares, lo que aumenta el riesgo de inconsistencia al realizar cambios.

Clases con demasiadas responsabilidades: Si una clase maneja muchas funciones distintas, viola el principio de responsabilidad única.



Métricas clave en SonarQube

Coverage

La **cobertura de pruebas** mide el porcentaje de código que está cubierto por pruebas automatizadas (como pruebas unitarias o de integración). Evalúa qué tan exhaustivamente el código ha sido probado.

Duplicación de Código

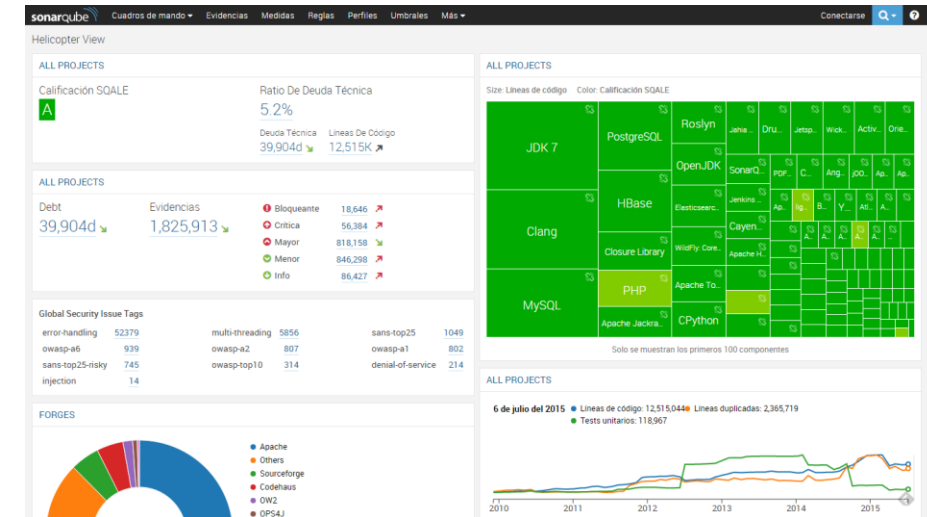
La **duplicación de código** mide la cantidad de código repetido en diferentes partes del proyecto. El código duplicado es un indicador de mala mantenibilidad, ya que los cambios en una parte del código deben replicarse manualmente en todas las copias.

Complejidad Ciclomática

La **complejidad ciclomática** mide la cantidad de rutas lógicas independientes que existen dentro de una función o método. En otras palabras, determina cuán complejo es el flujo de control dentro del código.

Mantenibilidad

La **mantenibilidad** mide la facilidad con la que el código puede ser comprendido, modificado o corregido. SonarQube clasifica la mantenibilidad en cinco niveles, desde A (excelente) hasta E (muy pobre).



Instalaciones

El link pueden encontrarlo directamente en GitHub

Para poder proceder con la descarga deben seleccionar el Community Edition y descomprimir el archivo en una carpeta

Community Edition

Free and open source for productivity & code quality

[Download for free](#)

All of the following features:

- ✓ Static code analysis for 20 languages and frameworks: Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Docker, Kubernetes, Helm Charts, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML, VB.NET, and Azure Resource Manager
- ✓ Detect issues in AI generated code
- ✓ SonarQube server runs in

Developer Edition

Essential capabilities for small teams & businesses

[Download](#)

