SonarQube

Natalia Dybczak i Jagoda Flejmer

Plan prezentacji

01

Podstawowe Pojęcia

02

Funkcje SonarQube

03

Działanie SonarQube

04

Instalacja

05

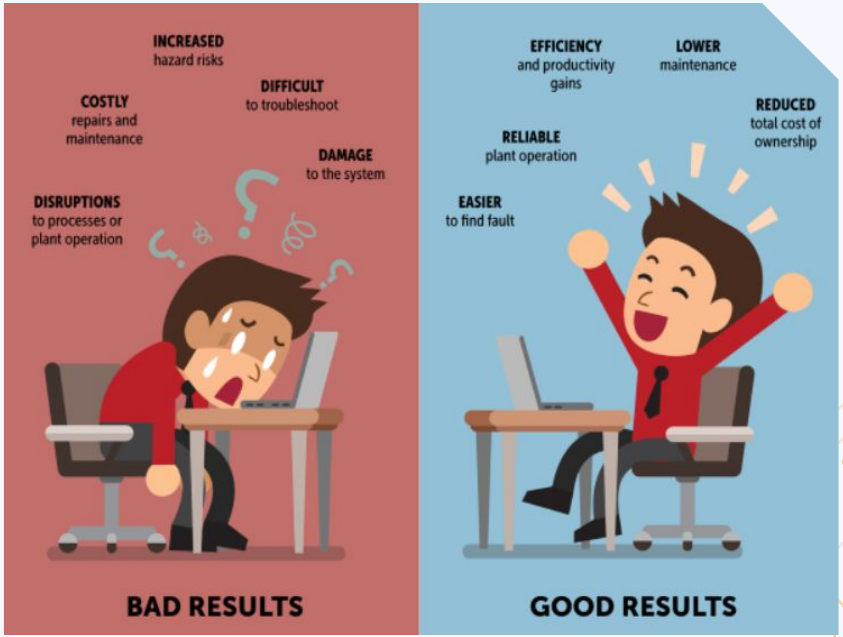
Użytkowanie oraz UI

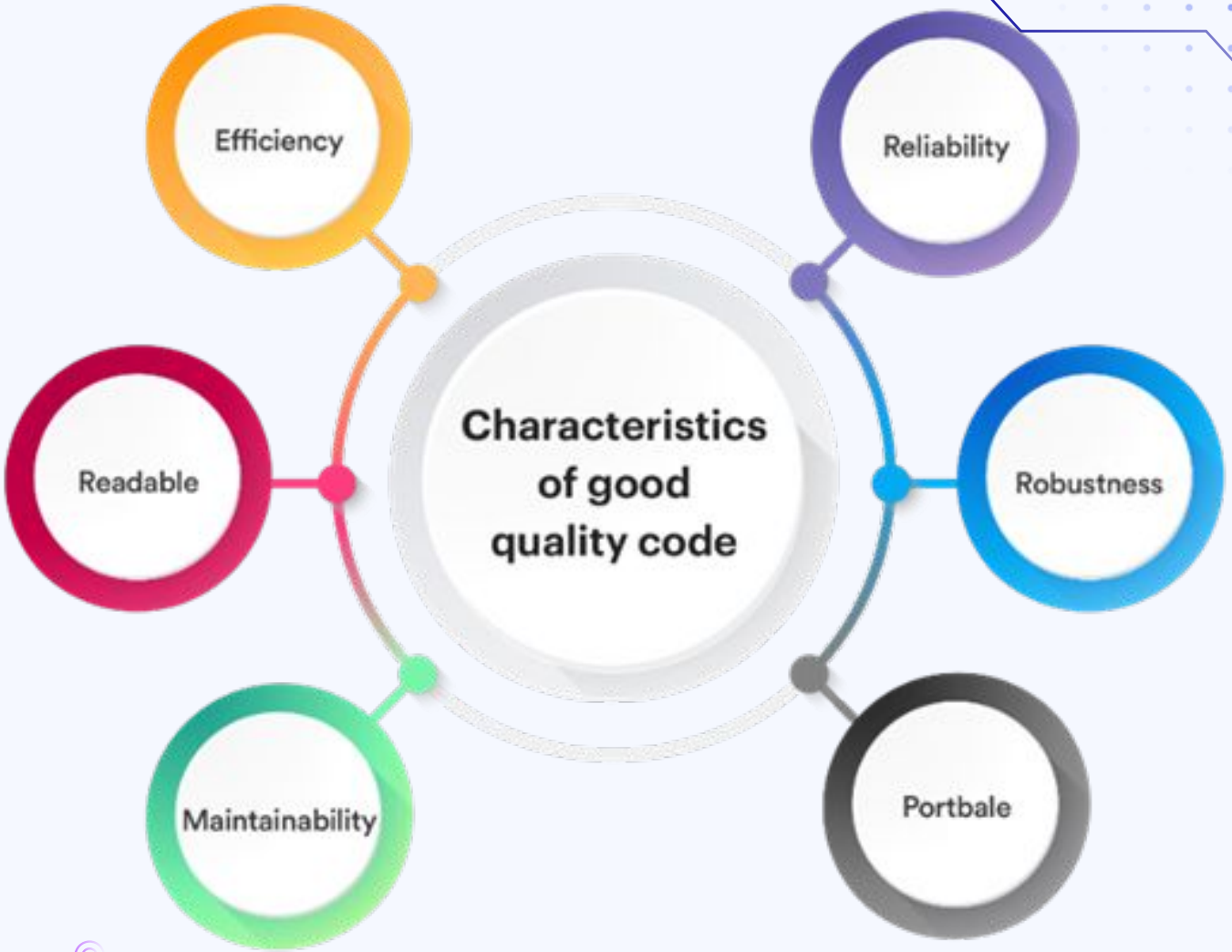
01

Wprowadzenie

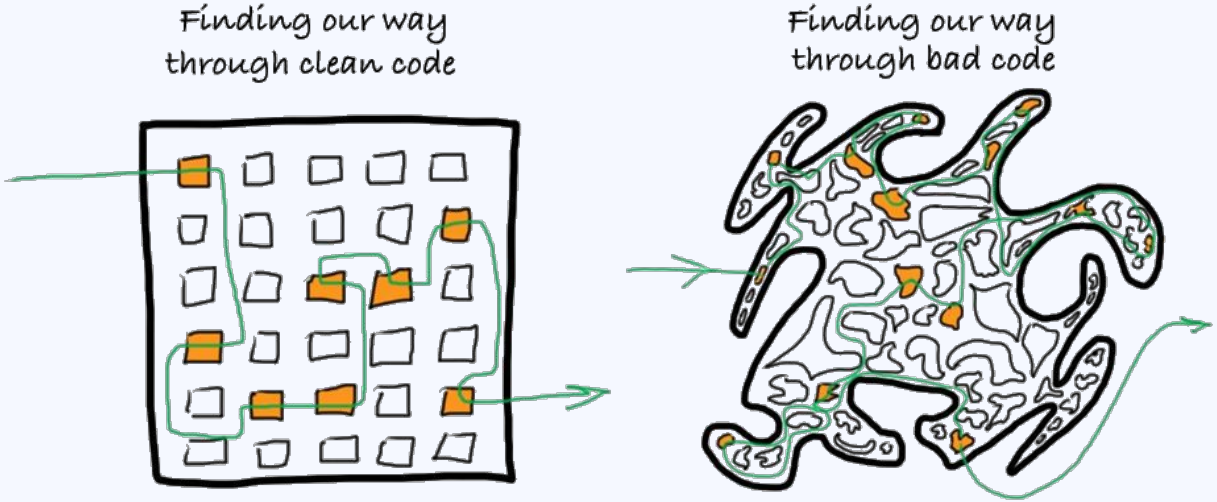
Code quality

Code quality odnosi się do ogólnej oceny efektywności, czytelności, użyteczności i łatwości utrzymania kodu. Określa jak dobrze kod przestrzega ustalonych standardów lub praktyk programistycznych oraz jak dobrze działa zgodnie z zamierzeniami. Wysoka jakość kodu ma kluczowe znaczenie dla długoterminowego sukcesu projektu programistycznego.

