



Integração Contínua – Prática - Etapa D

Prof. Jean Carlo Rossa Hauck
Prof. Osmar de Oliveira Braz Junior
Prof. Richard Henrique de Souza

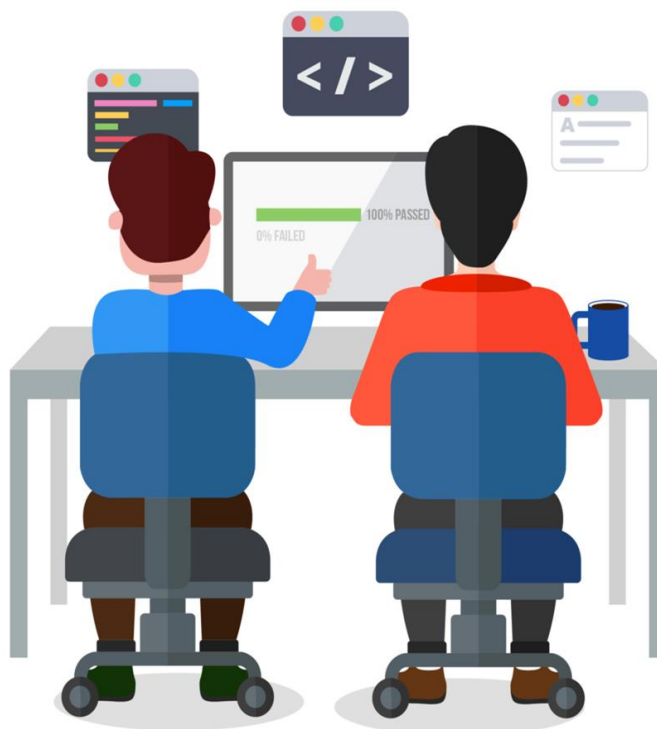
Objetivos

- Realizar um exemplo de **integração contínua** da **análise** até **cobertura** do código.
- A **integração** continua será realizada em 3 ambientes distintos com tarefas distintas.
- A **análise** irá considerar diversas métricas de qualidade de software como confiabilidade, manutibilidade, segurança, **cobertura** e duplicação de código.
- Nesta etapa realizaremos a **cobertura** do código.

Atividade em Grupo

Para esta atividade crie grupos de 2 alunos, para desenvolver a atividade segundo ***Pair Programming***.