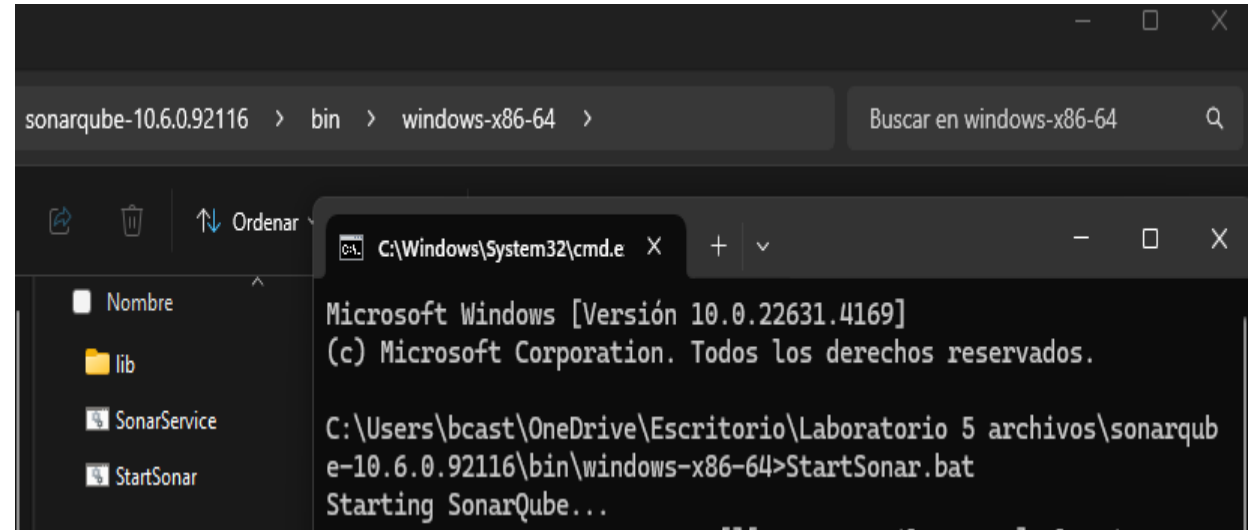
Community Edition plus:

- ✓ Additional languages: C, C++, Obj-C, Swift, ABAP, T-SQL and PL/SQL
- ✓ AutoConfig for C and C++ projects
- ✓ Taint analysis with deeper SAST for Java, C#, JavaScript, and TypeScript
- ✓ Detection of advanced bugs causing runtime errors and crashes in Python & Java

Instalaciones

Una vez la carpeta esta descomprimida a continuación, deben entrar en la consola con cmd e ir a la carpeta donde SonarQube está descomprimido en la dirección : \bin\windows-x86-64 y ejecutar **StartSonar.bat** ahí para que pueda ser ejecutado correctamente

En caso de que se presenten errores es necesario instalar Java 17 :
https://download.oracle.com/java/17/archive/jdk-17.0.8_windows-x64_bin.exe

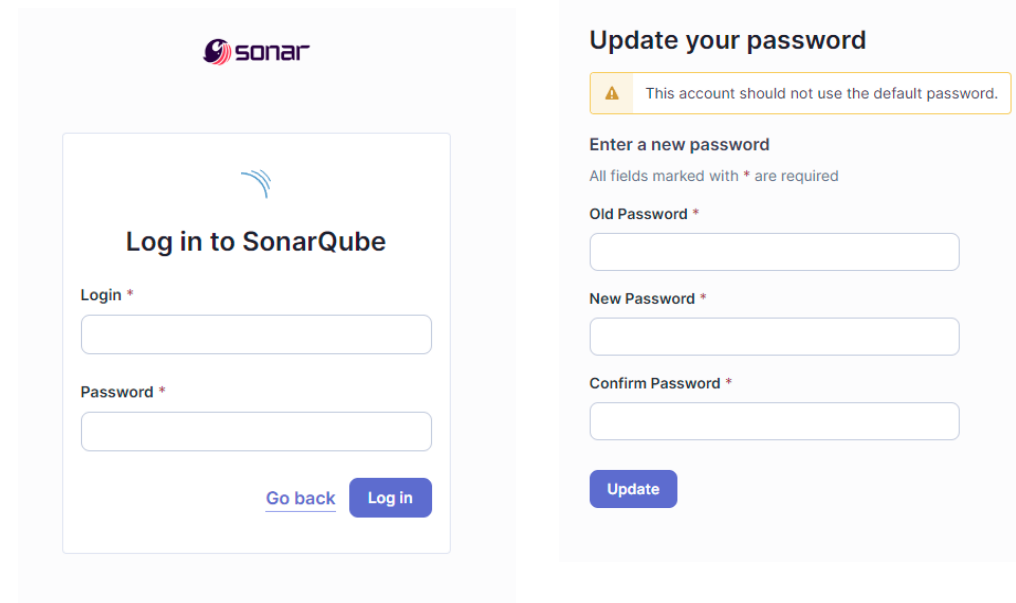


Instalaciones

Hay que verificar que el proceso se haya activado correctamente y luego entrar a localhost:9000