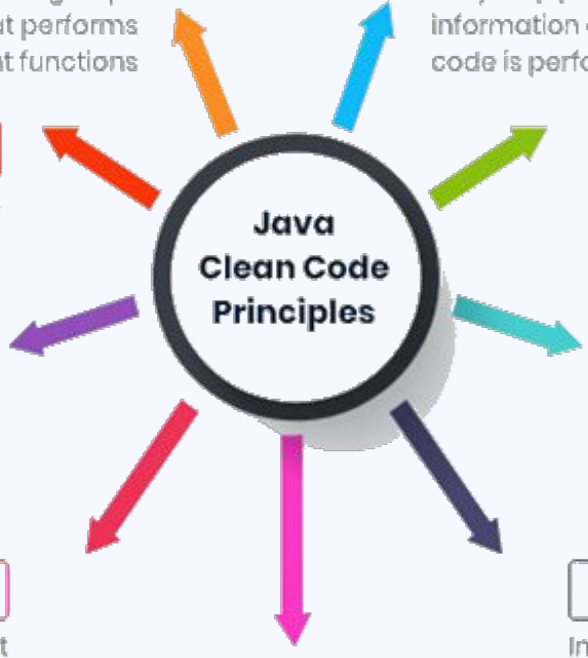



Characteristics of good quality code
Efficiency
Reliability
Readable
Robustness
Maintainability
Portbale

Clean code



Clean Code to standard dla wszelkiego rodzaju kodu, który prowadzi do tworzenia bezpiecznego, niezawodnego i łatwego w utrzymaniu oprogramowania, obejmującego kod źródłowy, testowy, infrastrukturalny, pomocniczy oraz skrypty. Umożliwia łatwe czytanie, zrozumienie i utrzymanie kodu, co przekłada się na poprawę jakości oprogramowania poprzez strukturę i spójność w zakresie wymagań dotyczących wydajności. Dzięki temu czysty kod pozwala uzyskać maksymalną wartość i funkcjonalność oprogramowania.



Correct Code Structure

Divide related code into groups and separate code that performs different functions

Add Code Comments

They help programmers add information about the function the code is performing

Clear & Descriptive Names

Makes the code much easier to understand, find bugs, or change later.

Address Duplicate Code

The repetitive code can be extracted to a separate method or class & then used for all instances of repetitions

Standard Source File Structure

Different types of elements would be present in the source file structure it should be followed across the entire coding

Well Written Logs

Quality logs serve several purposes including simplifying the debugging process

Code Format Consistency

Consistency in code format improves readability of the code & makes it easier to find bugs

Avoid Hardcoding

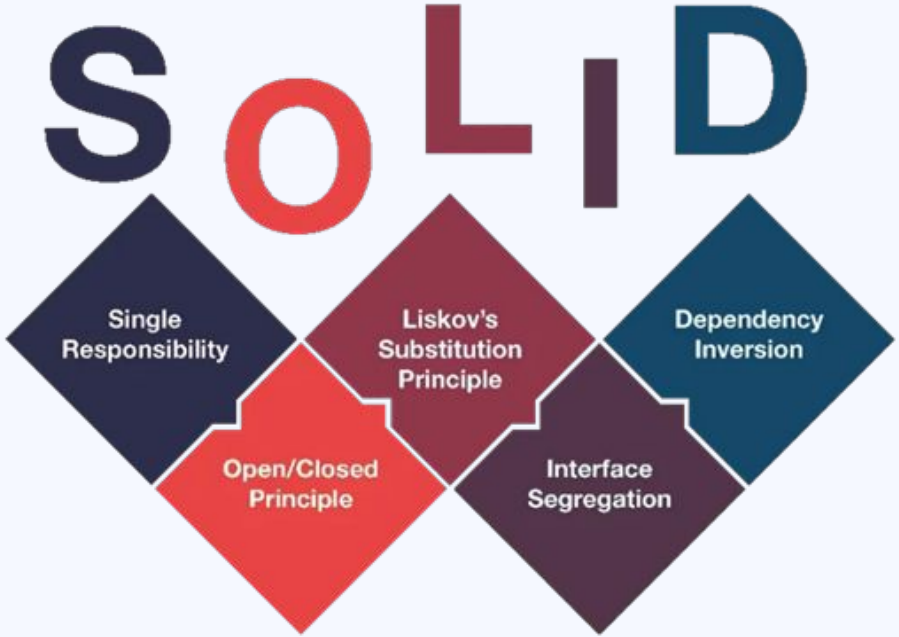
Including data directly into the code can affect the code's performance and make it prone to errors

Limited Parameters in Method

Having too many parameters in a method makes it unnecessarily complex & difficult to interpret

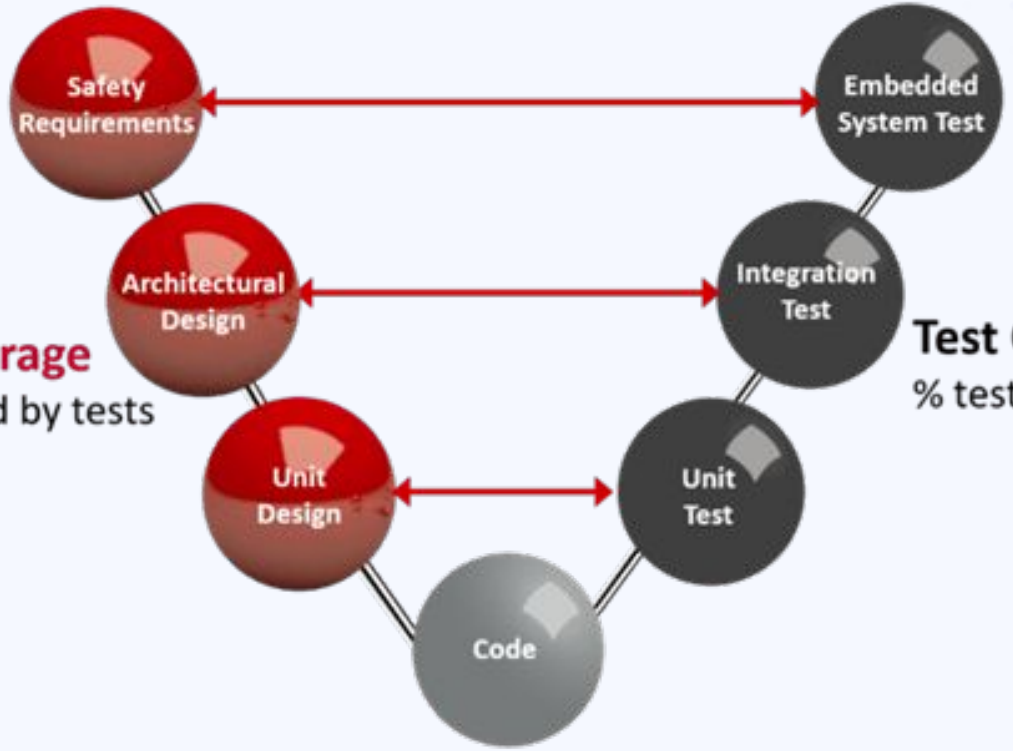
Koncepcja SOLID

Zasady SOLID są kluczowymi wytycznymi w programowaniu obiektowym, których przestrzeganie jest niezbędne dla osiągnięcia czystego kodu.



Pokrycie kodu

Pokrycie kodu to wskaźnik, który mierzy, w jakim stopniu testy jednostkowe obejmują kod źródłowy aplikacji. Wyższy poziom pokrycia kodu oznacza większą pewność, że aplikacja działa zgodnie z oczekiwaniami oraz że jest bardziej odporna na błędy. Wysokie pokrycie kodu świadczy o staranności w testowaniu, co może prowadzić do zwiększenia stabilności i niezawodności oprogramowania.