Navegador



Piloto

Pair Programming

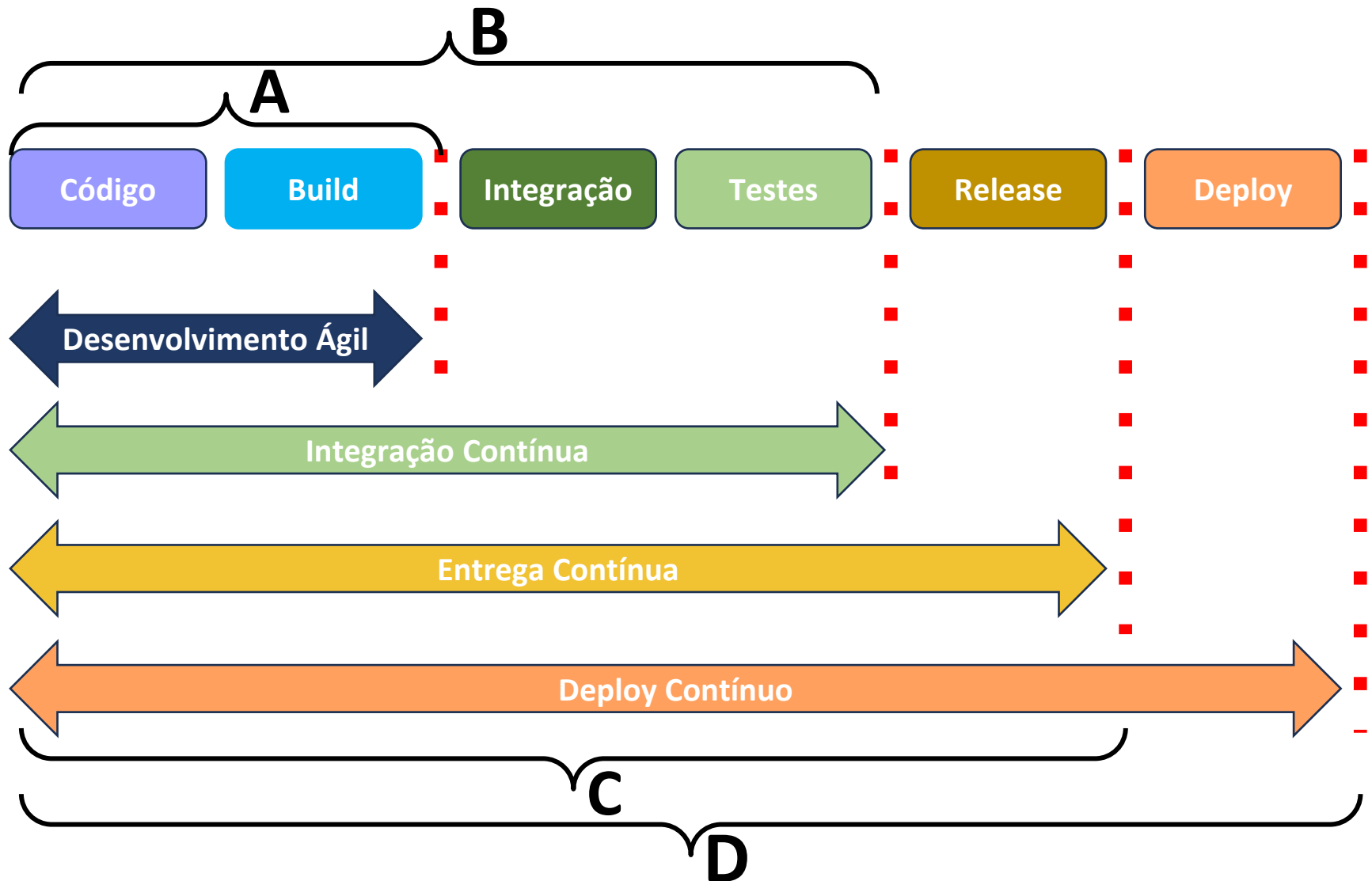
- Um é o **piloto**, responsável por escrever o código, o outro o navegador, acompanha a escrita de código e verificar se está de acordo com os **padrões do projeto** e de encontro à solução necessária.
- A intenção desta técnica é **evitar** erros de lógica, e ter um código mais confiável e melhor estruturado, utilizando-se para isso a máxima de que “**duas cabeças pensam melhor do que uma**”.

Integração, Entrega e Implantação Contínua

■ Abstração do Pipeline



Integração, Entrega e Implantação Contínua



Ferramentas utilizadas

- IDE com suporte
 - Apache Maven
 - JUnit 4
 - Github
- Github
 - Github Actions
 - Sonarcloud
 - JaCoCo

maven

JUnit



sonarcloud 

JACOCO
Java Code Coverage

Atividades práticas

- A - Criação de Projeto e testes unitários
 - Criar projeto na IDE
 - Automatizado com Apache Maven
 - Criar testes unitários com JUnit 4
 - Armazenar projeto no Github
- B - Integração Continua
 - Github Actions
 - JUnit 4
- C - Análise do Código
 - Sonarcloud
 - Integração com Github Actions
- D - Cobertura do código
 - Jacoco
 - Integração com Github Actions
 - Integração com Maven e Sonarcloud

D – Cobertura do Código

- Jacoco
- Github Action
- Integração com Maven e Sonarcloud

maven



sonarcloud 

JACOCO
Java Code Coverage

Passo 1

Adicionando o plugin do Jacoco ao pom.xml

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.6</version>
  <configuration>
    <excludes>
      <exclude>**/*Principal.*</exclude>
    </excludes>
  </configuration>
  <executions>
    <execution>
      <id>prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>report</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
</plugins>
</build>
</project>
```

Exclui localmente o programa **Principal.java** da cobertura.

Para excluir o arquivo do programa “**Principal.java**” do sonarcloud, adicione na tag **properties**.