Aquí se te pedirá un usuario y una contraseña; a los cuales son : admin para poder ingresar y poder cambiarla por una de su preferencia

```
WARNING: Please consider reporting this to the maintainers of org.sonar.process.PluginSecurityManager
WARNING: System::setSecurityManager will be removed in a future release
2024.09.26 16:19:56 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up
2024.09.26 16:19:56 INFO app[][o.s.a.SchedulerImpl] SonarQube is operational
```



The image shows two side-by-side screenshots of the SonarQube web interface. The left screenshot is the login page, titled 'Log in to SonarQube', featuring input fields for 'Login *' and 'Password *', a 'Go back' link, and a 'Log in' button. The right screenshot is the 'Update your password' page, which includes a warning message: 'This account should not use the default password.' Below this, it prompts the user to 'Enter a new password' with the note 'All fields marked with * are required'. It contains three input fields: 'Old Password *', 'New Password *', and 'Confirm Password *', followed by an 'Update' button.

Instalaciones

Una vez ya tenemos todo seteado también debemos descargar el sonar-scanner usando la versión de Windows x64 y luego descomprimir la carpeta en un lugar donde recordemos

SonarScanner CLI

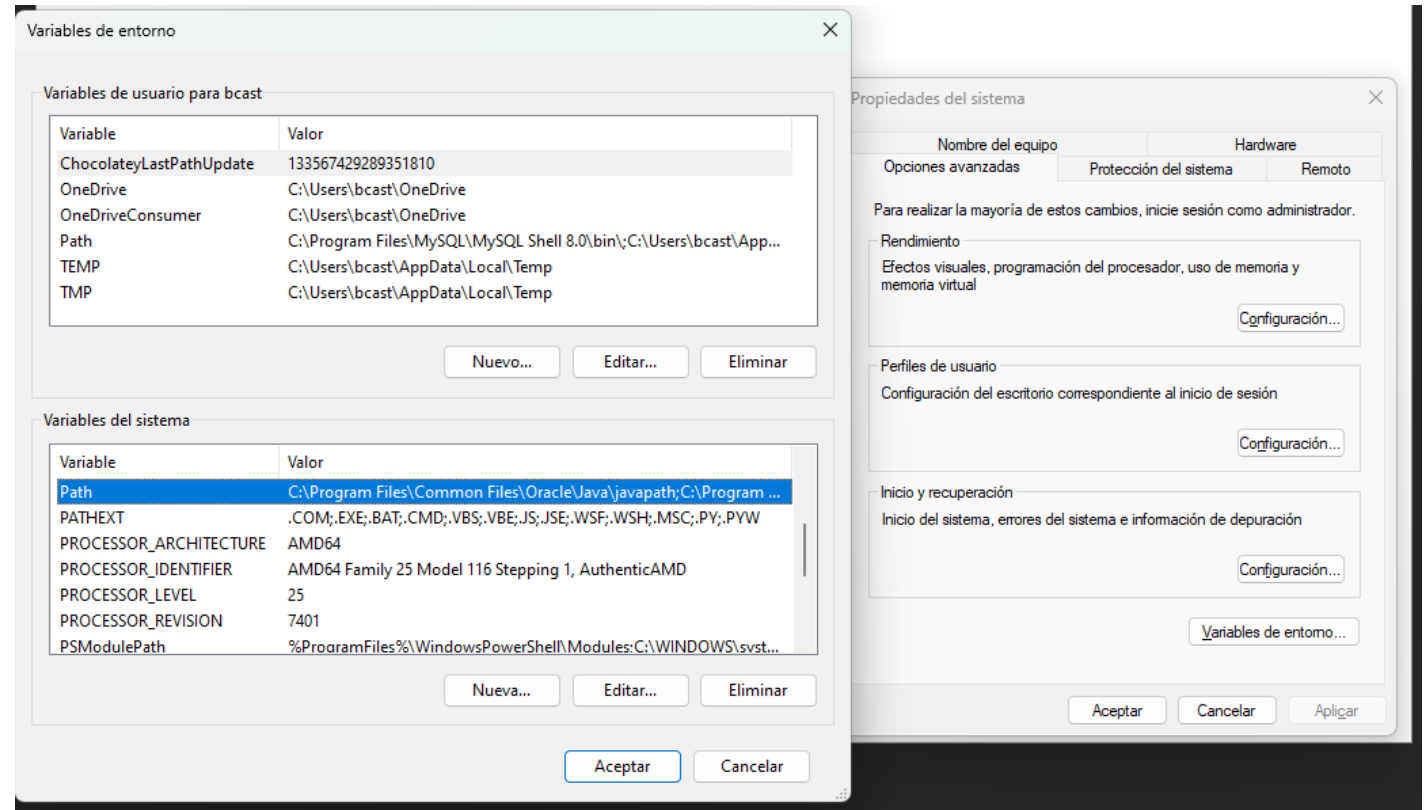
SonarScanner	Issue Tracker	Show more ▾
6.2		2024-09-17
Support PKCS12 truststore generated with OpenSSL		
Download scanner for: Linux x64 Linux AArch64 Windows x64 macOS x64 macOS AArch64 Docker		
Any (Requires a pre-installed JVM)		
Release notes		