Requirements Coverage

% Requirements verified by tests

Test Coverage

% tests executed & passing

Code Coverage

% code executed by tests

Static Code Analysis

Statyczna analiza kodu to proces automatycznego skanowania kodu źródłowego, przeprowadzany przez uruchomieniem programu, w celu wykrycia:

- błędów,
- potencjalnych zagrożeń bezpieczeństwa,
- niespójności,
- niezgodności z dobrymi praktykami programistycznymi.

BENEFITS OF USING SONARQUBE FOR STATIC CODE ANALYSIS

01

Detect
errors early

02

Prevent possible
vulnerabilities and
code duplications

03

Improve
code quality

04

Reduce time for fixing
code and increase
productivity

05

Scale the business, as
there are no restrictions
on the number of projects

06

Increase code
consistency according
to established criteria


sonarqube

02
Funkcje SonarQube

SonarQube

SonarQube narzędzie do analizy i oceny jakości kodu, które pozwala na automatyczne sprawdzanie kodu źródłowego pod kątem błędów, podatności oraz innych problemów, mogących wpłynąć na jakość kodu. Dzięki temu, programiści mogą szybko zidentyfikować i naprawić potencjalne problemy, zanim zostaną one wdrożone do produkcji. Oferuje zestaw metryk i wskaźników, które pozwalają na ocenę jakości kodu w oparciu o różne kryteria, takie jak czytelność, złożoność, zgodność ze standardami oraz bezpieczeństwo.

Funkcje SonarQube

- **Analiza kodu** - oferuje szczegółową ocenę jakości kodu oraz identyfikuje obszary, w których zostały złamane ustalone reguły.
- **Pokrycie kodu** – mierzy , w jakim stopniu testy jednostkowe obejmują kod źródłowy, co pozwala ocenić jakość testowania
- **Wykrywanie duplikatów kodu**
- **Skanowanie pod kątem podatności** - automatycznie wykrywa potencjalne błędy i słabości w kodzie, które mogą obniżać bezpieczeństwo aplikacji.
- **Wykrywanie „code smells”** – identyfikuje problemy w kodzie (niekoniecznie będące błędami), które świadczą o jego niskiej jakości i mogą prowadzić do problemów w przyszłości.

- **Zarządzanie jakością kodu** - Umożliwia definiowanie reguł i standardów jakości, zarówno za pomocą profilu „Sonar Way”, jak i własnych standardów w ramach Quality Gate.
- **Analiza bezpieczeństwa aplikacji** – posiada moduł do śledzenia potencjalnych manipulacji danymi przez atakujących, identyfikujący wrażliwe miejsca w kodzie narażone na ataki.
- **Wsparcie dla znanych API** - monitoruje znane API, które mogą być źródłem lub celem potencjalnych ataków.
- **Konfiguracja dla autorskich frameworków** - umożliwia dostosowanie narzędzia do specyficznych frameworków, co pozwala na efektywną sanityzację kodu.
- **Ochrona przed atakami** - pomaga w zabezpieczeniu kodu przed różnymi rodzajami ataków, takimi jak SQL Injection, Code Injection czy Server-Side Request Forgery (SSRF).

03
Działanie SonarQube

SONARQUBE COMPONENTS

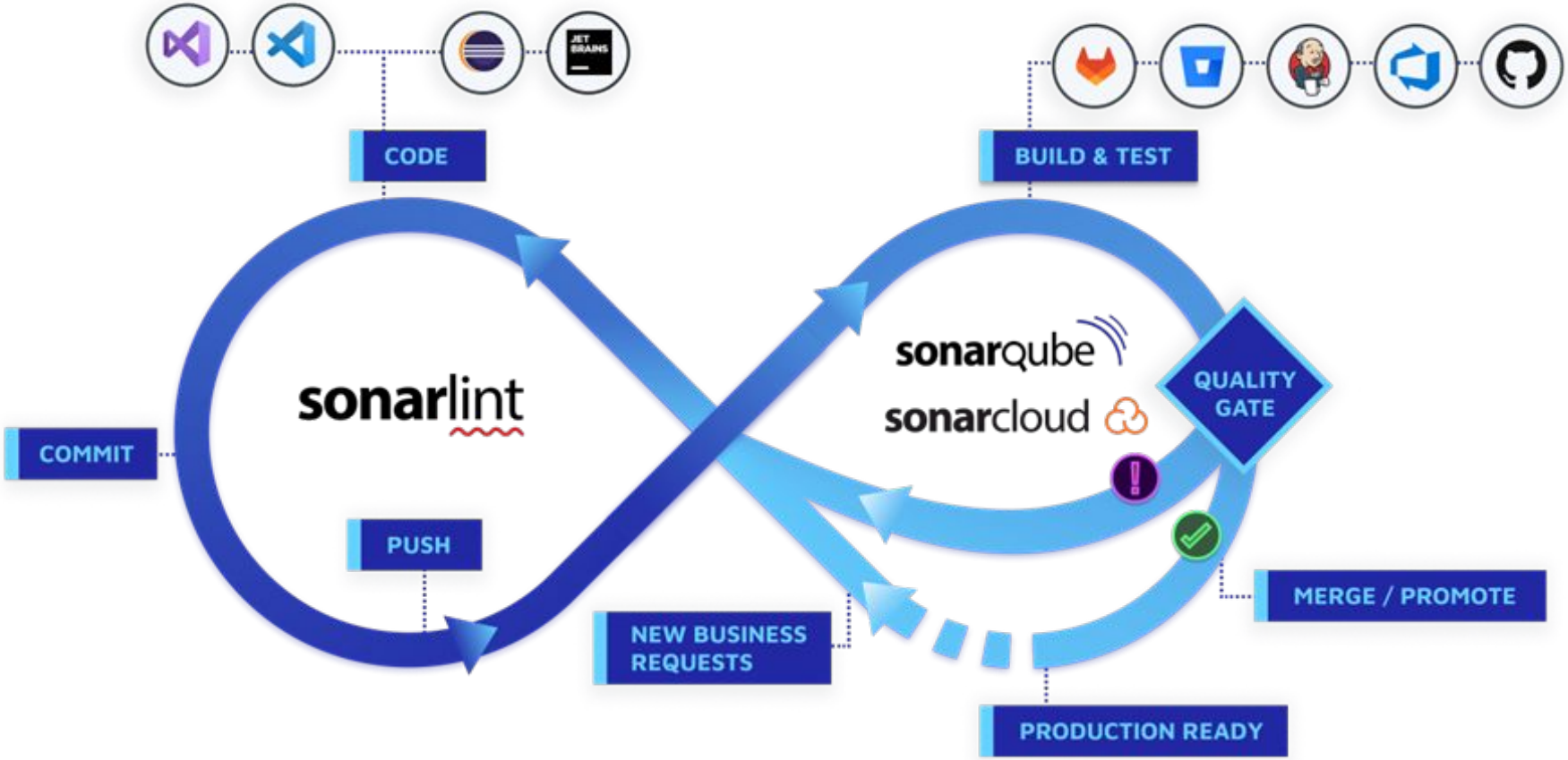
Komponenty SonarQube

Skaner - analizuje kod źródłowy, wykonując statyczną analizę pod kątem różnych aspektów jakości

Baza danych - przechowywane są w niej metryki jakości kodu oraz konfiguracja dotycząca analiz i reguł, co pozwala na śledzenie postępów i wprowadzanie zmian w analizie.

Serwer SonarQube - udostępnia interfejs użytkownika, przetwarza raporty z analizy kodu i zapisuje je do bazy danych

How SonarQube Works!


