

Qualidade de Software e SonarQube

Treinamento: Angular, Node e Java – Capgemini

Instrutor: Ivan J. Borchardt

©2021

Proway

Qualidade de Software

- ❖ A partir do momento que o software ficou cada vez mais integrado às atividades do dia a dia das empresas, a necessidade de produtos com qualidade se tornou essencial e um diferencial no mercado. A qualidade é considerada, hoje, um requisito muito importante em todas as áreas, pois todos querem produtos e serviços de excelência (GUERRA; COLOMBO, 2000).

Qualidade de Software

QUALIDADE DE SOFTWARE:

O que é qualidade de software?

- RUIM
- BOM
- REGULAR
- EXCELENTE



Qualidade de Software

Para iniciar, veremos a definição dada ao termo “qualidade”, pois ele pode variar de acordo com a abordagem utilizada. A seguir, algumas definições encontradas na literatura sobre o termo.

Qualidade de Software

Qualidade:

“Conformidade com as especificações”: definição abordada por Philip B. Crosby e que sugere que a qualidade deve ser verificada desde o início do desenvolvimento, para tentar evitar defeitos e diminuir o retrabalho (CROSBY, 1990).

Qualidade de Software

Qualidade:

“Adequação ao uso”: definição sugerida por Joseph M. Juran e que significa que as expectativas do cliente devem ser atendidas (JURAN; DEFEO, 2015).

Qualidade de Software

Qualidade:

“Qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas”: definição segundo a Norma Brasileira de Referência NBR ISO 8402 (ABNT, 1994), a qual diz que as necessidades explícitas são expressas na definição de requisitos que definem as condições em que o produto deve ser utilizado, seus objetivos, suas funções e qual será o desempenho esperado desse produto. As necessidades implícitas são necessárias para o usuário no manuseio do produto, no seu dia a dia, e não expressas em documentos. A entidade é o produto, o qual pode ser um bem ou um serviço (ABNT, 1994, p. 1).

Qualidade de Software

Como reconhecer se o produto ou serviço tem qualidade? A resposta a esta pergunta não é tão simples quanto se imagina, pois sabe-se o que é qualidade ao vê-la e, mesmo assim, é difícil de ser definida. De acordo com Pressman e Maxim (2016), temos cinco maneiras de identificar e defini-la:

Qualidade de Software

Visão do usuário	<p>Um produto atende às metas específicas de um usuário final, aquele que melhor atender às preferências do usuário.</p>
Visão do fabricante	Conformidade com as especificações predefinidas no início do projeto. Se o produto atende às especificações originais do produto.
Visão do produto	Sugere que a qualidade pode ser ligada às funções e recursos de um produto. A qualidade poderá ser medida por meio de alguns atributos do produto. Mais qualidade - mais atributos - custos mais elevados.
Visão baseada em valor	Mede a qualidade tomando como base o quanto um cliente estaria disposto a pagar por um produto.
Transcendental	Sustenta que qualidade é algo que se reconhece imediatamente, mas não se consegue definir explicitamente. A qualidade não pode ser medida de maneira precisa, sendo reconhecida, somente, por meio do contato que o cliente terá com o produto.

Qualidade de Software

Quando consideramos a qualidade do ponto de vista multidimensional, o qual, segundo Pressman e Maxim (2016), começa com uma avaliação da conformidade e termina com uma visão transcendental (estática), temos algumas dimensões que precisamos considerar. Elas são chamadas de Dimensões de Qualidade de Garvin (GARVIN, 1987 apud PRESSMAN; MAXIM, 2016).

Qualidade de Software

Qualidade do desempenho	O software fornece todo o conteúdo, as funções e os recursos que são especificados como parte do modelo de requisitos, de forma a gerar valor ao usuário final?
Qualidade dos recursos	O software fornece recursos que surpreendem e encantam usuários finais que os utilizam pela primeira vez?
Confiabilidade	O software fornece todos os recursos e as capacidades sem falhas? Está disponível quando necessário? Fornece funcionalidade sem a ocorrência de erros?
Conformidade	O software está de acordo com os padrões de software locais e externos relacionados com a aplicação? Segue as convenções de projeto e codificação de fato? Por exemplo, a interface com o usuário está de acordo com as regras de projeto aceitas para a seleção de menus ou a entrada de dados?
Durabilidade	O software pode ser mantido (modificado) ou corrigido (depurado) sem a geração involuntária de efeitos colaterais indesejados? As mudanças farão com que a taxa de erros ou a confiabilidade diminuam com o passar do tempo?