Instalaciones

Una vez este descomprimida la carpeta debemos agregar el archivo bin de la misma en el path de las variables de entorno para que pueda ser reconocido por el sistema :

`\sonar-scanner-6.2.0.4584-windows-x64\bin`

Y con esto ya tendríamos seteado todo correctamente



Validaciones

Ahora para empezar las validaciones , en un proyecto Java, crearemos un archivo sonar-project.properties e Introduciremos los datos mostrados y luego guardamos.

En este caso estamos utilizando un proyecto de Microservicios como ejemplo

```
sonar.host.url=http://localhost:9000
sonar.login=admin
sonar.password=pucp2024
sonar.projectKey=SIJ
sonar.projectName=SIJ
sonar.projectVersion=1.0
sonar.language=java
sonar.sources=src/
sonar.java.binaries=build/
```

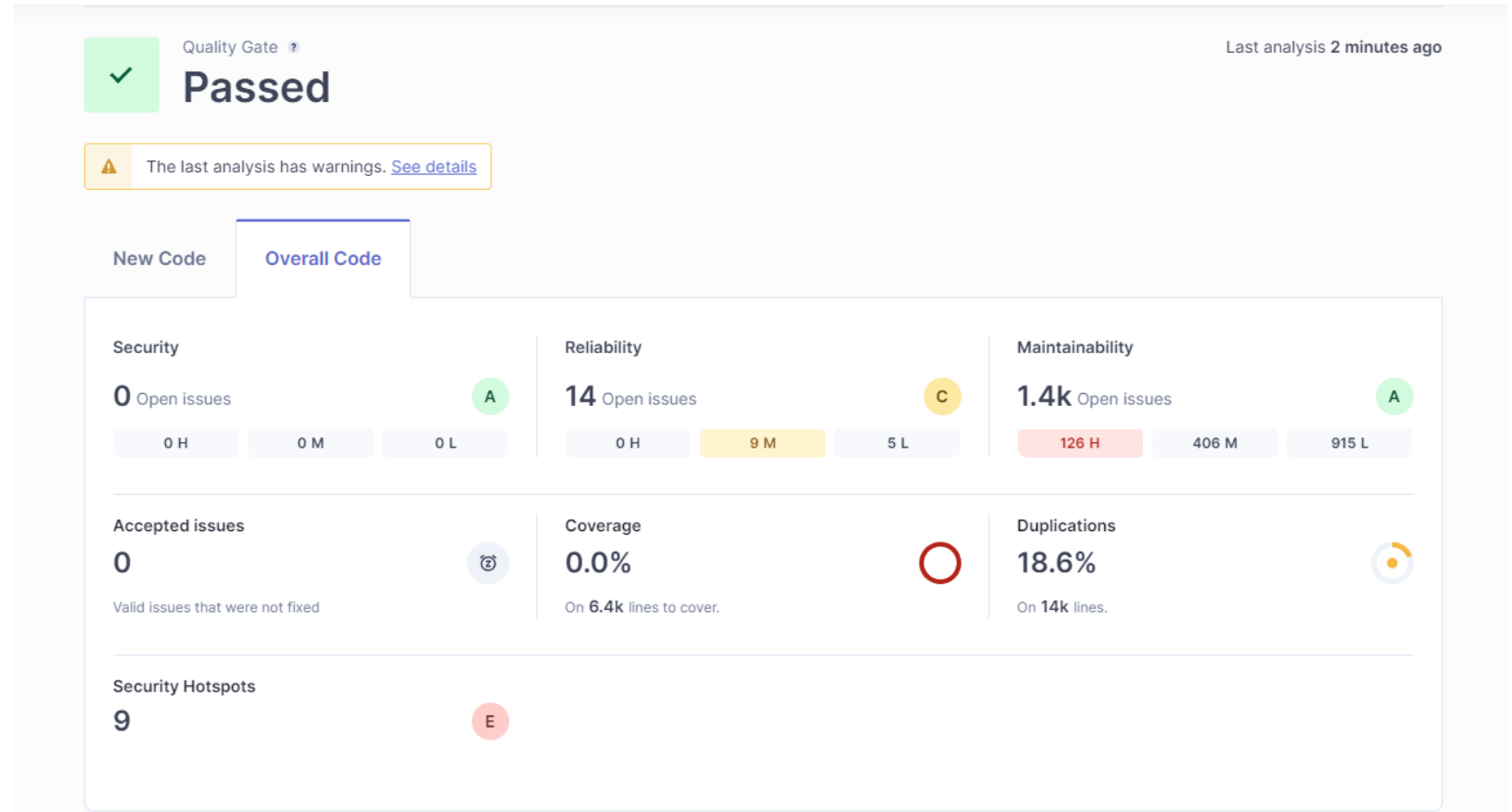
Validaciones

Ahora para poder ejecutar el proyecto debemos ir a la carpeta del proyecto y ejecutar **sonar-scanner**, todo esto sin cerrar el sonarqube que ya fue abierto y esperar a que se ejecute correctamente

```
Load UCFGs: Begin: 2024-09-26T23:04:24.743729400Z, End: 2024-09-26T23:04:24.743729400Z, Duration: 00:00:00.000
18:04:24.751 INFO   js security sensor peak memory: 564 MB
18:04:24.752 INFO   Sensor JsSecuritySensor [security] (done) | time=2ms
18:04:24.752 INFO   ----- Run sensors on project