```
<sonar.coverage.exclusions>**/*Principal.*</sonar.coverage.exclusions>
```



Passo 2

Alterar o arquivo da integração contínua (**maven.yml**) para enviar o arquivo do relatório do Jacoco para o sonar.

Localize a linha com o texto abaixo:

```
run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar  
-Dsonar.projectKey=osmarbraz_calculadora
```

Adicione o texto abaixo ao final da linha anterior:

```
org.jacoco:jacoco-maven-plugin:prepare-agent  
-Dsonar.coverage.jacoco.xmlReportPaths=target/site/jacoco/jacoco.xml
```

Linha alterada:

```
run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar  
-Dsonar.projectKey=osmarbraz_calculadora  
org.jacoco:jacoco-maven-plugin:prepare-agent  
-Dsonar.coverage.jacoco.xmlReportPaths=target/site/jacoco/jacoco.xml
```

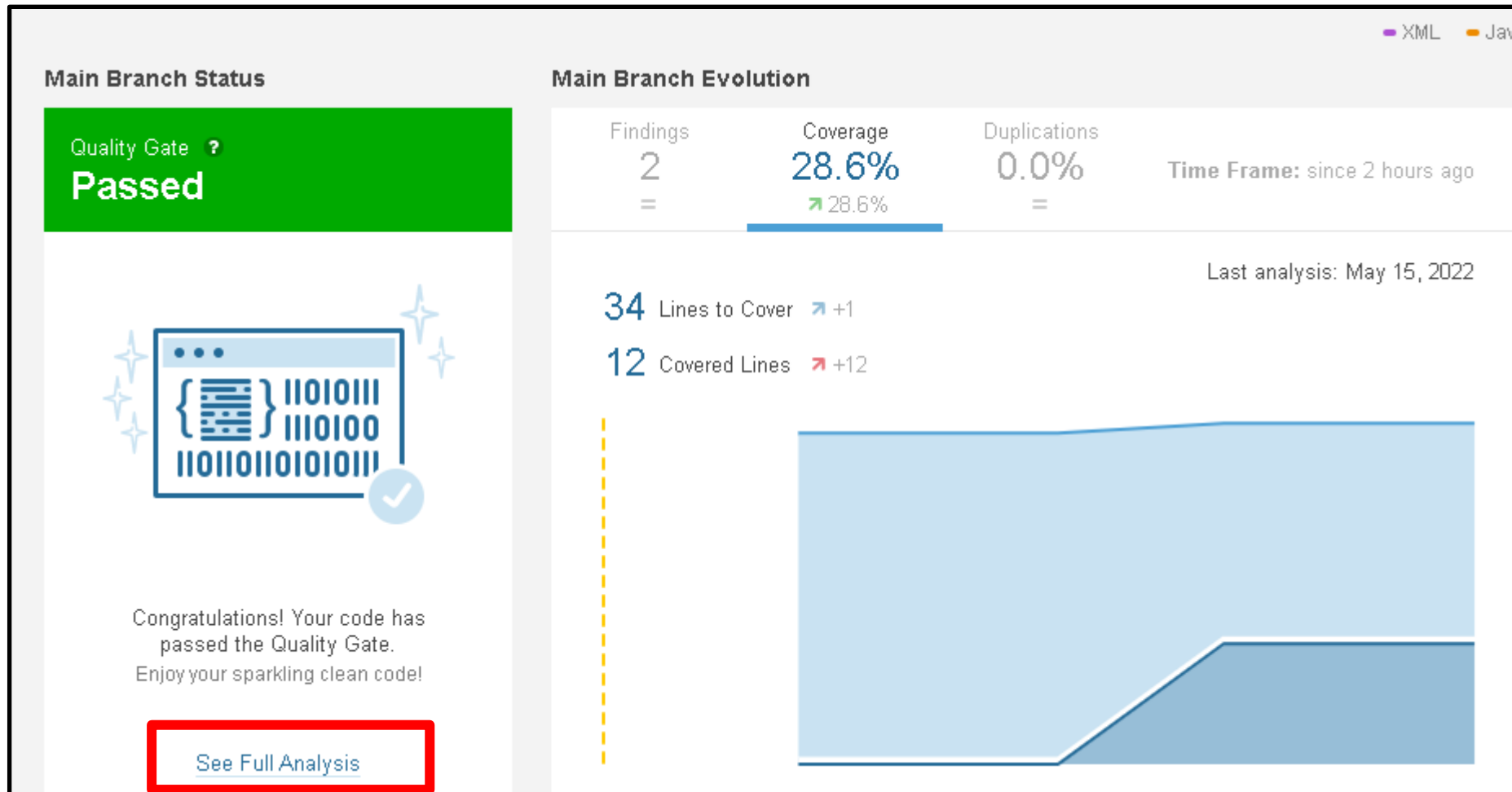
O comando deve ser colocado todo em uma linha.

Passo 3

Submeta ao repositório **github** as alterações dos arquivos **pom.xml** e **maven.yml**.

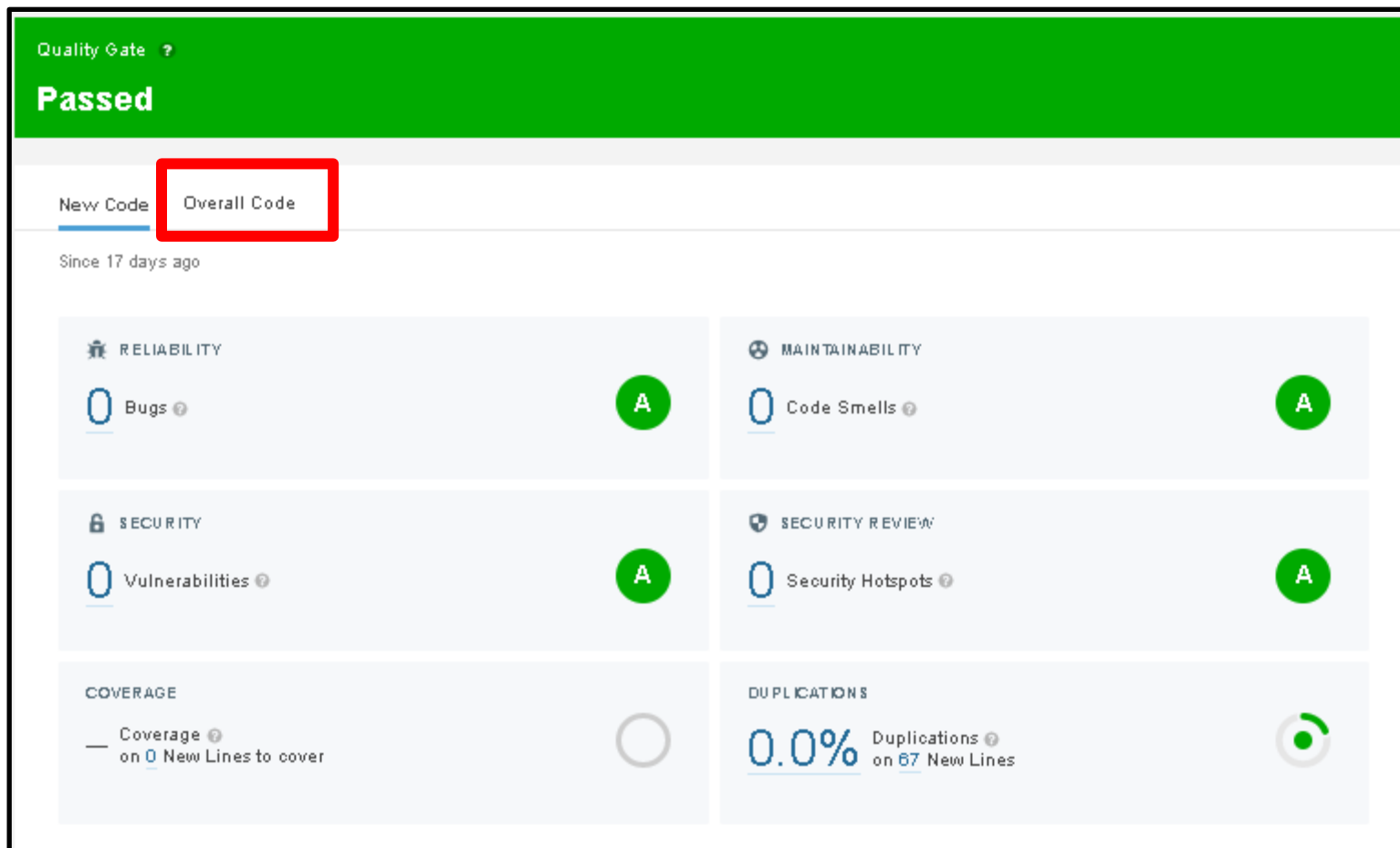
Passo 4

Visualizando as informações de cobertura(*coverage*).



