

SonarQube

Berenike Banek, Mateusz Bielówka >>>

Czym jest SonarQube

SonarQube to narzędzie do analizy kodu, które pomaga w identyfikacji błędów, problemów z bezpieczeństwem, duplikacji kodu i innych kwestii związanych z jakością kodu.

Zalety SonarQube: Umożliwia automatyczną analizę kodu, co oszczędza czas i wysiłek, oraz prezentuje wyniki w przystępny sposób za pomocą interfejsu graficznego.

Komponenty SonarQube: Składa się z serwera, bazy danych, interfejsu webowego oraz narzędzia Sonar Scanner, które analizuje kod źródłowy.

Jak działa SonarQube



Developer



Sonar Scanner zbiera wymagane informacje z kodu źródłowego,
Zbiera odpowiednich dla projektu reguły,
Generuje raporty,
Umieszcza raporty w Bazie Danych,
Wyświetla wykresy w GUI



**SonarQube
Server**



Instalacja (Windows)

- Wymagania: Java 17.
- Zmienną środowiskową PATH edytować tak, aby na pierwszej pozycji zawierała katalog *bin* Javy 17.
- Warto dodać/edytować zmienną JAVA_HOME na *[ścieżka]\Java\jdk-17*.
- SonarQube i SonarScanner z oryginalnej strony pobrać i wypakować.
 - Nazwa ścieżki nie może zawierać nawiasów oraz winna mieć pełne uprawnienia rozporządzania plikami.
- Uruchomić StartSonar.bat przy pomocy wiersza poleceń uruchomionego z poziomu administratora
(np. `$ cd C:\SonarQube\sonarqube-10.7.0.96327\bin\windows-x86-64`
`$ StartSonar.bat`)
- Zalogować się do serwera na <http://localhost:9000> (domyślnie login=admin, hasło=admin)
- Zmienić hasło do logowania

Tworzenie projektu

1 of 2

Create a local project

Project display name *



Project key *



Main branch name *

The name of your project's default branch [Learn More](#) 

Cancel

Next

Tworzenie projektu

2 of 2



Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project



Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

Generowanie raportu

Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

 With Jenkins

 With GitHub Actions

 With Bitbucket Pipelines

 With GitLab CI

 With Azure Pipelines

Other CI

SonarQube integrates with your workflow no matter which CI tool you're using.

Locally

Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.

Generowanie raportu

1 Provide a token

Generate a project token

Use existing token

Token name ?

Expires in

Analyze "myJavaProject" 1

30 days



Generate



Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#)  for more information.

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

Generowanie raportu

2 Run analysis on your project

What option best describes your project?