Qualidade de Software

Facilidade de manutenção	<p>O software pode ser mantido (modificado) ou corrigido (depurado) em um período de tempo aceitável e curto? O pessoal de suporte pode obter todas as informações necessárias para realizar alterações ou corrigir defeitos?</p>
Estética	<p>Não há dúvida nenhuma de que cada um de nós tem uma visão diferente e muito subjetiva do que é estética. Mesmo assim, a maioria de nós concordaria que uma entidade estética tem certa elegância, um fluir único e uma “presença” que são difíceis de quantificar, mas que, não obstante, são evidentes. Um software estético possui estas características.</p>
Percepção	<p>Em algumas situações, temos alguns preconceitos que influenciarão nossa percepção de qualidade. Por exemplo, se for apresentado um produto de software construído por um fornecedor que, no passado, havia produzido software de má qualidade, ficaremos com nossa percepção de qualidade do produto de software influenciada negativamente. Similarmente, se um fornecedor tem uma excelente reputação, talvez percebamos qualidade, mesmo quando ela realmente não existe.</p>

Qualidade de Software

Fatores de Qualidade de Software (Pressman e Maxim (2016, p. 361))

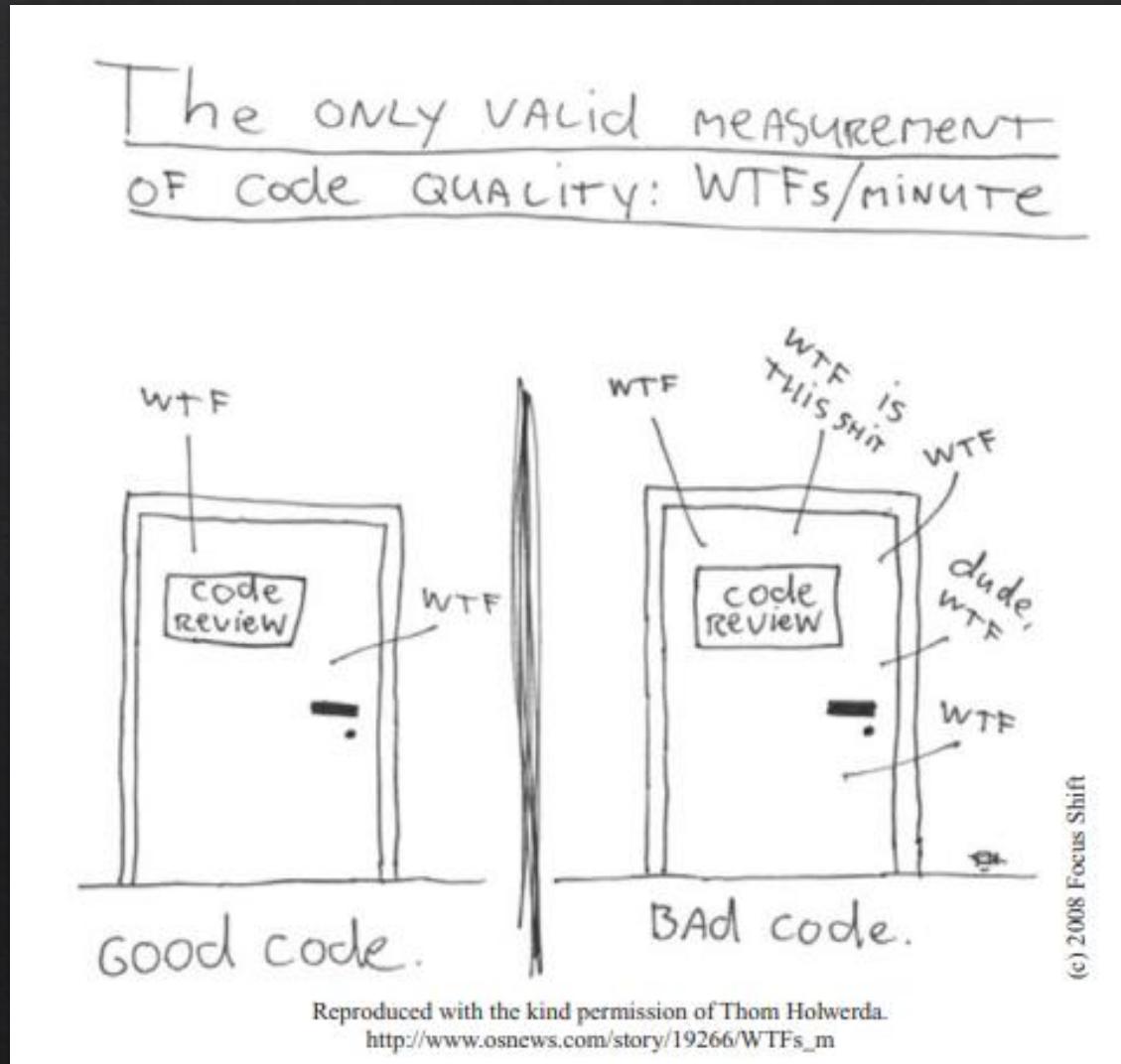
Fatores	Descrição
Correção	O quanto um programa satisfaz à sua especificação e atende aos objetivos da missão do cliente.
Confiabilidade	O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.
Eficiência	A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função.
Integridade	O quanto o acesso ao software ou a dados por pessoas não autorizadas pode ser controlado.
Usabilidade	Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa.
Facilidade de manutenção	Esforço necessário para localizar e corrigir um erro em um programa.