18:04:24.846 INFO   Sensor Zero Coverage Sensor
18:04:24.888 INFO   Sensor Zero Coverage Sensor (done) | time=43ms
18:04:24.889 INFO   Sensor Java CPD Block Indexer
18:04:25.017 INFO   Sensor Java CPD Block Indexer (done) | time=128ms
18:04:25.019 INFO   SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly
specify it.
18:04:25.036 INFO   CPD Executor 11 files had no CPD blocks
18:04:25.036 INFO   CPD Executor Calculating CPD for 56 files
18:04:25.123 INFO   CPD Executor CPD calculation finished (done) | time=87ms
18:04:25.572 INFO   Analysis report generated in 415ms, dir size=1 MB
18:04:27.893 INFO   Analysis report compressed in 2321ms, zip size=464 KB
18:04:29.455 INFO   Analysis report uploaded in 1562ms
18:04:29.456 INFO   ANALYSIS SUCCESSFUL, you can find the results at: https://sonarcloud.io/dashboard?id=SIJ
18:04:29.457 INFO   Note that you will be able to access the updated dashboard once the server has processed the submitte
d analysis report
18:04:29.457 INFO   More about the report processing at https://sonarcloud.io/api/ce/task?id=AZIwkyY5BzBgaIxETeGX
18:04:30.816 INFO   Sensor cache published successfully
18:04:30.886 INFO   Analysis total time: 1:16.158 s
18:04:30.902 INFO   SonarScanner Engine completed successfully
18:04:31.453 INFO   EXECUTION SUCCESS
18:04:31.453 INFO   Total time: 1:20.778s
```