Maven

Gradle

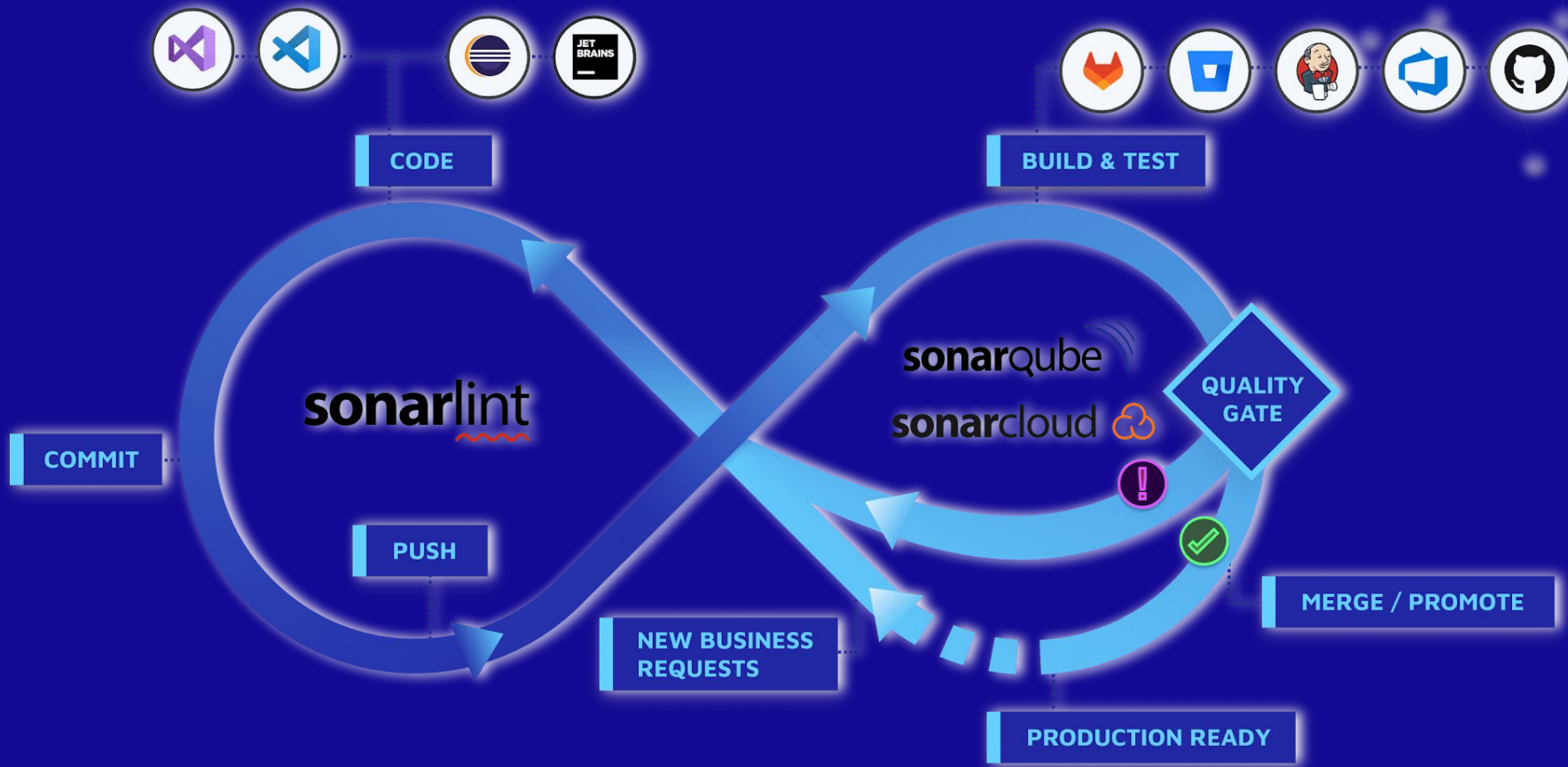
.NET

Other (for JS, TS, Go, Python, PHP, ...)

Execute the Scanner for Maven

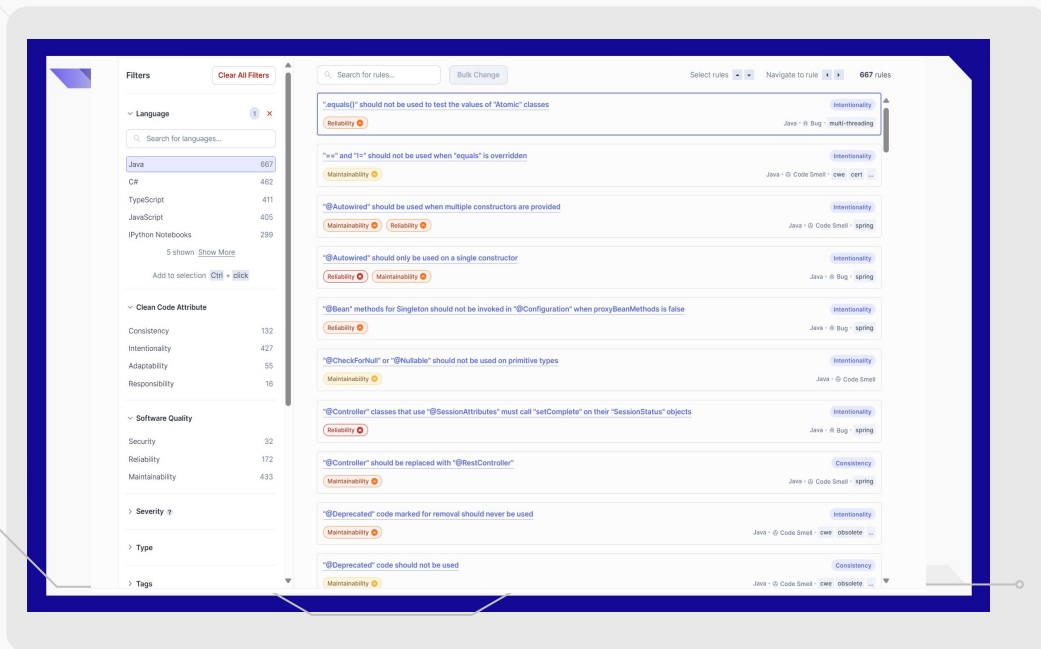
Running a SonarQube analysis with Maven is straightforward. You just need to run the following command in your project's folder.