Qualidade de Software

Fatores de Qualidade de Software (Pressman e Maxim (2016, p. 361))

Fatores	Descrição
Flexibilidade	Esforço necessário para modificar um programa em operação.
Testabilidade	Esforço necessário para testar um programa de modo a garantir que ele desempenhe a função destinada.
Portabilidade	Esforço necessário para transferir o programa de um ambiente de hardware e/ou software para outro.
Reusabilidade	O quanto um programa (ou partes de um programa) pode ser reutilizado em outras aplicações – relacionado com o empacotamento e o escopo das funções que o programa executa.
Interoperabilidade	Esforço necessário para integrar um sistema a outro.

Clean Code



Clean Code

Clean Code, ou Código Limpo, é uma filosofia de desenvolvimento de softwares que consiste em aplicar **técnicas simples que facilitam a escrita e a leitura de um código**. Tornando-o, assim, de fácil compreensão.

Clean Code

- Livro “[Clean Code: A Handbook of Agile Software Craftsmanship](#)“, 2008 - Robert Cecil Martin (Uncle Bob).
- Uncle Bob é um dos profissionais por trás do [Manifesto Ágil](#), lançado em 2001.

Clean Code

- O gargalo principal no desenvolvimento de software está na manutenção!
- A proporção média de leitura e escrita de códigos fonte é de 10 para 1...
- Um sistema nunca está totalmente finalizado.

Clean Code

As 7 principais regras do Clean Code:

1. Nomes são muito importantes

- ◊ Ele deve ser preciso e passar logo de cara sua ideia central. Ou seja, deve ir direto ao ponto;
- ◊ Não se deve ter medo de nomes grandes. Se a sua função ou parâmetro precisa de um nome extenso para demonstrar o que realmente representa, é o que deve ser feito.

2. Regra do escoteiro

- ◊ Há um princípio do escotismo que diz que, uma vez que você sai da área em que está acampando, você deve deixá-la mais limpa do que quando a encontrou. Trazendo a regra para o mundo da programação, a regra significa **deixar o código mais limpo do que estava antes de mexer nele**.

Clean Code

As 7 principais regras do Clean Code:

3. Seja o verdadeiro autor do código

- ❖ O ser humano é acostumado a pensar de forma narrativa , portanto, o código funciona da mesma forma. Logo, ele é uma história e, como os programadores são seus autores, **precisam se preocupar na maneira com que ela será contada.**
- ❖ Em resumo, para estruturar um código limpo, é necessário criar funções simples, claras e pequenas. Existem 2 regras para criar a narrativa via código:
 - ❖ As funções precisam ser pequenas;
 - ❖ Elas têm de ser ainda menores.

Clean Code

As 7 principais regras do Clean Code:

4. DRY (Don't Repeat Yourself)

- ◊ O DRY diz que cada pedaço do conhecimento de um sistema deve ter uma representação única e **ser totalmente livre de ambiguidades**. Em outras palavras, define que não pode existir duas partes do programa que desempenhem a mesma função.

Clean Code

As 7 principais regras do Clean Code:

5. Comente apenas o necessário

- ◊ Esse princípio afirma que comentários podem ser feitos, porém, se forem realmente necessários. Segundo Uncle Bob, os comentários mentem. E isso tem uma explicação lógica.
- ◊ O que ocorre é que, **enquanto os códigos são constantemente modificados, os comentários não**. Eles são esquecidos e, portanto, deixam de retratar a funcionalidade real dos códigos.
- ◊ Logo, se for para comentar, que seja somente o necessário e que seja revisado juntamente com o código que o acompanha.

Clean Code

As 7 principais regras do Clean Code:

6. Tratamento de erros

- ◊ Tem uma frase do autor Michael Feathers, muito conhecido na área de desenvolvimento, que diz que as coisas podem dar errado, mas, quando isso ocorre, os programadores são os responsáveis por garantir que o código continuará fazendo o que precisa.
- ◊ Ou seja: saber tratar as exceções de forma correta é um grande e importante passo para um programador em desenvolvimento.