Validaciones

Una vez haya terminado podemos dirigirnos a <http://localhost:9000/> y veremos el proyecto con el que vamos a trabajar



Validaciones – Code Smells

The screenshot displays a code quality tool interface. On the left, a sidebar shows filters for Security (0), Reliability (0), and Maintainability (1.4k). Under 'Severity', there are options for High (126), Medium (406), and Low (915). Under 'Type', there are options for Bug (14), Vulnerability (0), and Code Smell (1.4k). The 'Code Smell' option is selected. Below the sidebar, there are sections for 'Scope', 'Status', and 'Security Category'. The main area shows a list of code smells. The first smell is 'Complete the task associated to this TODO comment.' with a severity of 'Maintainability' (Low) and a category of 'Intentionality'. The second smell is 'Replace this use of System.out by a logger.' with a severity of 'Maintainability' (Medium) and a category of 'Adaptability'. The third smell is 'Complete the task associated to this TODO comment.' with a severity of 'Maintainability' (Low) and a category of 'Intentionality'. The fourth smell is 'Replace this use of System.out by a logger.' with a severity of 'Maintainability' (Medium) and a category of 'Adaptability'. The interface also includes a 'Bulk Change' button, a 'Select issues' dropdown, a 'Navigate to issue' dropdown, and a summary of '1,447 issues' and '20d effort'.

Security 0
Reliability 0
Maintainability 1.4k

Severity ?
High 126
Medium 406
Low 915

Type 1 x
Bug 14
Vulnerability 0
Code Smell 1.4k

Add to selection Ctrl + click

Scope
Status
Security Category

Bulk Change

Select issues Navigate to issue 1,447 issues 20d effort

src/pe/gob/oefa/sij/comun/util/Log4jListener.java

Complete the task associated to this TODO comment. Intentionality
Maintainability Low
Open Not assigned L18 • 0min effort • 4 minutes ago • Code Smell • Info

Replace this use of System.out by a logger. Adaptability
Maintainability Medium
Open Not assigned L19 • 10min effort • 4 minutes ago • Code Smell • Major

Complete the task associated to this TODO comment. Intentionality
Maintainability Low
Open Not assigned L25 • 0min effort • 4 minutes ago • Code Smell • Info

Replace this use of System.out by a logger. Adaptability
Maintainability Medium
Open Not assigned L30 • 10min effort • 4 minutes ago • Code Smell • Major

Validaciones – Bugs

The screenshot displays a software issue tracking interface. On the left, a sidebar shows filters for Severity (High: 0, Medium: 9, Low: 5) and Type (Bug: 14, Vulnerability: 0, Code Smell: 1.4k). The 'Bug' type is selected. The main area shows a list of bugs with details for three specific issues:

- Bug 1:** `src/pe/gob/oefa/sij/comun/util/Util.java`. Issue: `Make "formatter" an instance variable.` Severity: Reliability. Status: Open. Effort: L30 (15min). Category: Bug, Major.
- Bug 2:** `src/pe/gob/oefa/sij/informe/mbean/InformeConsultaMBean.java`. Issue: `Make "formatterHour" an instance variable.` Severity: Reliability. Status: Open. Effort: L31 (15min). Category: Bug, Major.
- Bug 3:** `src/pe/gob/oefa/sij/informe/mbean/InformeRegistroMBean.java`. Issue: `Cast one of the operands of this division operation to a "double".` Severity: Reliability. Status: Open. Effort: L433 (5min). Category: Bug, Minor.