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=myJavaProject \
  -Dsonar.projectName='myJavaProject' \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.token=sqp_0f7472c5ff5c44b3ffbc0a9da45ac5ef482726a4
```



Koncepty

<u>Clean Code</u>	Kod, który sprawia, że oprogramowanie jest solidne, bezpieczne i łatwe do utrzymania
<u>Bug</u>	Problem, który oznacza, że coś jest nie tak w kodzie. Jeśli nie spowodował jeszcze awarii, prawdopodobnie spowoduje ją w najgorszym możliwym momencie. Należy go naprawić jak najszybciej
<u>Code Smell</u>	Problem związany z utrzymaniem kodu. Pozostawienie go bez zmian sprawi, że w najlepszym przypadku programiści będą mieli trudności z wprowadzaniem zmian, a w najgorszym - mogą dodać nowe błędy przez niejasność kodu.
<u>Debt</u>	Czas potrzebny do naprawy wszystkich problemów związanych z utrzymaniem kodu oraz usunięcia code smells
<u>Vulnerability</u>	Problem związany z bezpieczeństwem, który stanowi potencjalne „tylne drzwi” dla atakujących.



Issues

Są to konkretne problemy, które SonarQube wykrywa w analizowanym projekcie na podstawie wcześniej zdefiniowanych reguł.

Issues

Overview

Issues

Security Hotspots

Measures

Code

Activity

Project Settings ▾

Project Information

My Issues

All

Filters

Issues in new code

▼ Clean Code Attribute

Consistency0

Intentionality2

Adaptability1

Responsibility0

▼ Software Quality

Security0

Reliability0

Maintainability3

> Severity ?

☐ Bulk Change

Select issues ▴ ▾

Navigate to issue ◀ ▶

3 issues

20min effort

CardGameWithModules1-commons/src/main/java/org/example/CryptUtil.java

☐ Add a private constructor to hide the implicit public one.

Intentionality

Maintainability ⚙

design +

☐ Open ▾

☐ Not assigned ▾

L5 • 5min effort • 23 hours ago • 🐞 Code Smell • 🚨 Major

CardGameWithModules1-commons/src/main/java/org/example/Main.java

☐ Replace this use of System.out by a logger.

Adaptability

Maintainability ⚙

bad-practice cert +

☐ Open ▾

☐ Not assigned ▾

L5 • 10min effort • 23 hours ago • 🐞 Code Smell • 🚨 Major

Main/src/test/java/org/example/DeckTest.java

☐ This block of commented-out lines of code should be removed.

Intentionality

Maintainability ⚙

unused +

☐ Open ▾

☐ Not assigned ▾

L45 • 5min effort • 23 hours ago • 🐞 Code Smell • 🚨 Major

Rules

SonarQube dzieli reguły ze względu na m.in. język programowania, czy na to, jak poważny jest problem. Wyróżnia on też typy reguł: security, reliability, maintainability. Dzięki temu możemy je łatwo przefiltrować, aby do analizy naszego projektu były brane pod uwagę tylko te, które nas interesują. Możemy również budować własne zasady poprzez stworzenie pluginu do SonarQube.

Filters

Clear All Filters

> Language1 X

> Clean Code Attribute

▼ Software Quality

Security32

Reliability172

Maintainability433

> Severity ?

Rules

".equals()" should not be used to test the values of "Atomic" classes

Intentionality

Reliability ⬆

Java • 🐞 Bug • multi-threading

"==" and "!=" should not be used when "equals" is overridden

Intentionality

Maintainability ⬇

Java • 🕒 Code Smell • cwe cert ...

"@Autowired" should be used when multiple constructors are provided

Intentionality

Maintainability ⬇

Reliability ⬆

Java • 🕒 Code Smell • spring

"@Autowired" should only be used on a single constructor

Intentionality

Reliability ⬆

Maintainability ⬇

Java • 🐞 Bug • spring

"@Bean" methods for Singleton should not be invoked in "@Configuration" when proxyBeanMethods is false

Intentionality

Reliability ⬆

Java • 🐞 Bug • spring

Rules

".equals()" should not be used to test the values of "Atomic" classes [🔗](#)

Rule ID: java:S2204 • Analysis scope: main sources • Rule repo: Sonar (Java) • Effort: 5min

🐛 Bug • 🚨 Major • multi-threading +

Why is this an issue?

How can I fix it?

More info

The `equals` method in `AtomicInteger` and `AtomicLong` returns `true` only if two instances are identical, not if they represent the same number value.

This is because `equals` is not part of the API contract of these classes, and they do not override the method inherited from `java.lang.Object`. Although both classes implement the `Number` interface, assertions about `equals` comparing number values are not part of that interface either. Only the API contract of implementing classes like `Integer`, `Long`, `Float`, `BigInteger`, etc., provides such assertions.

Clean Code attribute

Intentionality | Clear

Software qualities
impacted

Reliability 📈

Code coverage

Pokrycie kodu to sposób na sprawdzenie ile linii naszego kodu jest sprawdzanych przez testy. Aby mieć pewność, że nasz program działa dobrze, powinniśmy osiągnąć co najmniej 80% pokrycia. Dobrym narzędziem do jego monitorowania jest JaCoCo.

JaCoCo wygeneruje dla nas raport pokrycia kodu (folder `target/site/jacoco`), oraz plik `target/jacoco.exec`, który pobierze SonarQube a następnie wyświetli dokładną analizę, pokazując które klasy i które linijki mogą być podatne na błędy.

New Code 1 failed

Overall Code

Security

0 Open issues

A

Reliability

0 Open issues

A

Maintainability

8 Open issues

A

Accepted issues

0

B

Valid issues that were not fixed

Coverage

63.7%

On 69 lines to cover.

Duplications

0.0%

On 456 lines.

Security Hotspots

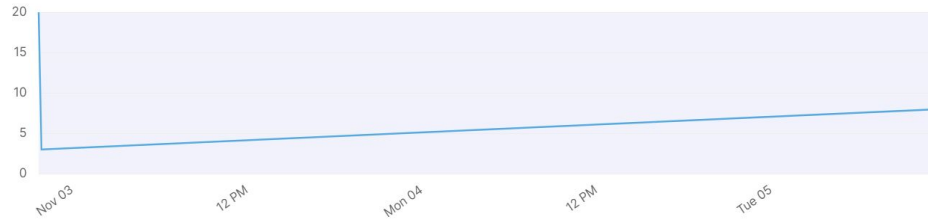
0

A

Activity

Graph type Issues

Issues New Code



Konfiguracja JaCoCo

Dodaj plugin JaCoCo do pliku pom.xml (nie może znajdować się w sekcji pluginManagement), a następnie wykonaj *mvn verify test*

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.7</version>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Raport pokrycia kodu

SonerTest		View as	List	Select files	Navigate	2 files
Coverage 36.4%		See history		New Code: Since November 5, 2024		
				Coverage	Uncovered Lines	Uncovered Conditions
src/main/java/com/example/App.java				0.0%	6	0
src/main/java/com/example/Calculator.java				80.0%	1	80

Raport pokrycia kodu

Coverage 80.0%