CODE
BUILD & TEST
sonarlint
sonarqube
sonarcloud
QUALITY GATE
COMMIT
PUSH
NEW BUSINESS REQUESTS
MERGE / PROMOTE
PRODUCTION READY

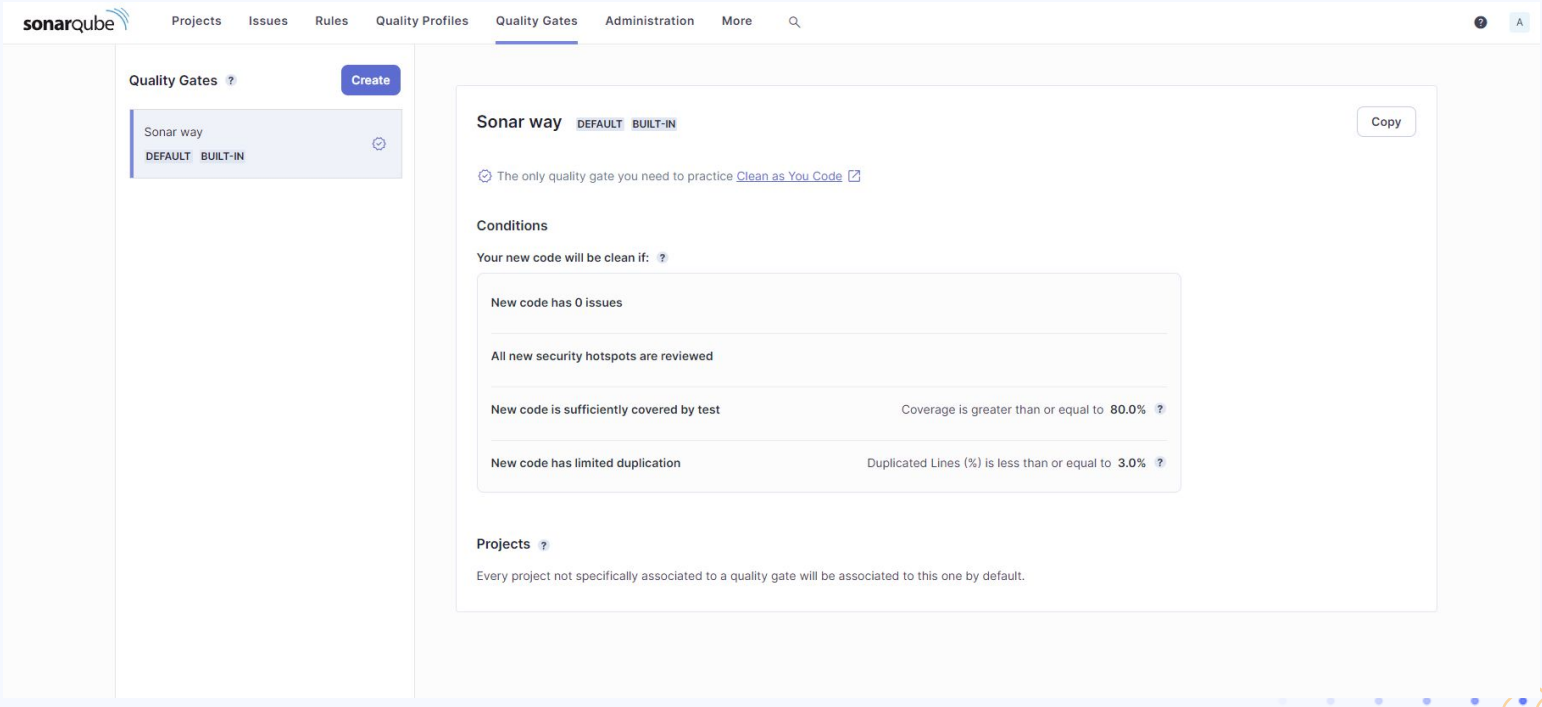
Quality Gate

Quality Gate jest narzędziem determinującym standardy dotyczące jakości przyjęte dla projektu. Jest to zbiór warunków określających gotowość kodu do wypuszczenia na środowisko produkcyjne. Przykładowe warunki definiowane w ramach Quality Gate:

- dopuszczalny minimalny poziom pokrycia testami
- procentowy udział powielonego kodu
- brak wystąpienia nowych błędów na poziomie *Blocker*.

Domyślną konfiguracją, zalecaną dla większości projektów, jest *Sonar way*.

Quality Gate



04

Instalacja

Wymagana Java 17 !!!

Instalacja SonarQube

01 Pobierz SonarQube Community Edition zip file ze strony:
https://www.sonarsource.com/products/sonarqube/downloads/

02 Rozpakuj plik np: C:\sonarQube\sonarqube-10.7.0.96327

03 W pliku conf\sonar.properties dokonaj potrzebnych zmian:

```
# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC user.
# The schema must be created first.
sonar.jdbc.username=admin
sonar.jdbc.password=admin

#    Startup can be long if entropy source is short of entropy. Adding
#    -Djava.security.egd=file:/dev/./urandom is an option to resolve the problem.
#    See https://wiki.apache.org/tomcat/HowTo/FasterStartUp#Entropy_Source
#
sonar.web.javaOpts=-Xmx1024m -Xms512m
# Web context. When set, it must start with forward slash (for example /sonarqube).
# Changing this value and restarting the server can have unexpected consequences for logged in users,
# as they may end up with multiple conflicting browser cookies. If your users experience odd
# behaviors after you changed this value and restarted the server, ask them to clear their browser's
# cookies and log in again.
# The default value is root context (empty value).
#sonar.web.context=
# TCP port for incoming HTTP connections. Default value is 9000.
sonar.web.port=9000
```

Instalacja SonarQube

04

W konsoli uruchom plik: StartSonar.bat z folderu

C:\sonarQube\sonarqube-10.7.0.96327\bin\windows-x86-64

a. C:\sonarQube\sonarqube-10.7.0.96327\bin\linux-x86-64
b. C:\sonarQube\sonarqube-10.7.0.96327\bin\macosx-universal-64

Oczekiwany wynik:

```
2024.11.04 13:09:32 INFO  app[][o.s.a.SchedulerImpl] Process[ce] is up
2024.11.04 13:09:32 INFO  app[][o.s.a.SchedulerImpl] SonarQube is operational
```

Instalacja SonarScanner CLI

01 Pobierz SonarScanner CLI ze strony:
https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/

02 Rozpakuj plik np: C:\sonar-scanner-6.2.1.4610-windows-x64

03 W pliku conf\sonar-scanner.properties dokonaj potrzebnych zmian:

```
#----- SonarQube server URL (default to SonarCloud)
sonar.host.url=https://localhost:9000
```

04 Do zmiennej środowiskowej PATH dodaj lokalizację folderu bin sonarScannera

np.: C:\sonar-scanner-6.2.1.4610-windows-x64\bin

Instalacja SonarScanner CLI

05 Uruchom StartSonar.bat

06 Zaloguj się do serwera ustalonego w sonar.host.url np. : http://localhost:9000 używając domyślnego loginu i hasła admin (jeżeli nie działa: http://127.0.0.1:9000)

07 Ustaw nowe hasło logowania

05

Użytkowanie i UI

Z projektem Maven

Integracja z Mavenem

01 Do pliku pom.xml dodaj plugin:

```
<plugin>
    <groupId>org.sonarsource.scanner.maven</groupId>
    <artifactId>sonar-maven-plugin</artifactId>
    <version>4.0.0.4121</version> <!-- Aktualna wersja -->
</plugin>
```

Integracja z Mavenem

02 Dodaj również repozytorium:

```
<repositories>
      <repository>
            <id>sonarsource-repo</id>
            <url>https://repo.maven.apache.org/maven2 </url>
      </repository>
</repositories>
```