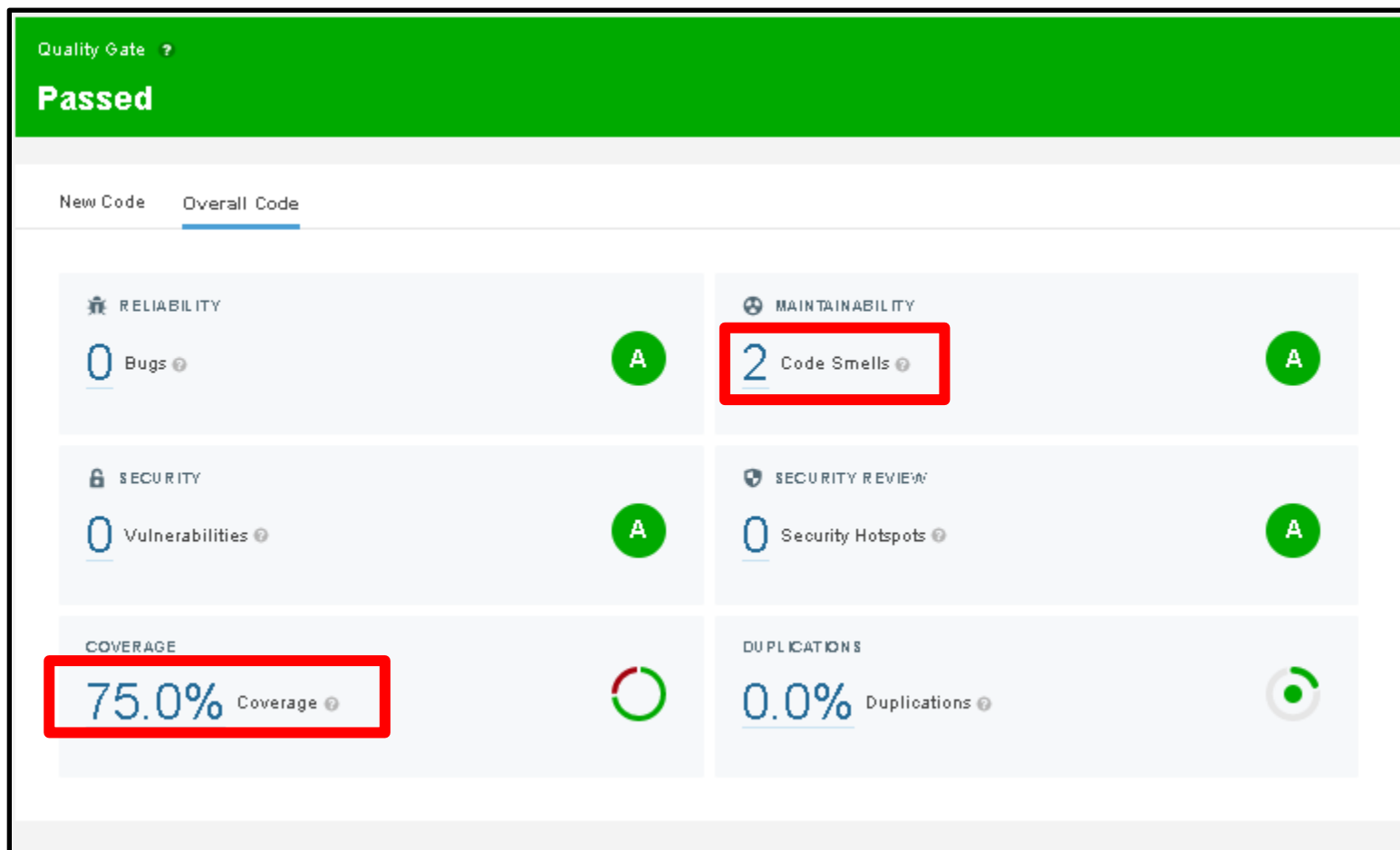
Passo 4

Análise de código novo (*New Code*) - Ultimo commit



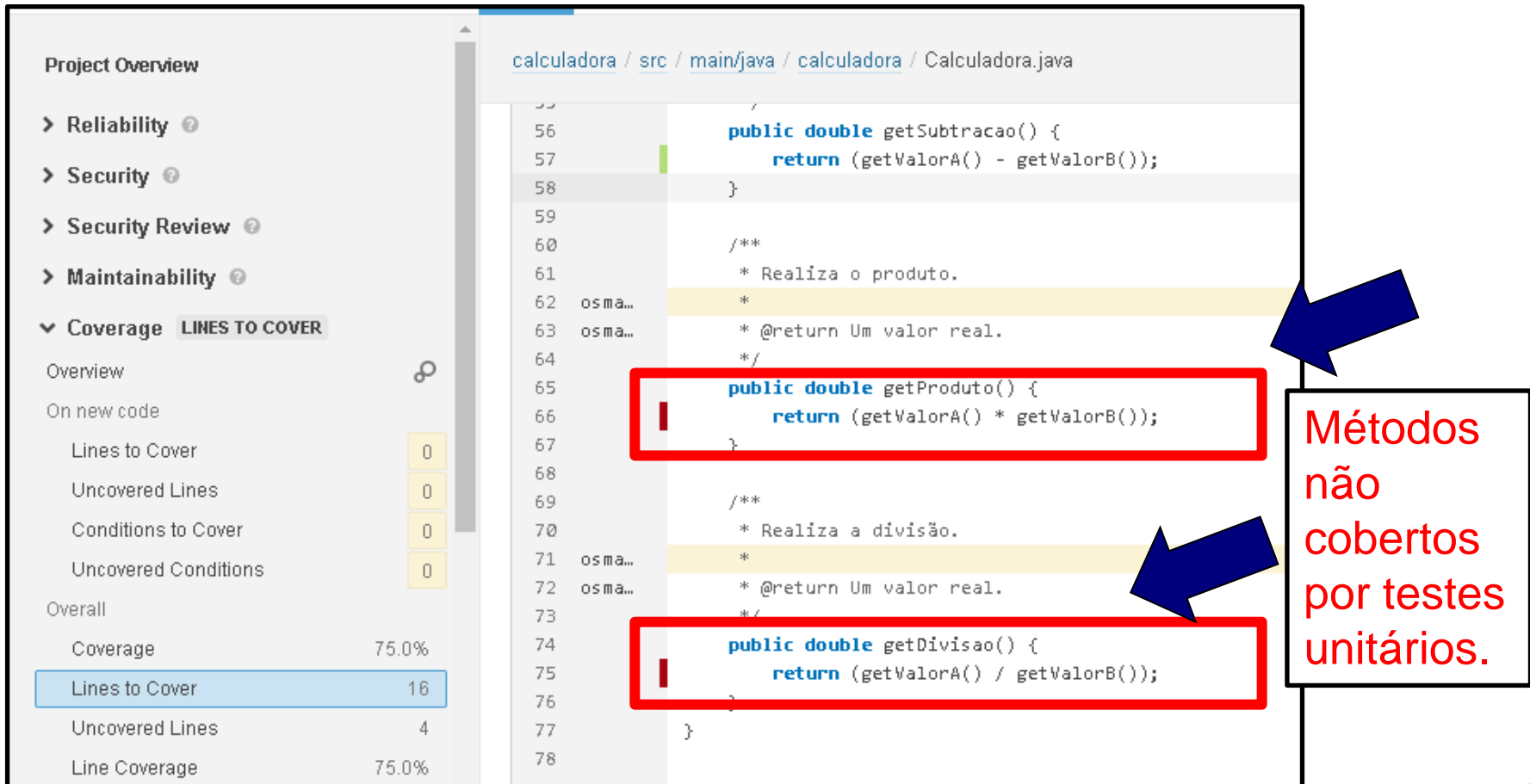
Passo 4

Clique nos números e análise as medidas.



Passo 4

Visualizando as informações de cobertura.



Project Overview

- > Reliability ?
- > Security ?
- > Security Review ?