Clean Code

As 7 principais regras do Clean Code:

7. Testes limpos

- ◊ Um código só é considerado limpo após ser validado através de testes – que também devem ser limpos.

Por isso, ele deve seguir algumas regras, como:

- ◊ **Fast:** O teste deve ser rápido, permitindo que seja realizado várias vezes e a todo momento;
- ◊ **Independent:** Ele deve ser independente, a fim de evitar que cause efeito cascata quando da ocorrência de uma falha – o que dificulta a análise dos problemas;
- ◊ **Repeatable:** Deve permitir a repetição do teste diversas vezes e em ambientes diferentes;
- ◊ **Self-Validation:** Os testes bem escritos retornam com as respostas true ou false, justamente para que a falha não seja subjetiva;
- ◊ **Timely:** Os testes de unidade devem ser escritos um pouco antes do código de produção que o faz passar. Aplicações devem ser projetadas para facilitar a testabilidade.



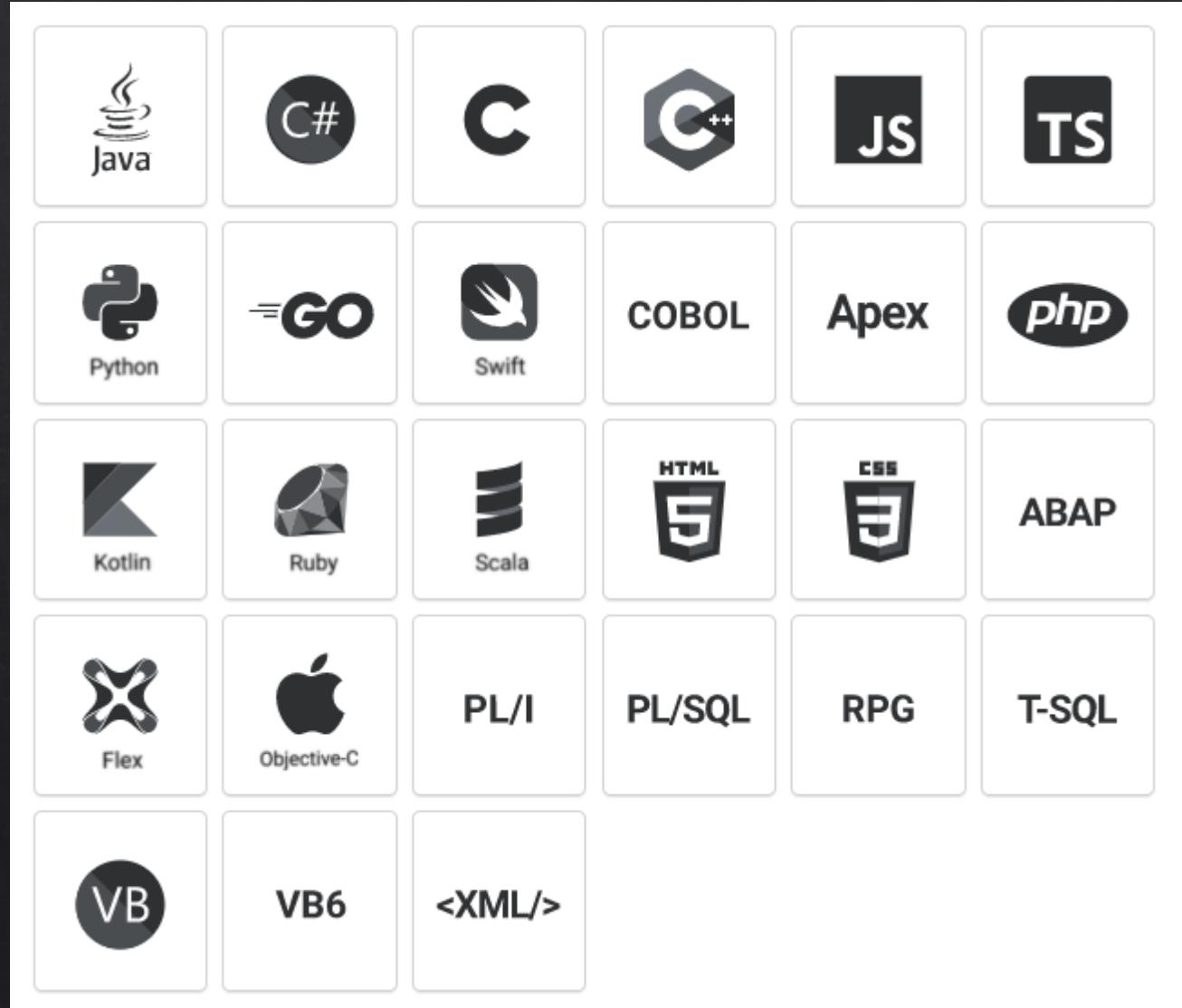
<https://www.sonarqube.org/>



O SonarQube é uma plataforma de código aberto desenvolvida pela SonarSource para **inspeção contínua** da qualidade do código, para executar revisões automáticas com análise estática do código para detectar bugs, odores de código e vulnerabilidades de segurança em mais de 20 linguagens de programação.