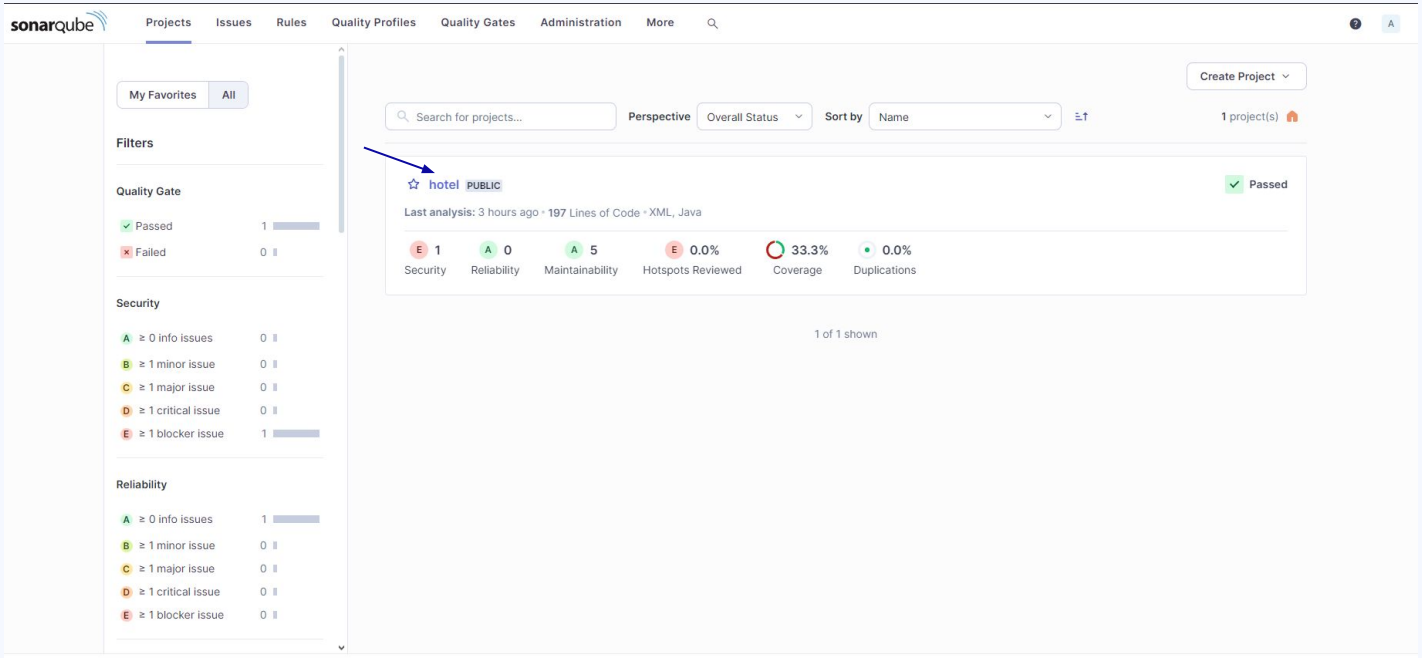
Integracja z Mavenem

03 Z konsoli wbudowanej intellij uruchom plik StartSonar.bat z folderu C:\sonarQube\sonarqube-10.7.0.96327\bin\windows-x86-64(lub odpowiadającemu sys. op)

04 Użyj komendy mvn clean install sonar:sonar

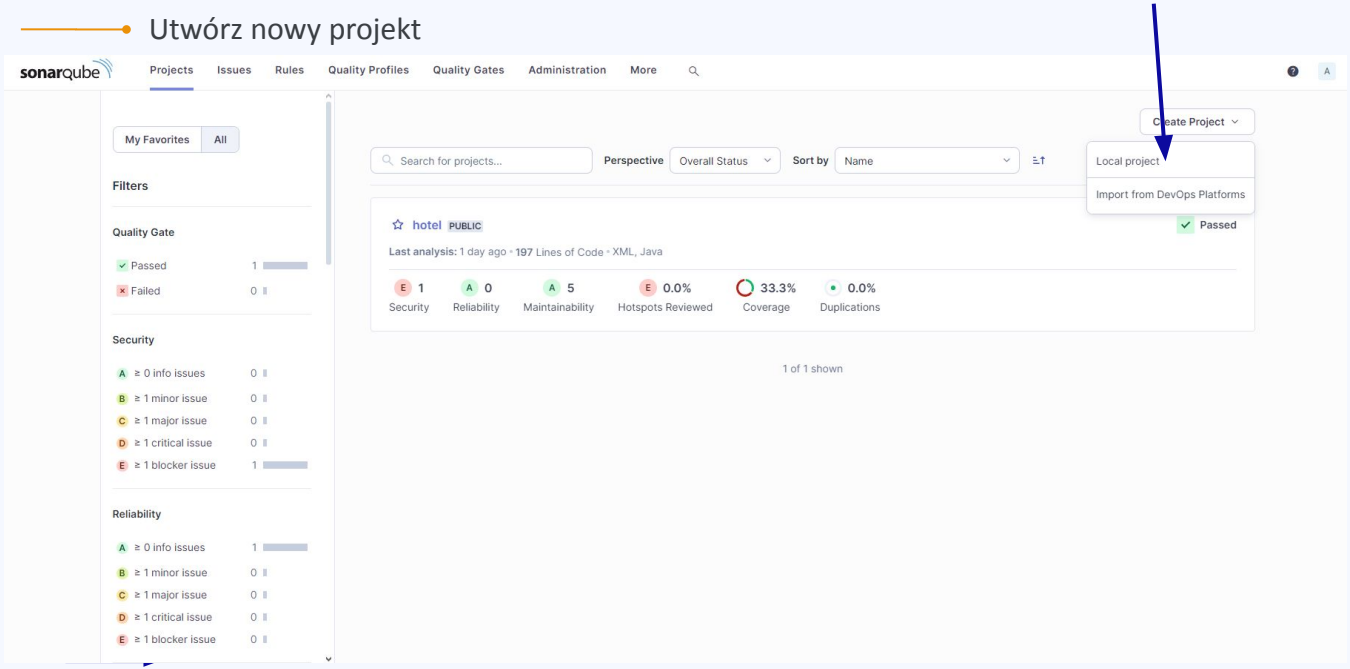
05 W przeglądarce otwórz adres ustawiony jako local.host