- > Maintainability ?
- ▼ **Coverage** **LINES TO COVER**

Overview

On new code

Lines to Cover	0
Uncovered Lines	0
Conditions to Cover	0
Uncovered Conditions	0

Overall

Coverage	75.0%
Lines to Cover	16
Uncovered Lines	4
Line Coverage	75.0%

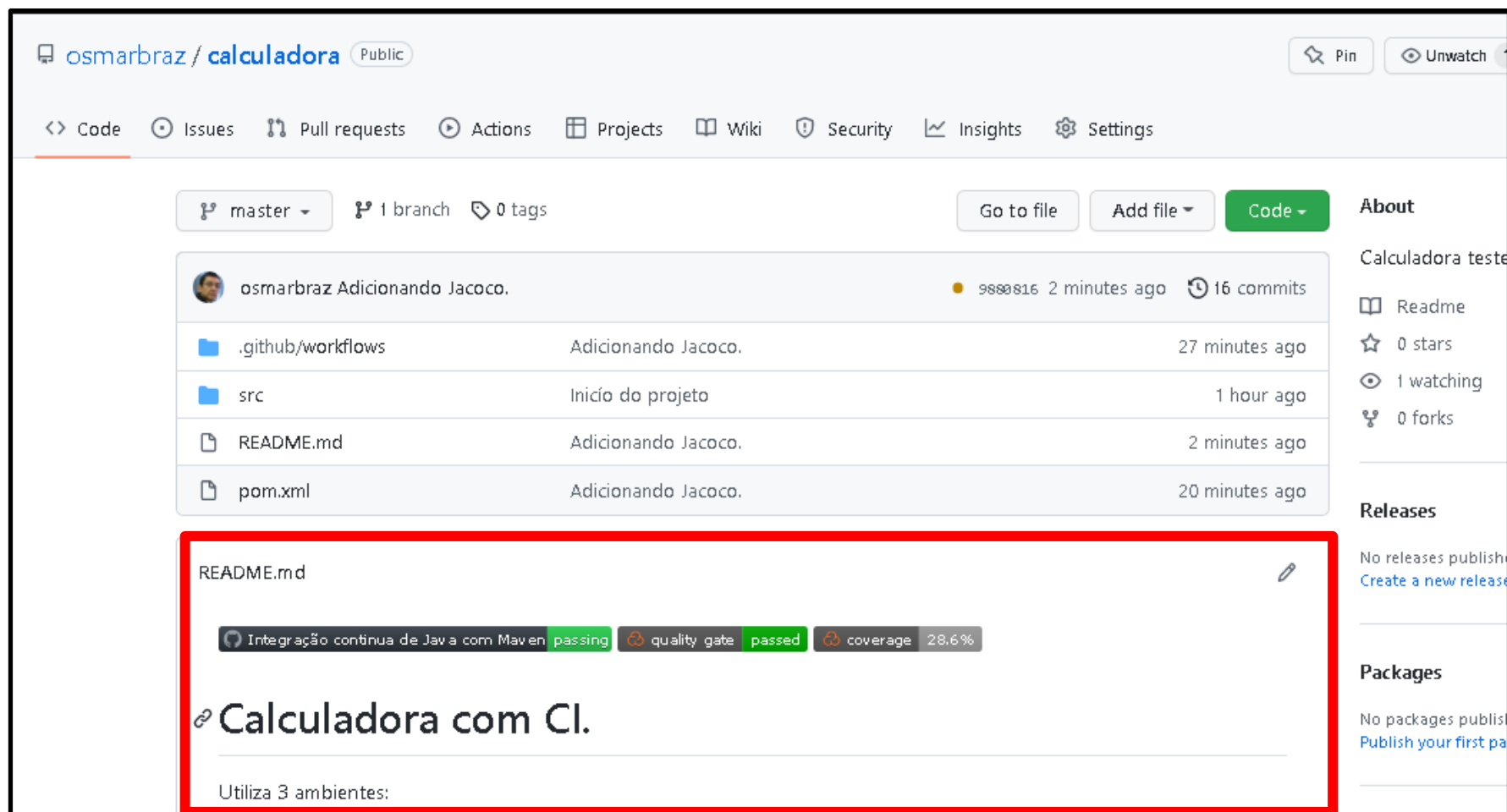
calculadora / src / main/java / calculadora / Calculadora.java