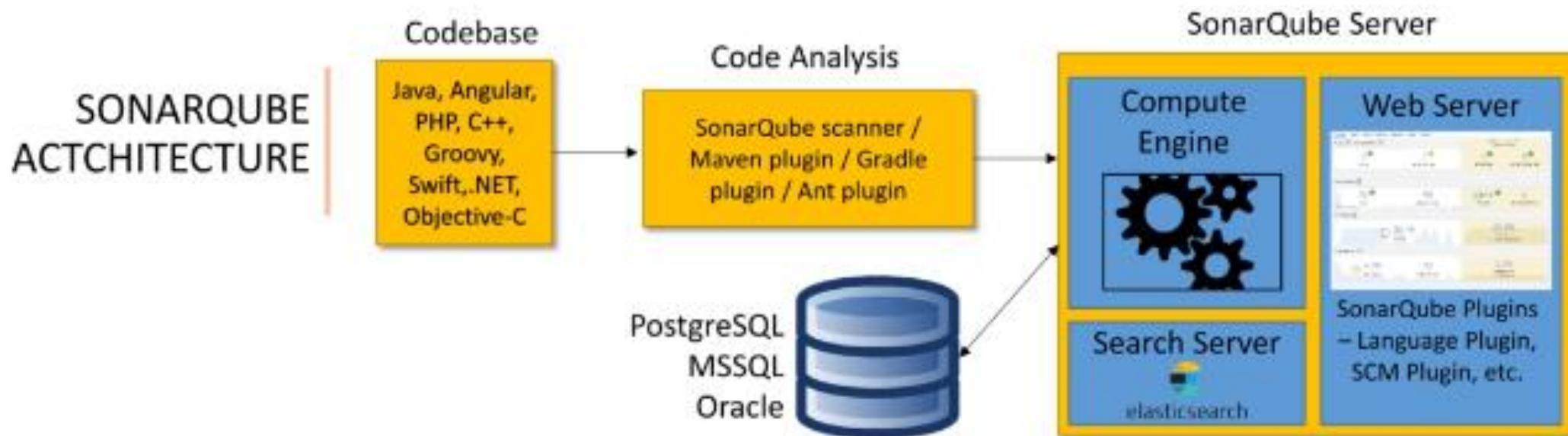


27 Linguagens de Programação...





SONARQUBE ARCHITECTURE OVERVIEW

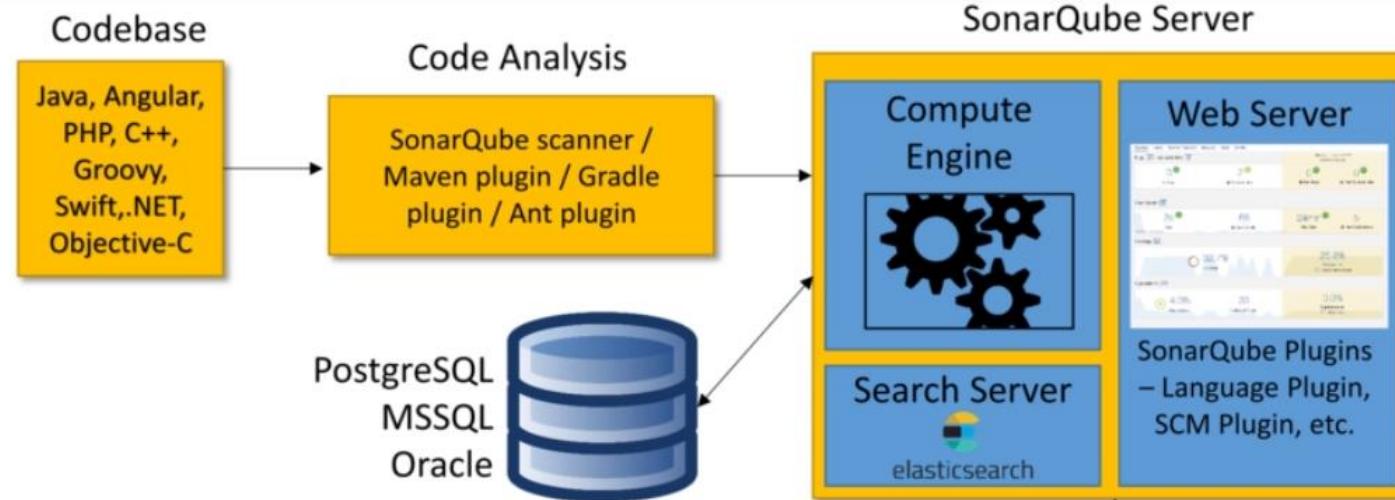




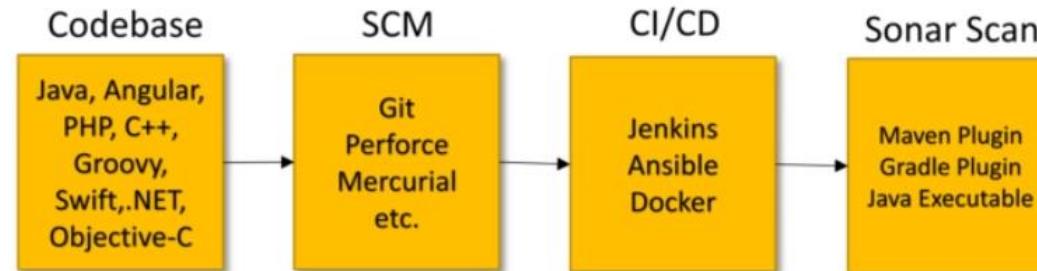
SonarQube Architecture overview

SONARQUBE ARCHITECTURE OVERVIEW

SONARQUBE ACTCHITECTURE



SONAR SCAN WORKFLOW





Instalação

<https://www.sonarqube.org/>

A screenshot of the SonarQube website homepage. The main headline reads "Open source roots, Editions for all use-cases". Below it, a mission statement says "Our mission is to empower developers first, and grow an open community around code quality and code security." A large red arrow points down to the "Community EDITION" section. This section is labeled "FREE & OPEN SOURCE" and describes the "Community EDITION" as "The starting point for adopting code quality in your CI/CD". It features a "Get started now" button. To the right, there are three other editions: "Developer EDITION", "Enterprise EDITION", and "Data Center EDITION", each with a brief description and a "Learn more" link.

**Open source roots,
Editions for all use-cases**

Our mission is to empower developers first, and grow an open community around code quality and code security.

FREE & OPEN SOURCE

Community EDITION

The starting point for adopting code quality in your CI/CD

Developer EDITION

Maximum Application Security; maximum value across branches & PRs

Enterprise EDITION