Utworzy się projekt



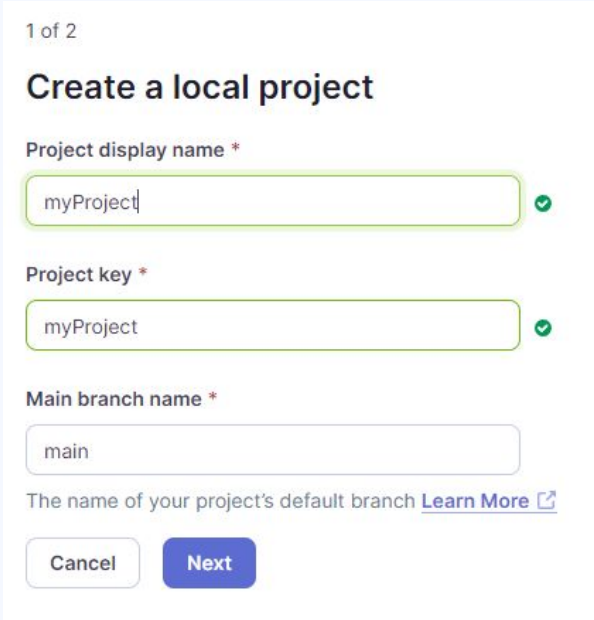
Tworzenie projektu lokalnie

01 Utwórz nowy projekt



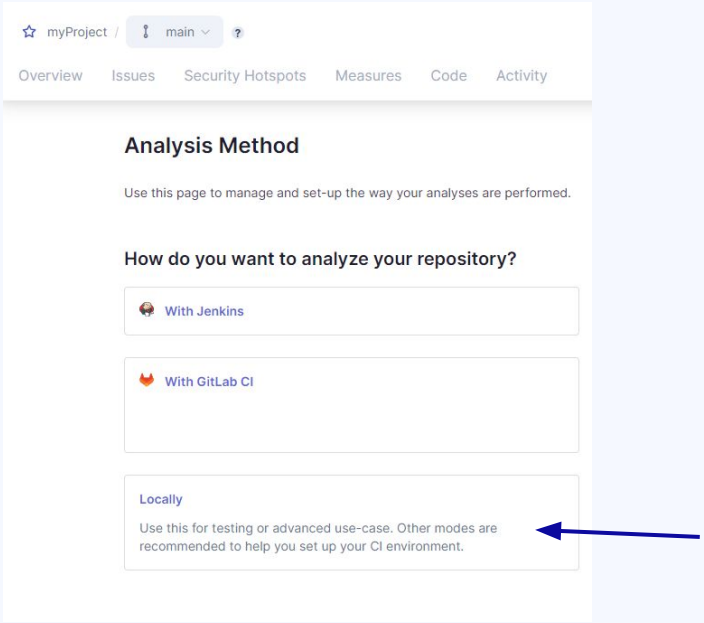
Tworzenie projektu lokalnie

01 Utwórz nowy projekt



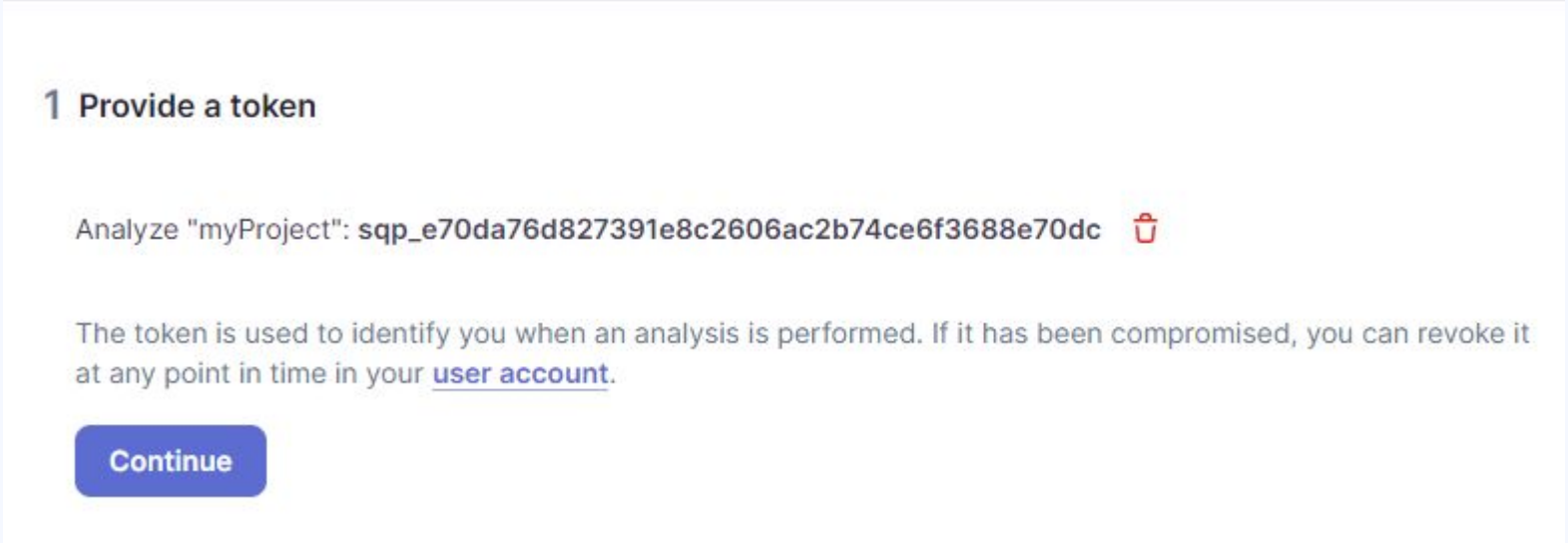
Tworzenie projektu lokalnie

02 Wybierz lokalną analizę



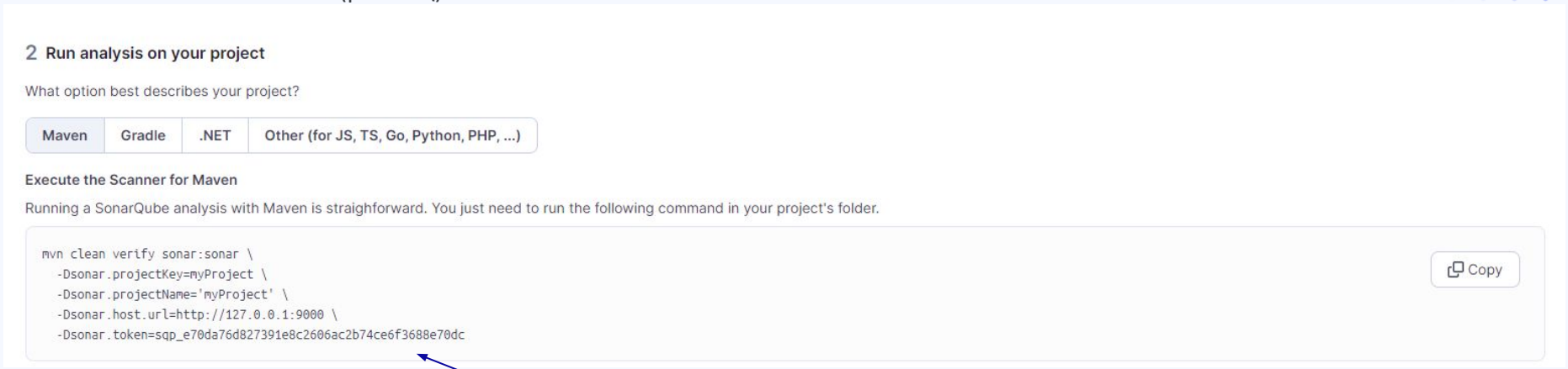
Tworzenie projektu lokalnie

03 Utwórz token

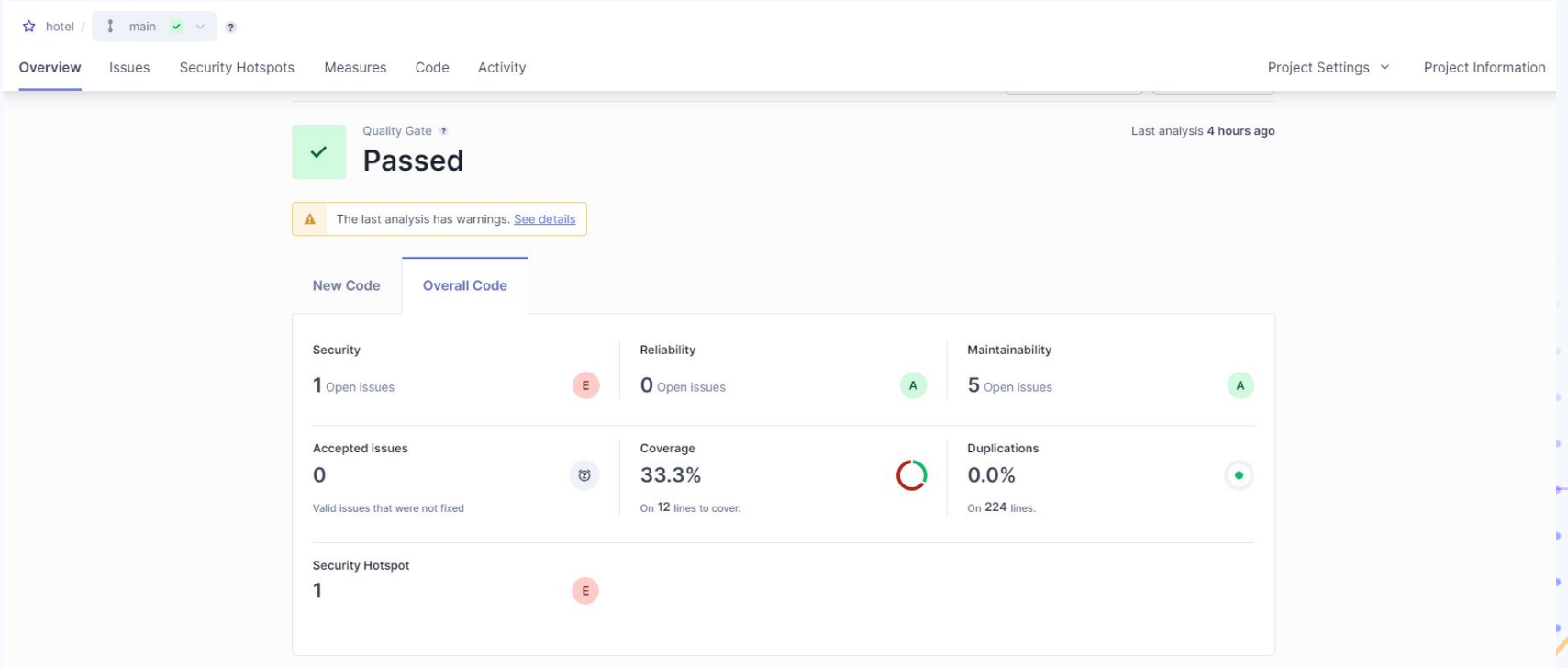


Tworzenie projektu lokalnie