```
1 package com.example;
2
3 public class Calculator {
4
5     public int add(int a, int b) { return a + b; }
6
7     public int subtract(int a, int b) { return a - b; }
8
9     public int multiply(int a, int b) { return a * b; }
10
11     public double divide(int a, int b) { return a / b; }
12
13 }
14
```

Uncovered code

Raport pokrycia kodu

Main > org.example

org.example

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 Main		0%		0%	6	6	17	17	2	2	1	1
 Deck		87%		100%	1	7	3	14	1	5	0	1
 Card		94%		80%	2	12	0	16	0	7	0	1
 Rank		100%		n/a	0	1	0	14	0	1	0	1
 Suit		100%		n/a	0	1	0	5	0	1	0	1
Total	101 of 345	70%	10 of 22	54%	9	27	20	66	3	16	1	5

Linki do pobrania

- SonarQube:
<https://www.sonarsource.com/products/sonarqube/downloads/>
- SonarScanner:
<https://docs.sonarsource.com/sonarqube/9.7/analyzing-source-code/scanners/sonarscanner/>
- JaCoCo:
<https://www.eclemma.org/jacoco/>

Bibliografia

- <https://docs.sonarsource.com/sonarqube/latest/user-guide/>
- <https://www.swtestacademy.com/sonarqube-tutorial/>
- <https://mkyong.com/maven/maven-jacoco-code-coverage-example/>
- <https://stackoverflow.com/questions/55716779/generating-a-jacoco-code-coverage-report-with-maven>

Dziękujemy za uwagę!

Berenike Banek, Mateusz Bielówka