Manage your Application Portfolio; enable Code Quality & Code Security at an Enterprise level

Data Center EDITION

High Availability, for global deployments

[Get started now](#)

[Learn more →](#)

[Learn more →](#)

[Learn more ↗](#)



Instalação

<https://www.sonarqube.org/>

Download SonarQube
The leading product for Code Quality and Security
HELPING DEVS SINCE 2008

Version: 8.9 LTS | Release: May 2021 | [Getting Started](#) | [Release Notes](#) | [Upgrade Notes](#) | Available From DockerHub

Community
EDITION

The starting point for adopting code quality in your CI/CD
FREE & OPEN SOURCE

[Download for free](#)

All the following features:

- ✓ Static code analysis for 15 languages
Java, JavaScript, C#, TypeScript, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
- ✓ Detect Bugs & Vulnerabilities
- ✓ Review Security Hotspots
- ✓ Track Code Smells & fix your Technical Debt
- ✓ Code Quality Metrics & History
- ✓ CI/CD integration
- ✓ Extensible, with 50+ community plugins

Developer
EDITION

Maximum Application Security
Maximum value across branches & PRs

[Download](#)

Community Edition plus:

- ✓ C, C++, Obj-C, Swift, ABAP, T-SQL, PL/SQL support
- ✓ Detection of Injection Flaws in Java, C#, PHP, Python, Javascript, Typescript
- ✓ Analysis of feature and maintenance branches
- ✓ Pull Request decoration for:
 - GitHub
 - Bitbucket
 - Azure DevOps
 - GitLab

Enterprise
EDITION

Manage your Application Portfolio, enable Code Quality & Security at an Enterprise level.