```
56 public double getSubtracao() {
57     return (getValorA() - getValorB());
58 }
59
60 /**
61  * Realiza o produto.
62  * osma...
63  * @return Um valor real.
64  */
65 public double getProduto() {
66     return (getValorA() * getValorB());
67 }
68
69 /**
70  * Realiza a divisão.
71  * osma...
72  * @return Um valor real.
73  */
74 public double getDivisao() {
75     return (getValorA() / getValorB());
76 }
77
78 }
```

Métodos não cobertos por testes unitários.

Passo 5

Acesse o repositório no **github**.



The screenshot shows the GitHub interface for the repository 'osmarbraz/calculadora'. The repository is public and has 1 branch (master) and 0 tags. The commit history shows a recent commit by 'osmarbraz' titled 'Adicionando Jacoco.' with 16 commits. The file list includes '.github/workflows', 'src', 'README.md', and 'pom.xml'. The 'README.md' file is highlighted with a red box, showing the title 'Calculadora com CI.' and the text 'Utiliza 3 ambientes:'. The CI status bar indicates 'Integração contínua de Java com Maven' is passing, 'quality gate' is passed, and 'coverage' is 28.6%.

osmarbraz / calculadora Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

osmarbraz Adicionando Jacoco. 9880816 2 minutes ago 16 commits

File	Commit Message	Time
.github/workflows	Adicionando Jacoco.	27 minutes ago
src	Início do projeto	1 hour ago
README.md	Adicionando Jacoco.	2 minutes ago
pom.xml	Adicionando Jacoco.	20 minutes ago

README.md

Integração contínua de Java com Maven **passing** quality gate **passed** coverage 28.6%

Calculadora com CI.

Utiliza 3 ambientes:

Atividade Final



- 1 - Aumente a cobertura do código com testes para os outros métodos da classe calculadora.
- 2 - Avalie o resultado da execução do workflow
- 3 - Avalie o resultado da análise do sonar.

Conclusão

- A Integração Continua é um processo essencial a qualquer software que deseja manter vivo por um período de tempo mais longo.
- Conhecer e dominar as ferramentas é um ponto crítico para garantir agilidade no processo de distribuição do software.
- Realizamos a cobertura do código e completamos todas as etapas propostas.

Referências

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016.
- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.
- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007



Fim