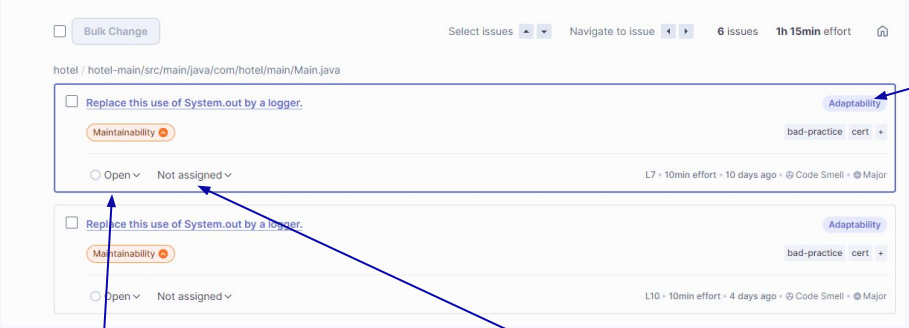
04 Przeprowadź analizę projektu, uruchamiając wygenerowaną komendę w konsoli projektu (pomiń \)



Project Overview



Issues



Clean Code

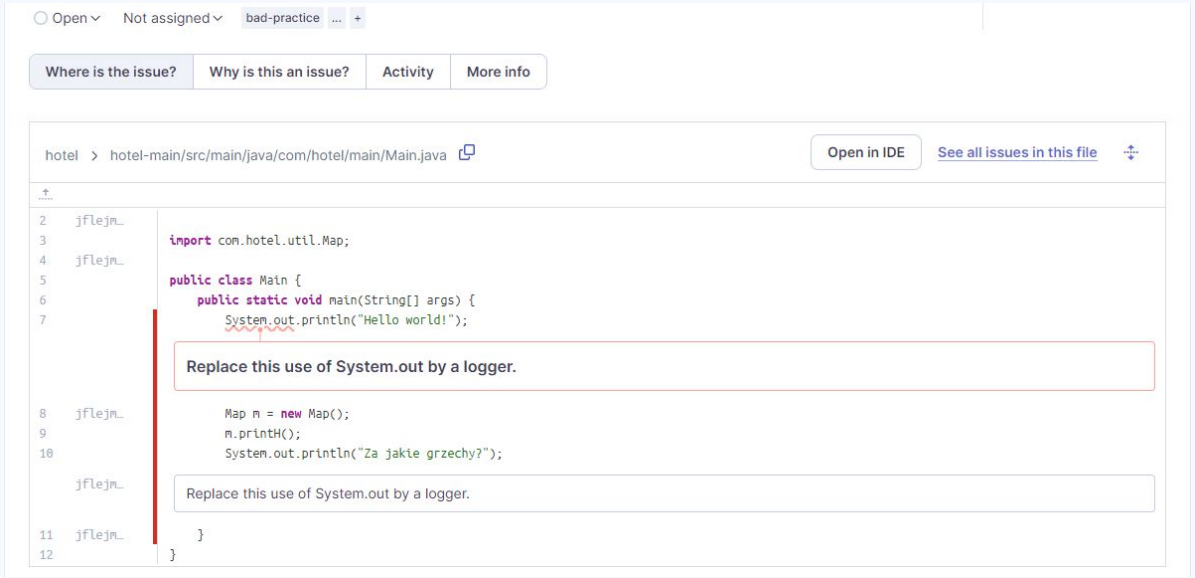
Przypisanie osoby

Aktualność problemu

Issues

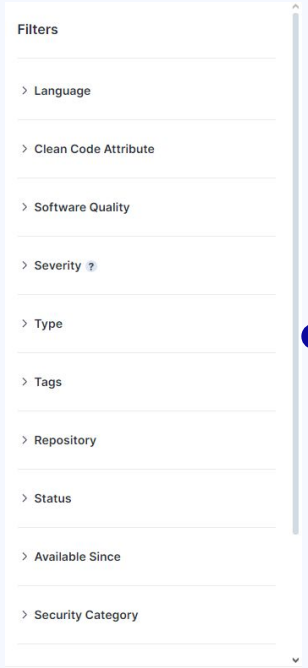
Poziom istotności

Typ



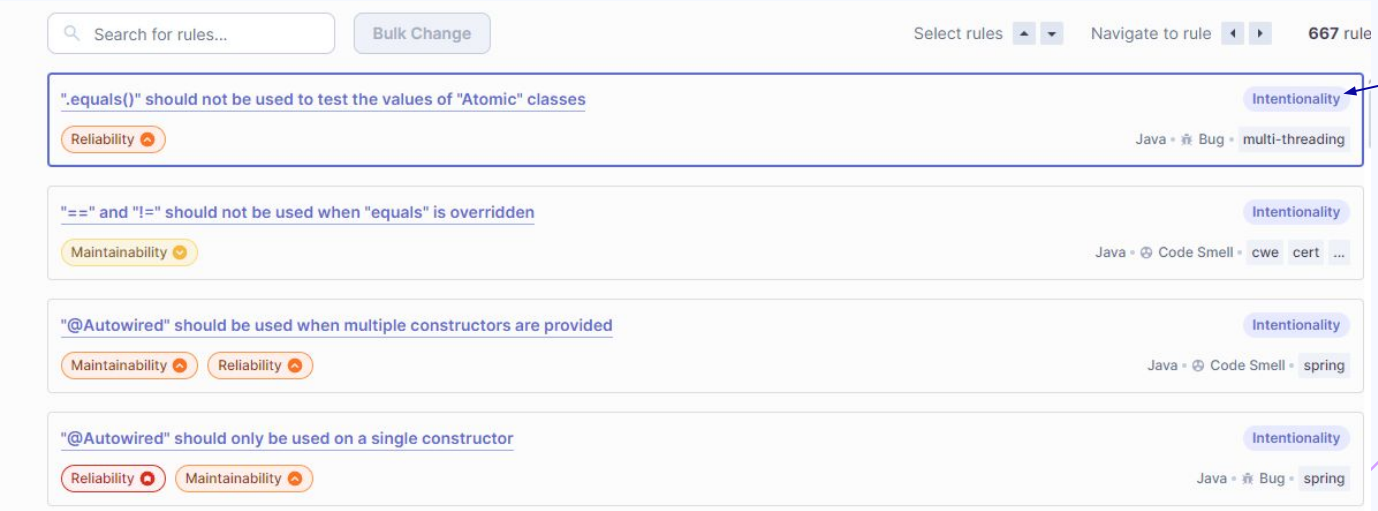
Rules

Reguły, które muszą być spełnione podczas analizy kodu. Możemy je dopasowywać do potrzeb projektu.