[Download](#)

Developer Edition plus:

- ✓ Portfolio Management & PDF Executive Reports
- ✓ Security Reports
- ✓ Project Transfer
- ✓ Parallel processing of analysis reports
- ✓ Support for Apex, COBOL, PL/I, RPG, VB6

Data Center
EDITION

High Availability, for global deployments

[Download](#)

Enterprise Edition plus:

- ✓ Component redundancy
- ✓ Data resiliency
- ✓ Horizontal Scalability



Instalação

<https://www.sonarqube.org/>

Thank you for downloading SonarQube Community Edition

Your download will start in a few seconds. If not, [Click here.](#)

NEW USER

Getting started guide
SonarQube is just two minutes away! Follow the guide to learn more.
[Learn more →](#)

UPGRADE

Upgrading to the latest release
To upgrade, read the guide and the relevant notes for your specific version.
[Check out our upgrade notes →](#)



DEVELOPER EDITION

Take your delivery pace to the next level with SonarQube Developer Edition

- ⚡ C, C, C++, Obj-C, Swift, ABAP, T-SQL, PL/SQL support
- 🔒 Taint analysis / injection detection for Java, C#, PHP, Python, JavaScript, TypeScript
- ↗ Extensive coverage of OWASP Top 10
- ↙ On-prem and in-cloud Pull Request analysis and decoration Options
- ↔ Pull Request analysis and decoration for:

 GitHub  GitLab  Bitbucket Server  Azure DevOps

[Request Free Trial](#)



Instalação

<https://www.sonarqube.org/>

- ✓ Executar como administrador
- ✓ Necessário Java 11

From the zip file

1. [Download](#) the SonarQube Community Edition zip file.
2. As a **non-root user**, unzip it, let's say in *C:\sonarqube* or */opt/sonarqube*.
3. As a **non-root user**, start the SonarQube Server:

```
# On Windows, execute:  
C:\sonarqube\bin\windows-x86-64\StartSonar.bat
```

```
# On other operating systems, as a non-root user execute:  
/opt/sonarqube/bin/[OS]/sonar.sh console
```

If your instance fails to start, check your [logs](#) to find the cause.



Analisando um Projeto

<https://www.sonarqube.org/>

<http://localhost:9000>

login: admin

password: admin

Analyzing a Project

Now that you're logged in to your local SonarQube instance, let's analyze a project:

1. Click the **Create new project** button.
2. Give your project a **Project key** and a **Display name** and click the **Set Up** button.
3. Under **Provide a token**, select **Generate a token**. Give your token a name, click the **Generate** button, and click **Continue**.
4. Select your project's main language under **Run analysis on your project**, and follow the instructions to analyze your project. Here you'll download and execute a Scanner on your code (if you're using Maven or Gradle, the Scanner is automatically downloaded).

Abrindo a porta 9000 no Windows

=====

```
C:\WINDOWS\system32>netsh advfirewall firewall add rule name="sonarqube 9000" dir=in action=allow protocol=TCP localport=9000  
C:\WINDOWS\system32>netsh advfirewall firewall add rule name="sonarqube 9000" dir=out action=allow protocol=TCP localport=9000
```

<https://wiki.mcneel.com/zoo4/usingnetsh>



Analizando um Projeto

Create a project

All fields marked with * are required

Project key *

A green validation icon with a checkmark is positioned to the right of the input field.

Up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Display name *

A green validation icon with a checkmark is positioned to the right of the input field.

Up to 255 characters

Set Up

Teste02 ★ master +

Overview Issues Security Hotspots Measures Code Activity

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1

Provide a token

Generate a token

Generate

Use existing token

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



Analizando um Projeto

1 Provide a token

teste02: 14acc4aae0f54d11cd54f24aa082e7118c4204ba ✖

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

[Continue](#)

2 Run analysis on your project

What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux Windows macOS



Analizando um Projeto

Download and unzip the Scanner for Windows

Visit the [official documentation of the Scanner](#) to download the latest version, and add the `bin` directory to the `%PATH%` environment variable

Execute the Scanner from your computer

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner.bat -D"sonar.projectKey=Teste02" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=14acc4aae0f54d11cd54f24aa082e7118c4204ba"
```

Copy

Please visit the [official documentation of the Scanner](#) for more details.

Once the analysis is completed, this page will automatically refresh and you will be able to browse the analysis results.

```
sonar-scanner.bat -D"sonar.projectKey=Teste02" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=14acc4aae0f54d11cd54f24aa082e7118c4204ba"
```

How do I set or change the PATH system variable?

=====

<https://java.com/en/download/help/path.html>

No Windows via Linha de Comando:

=====

```
setx path "%path%;C:\Users\Acer\Downloads\sonar-scanner-4.6.2.2472-windows\bin"
```

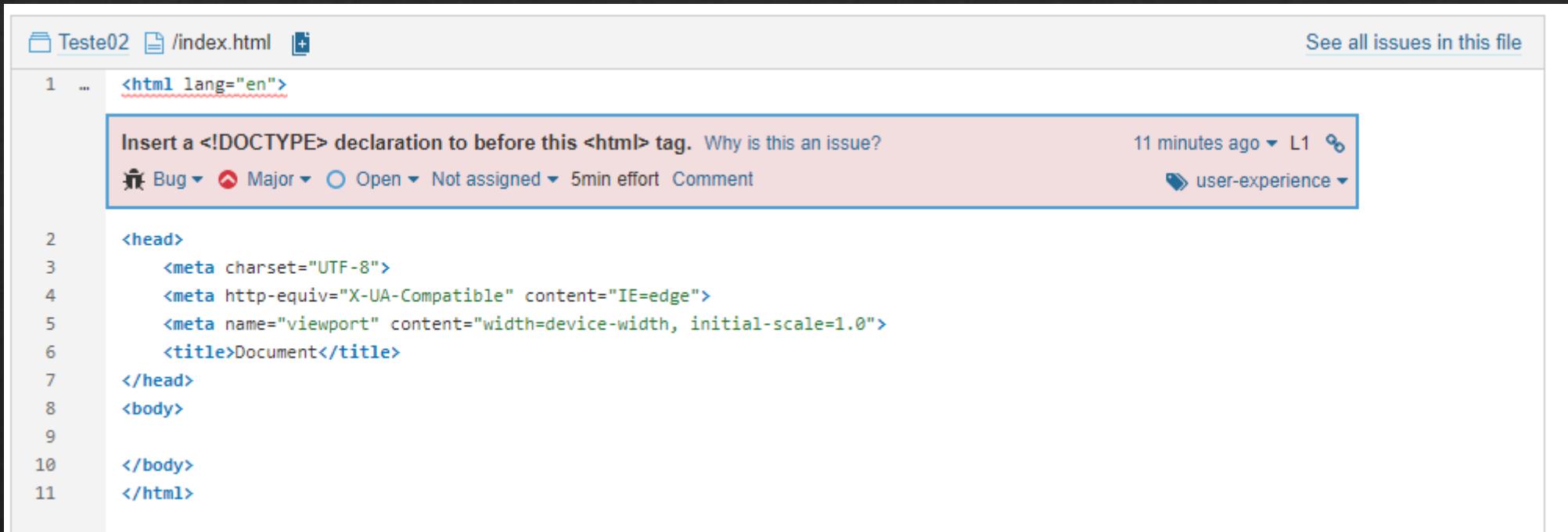


Analizando um Projeto

<p>★ teste01 Passed</p>	Last analysis: 3 hours ago					
<p>Bug</p> <p>0 A</p>	<p>Vulnerabilities</p> <p>0 A</p>	<p>Hotspots Reviewed</p> <p>- A</p>	<p>Code Smells</p> <p>0 A</p>	<p>Coverage</p> <p>-</p>	<p>Duplications</p> <p>0.0% O</p>	<p>Lines</p> <p>0 xs</p>
<p>★ Teste02 Passed</p>	Last analysis: 10 minutes ago					
<p>Bug</p> <p>1 C</p>	<p>Vulnerabilities</p> <p>0 A</p>	<p>Hotspots Reviewed</p> <p>- A</p>	<p>Code Smells</p> <p>0 A</p>	<p>Coverage</p> <p>-</p>	<p>Duplications</p> <p>0.0% O</p>	<p>Lines</p> <p>10 xs HTML</p>



Analizando um Projeto



The screenshot shows the SonarQube interface for a project named "Teste02" with a file named "/index.html". A specific issue is highlighted in a blue box:

1 ... <html lang="en">

Insert a <!DOCTYPE> declaration to before this <html> tag. Why is this an issue? 11 minutes ago ▾ L1 🔍

Bug ▾ Major ▾ Open ▾ Not assigned ▾ 5min effort Comment user-experience ▾

2 <head>
3 <meta charset="UTF-8">
4 <meta http-equiv="X-UA-Compatible" content="IE=edge">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 </body>
11 </html>



sonarcloud.io

Apps (4) Feed | LinkedIn Hotmart Webinar ... Notícias Mainframe Bancos de Imagens Web Marketing Documentação Tra... Linguagens de Pro... Data Base

sonarcloud Features What we do What's new Pricing Explore Log in

Clean Code Rockstar Status

Eliminate bugs and vulnerabilities. Champion quality code in your projects.

Go ahead! Analyze your repo:

GitHub Bitbucket

Azure DevOps GitLab

Free for Open-Source Projects

QUALITY GATE Passed

RELIABILITY 0 Bugs	MAINTAINABILITY 175 Code smells
SECURITY 0 Vulnerabilities	SECURITY REVIEW 0 Security Hotspots
COVERAGE 90%	DUPLICATIONS 1.4%

Coverage on 2.5k New Lines to cover
Duplications on 25k New Lines