Możliwość filtrowania

Rules

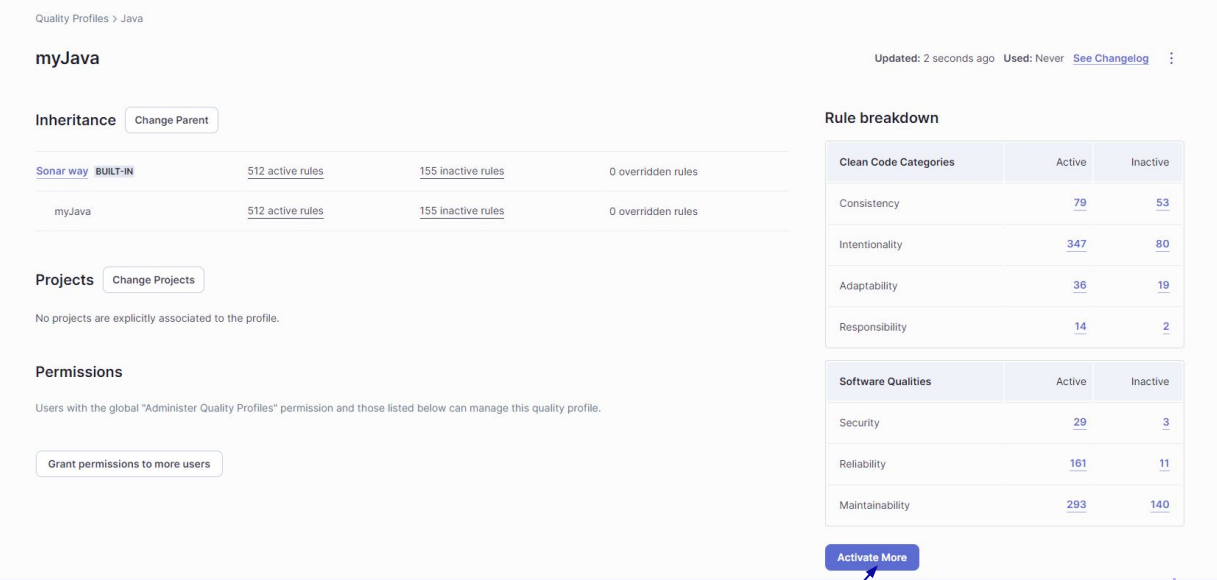


Clean code

Software qualities i istotność

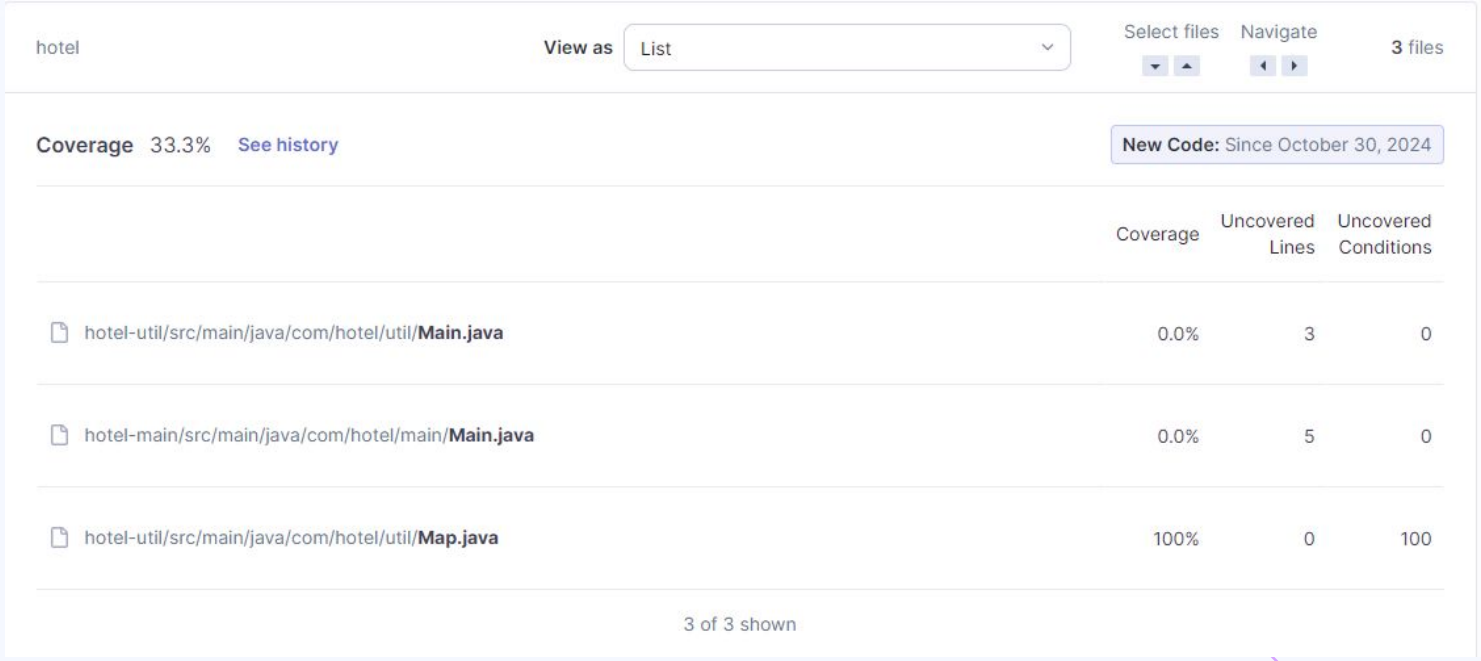
Dodawanie reguł

01 Reguły dodajemy do wybranego Quality Profile



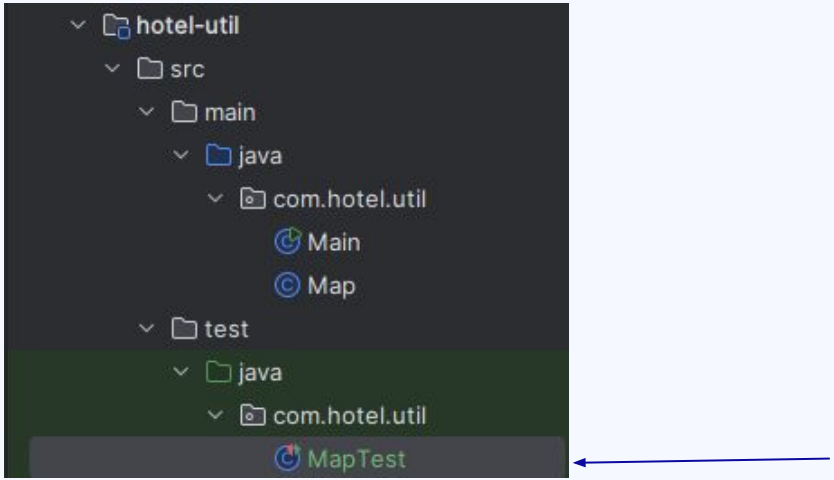
Wybieramy nową regułę

Code Coverage



Code Coverage

01 Utwórz testy do odpowiedniej klasy np.:



Code Coverage - Surefire plugin

02

Do pliku pom.xml modułu, w którym znajduje się testowana klasa dodaj pluginy (surefire):

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.2.5</version>
    <configuration>
        <includes>
            <include>**/*Test.java</include>
        </includes>
    </configuration>
    </plugin>
```

Code Coverage - JaCoCo

02

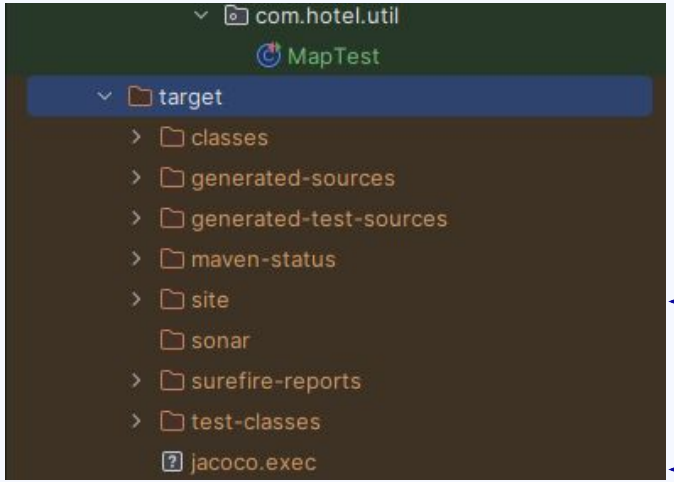
Do pliku pom.xml modułu, w którym znajduje się testowana klasa dodaj pluginy (JaCoCo):

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

Code Coverage

03 Uruchom testy np.: mvn clean test

04 W folderze target powinien utworzyć się plik jacoco:



W katalogu site/jacoco zostaną wygenerowane raporty w formacie html

Dziękujemy za uwagę