



Como garantir que o teste unitário que estou escrevendo **realmente** possui qualidade?

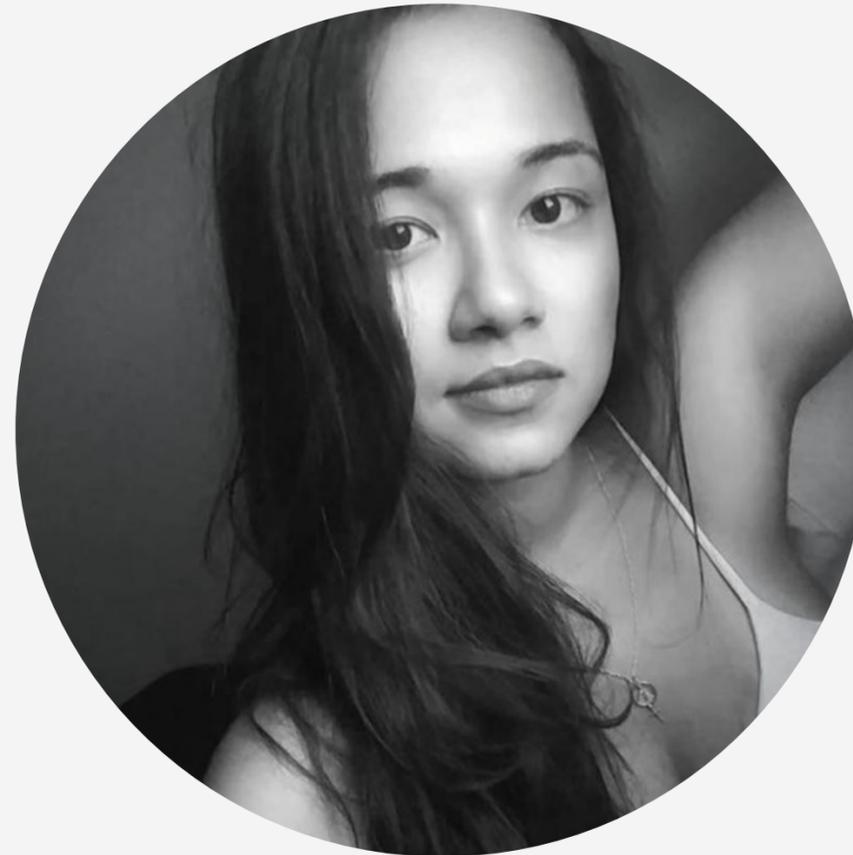
com Bárbara Rossalli & Daniela Yabe

Quem somos



Bárbara Rossalli

<https://linktr.ee/Rossalli>

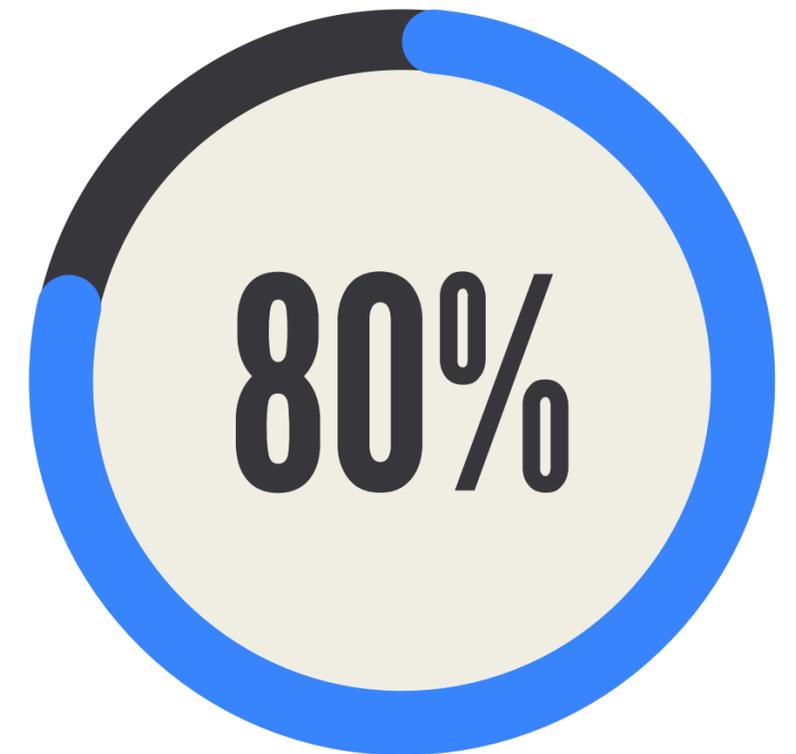


Daniela Yabe

<https://linktr.ee/DanielaYabe>

Como garantir que o teste unitário
que estou escrevendo **realmente**
possui qualidade?

Test Coverage



É uma métrica que valida o quanto do código foi coberto pelos testes

ou seja

Quais linhas de código foram executadas

O problema do Coverage

A cobertura
de código
não valida a
eficiência dos
testes



Métrica de test coverage
do meu projeto é 80%



```
public class FluxCapacitor {  
  
    private Double mrFusionEnergy;  
    private Integer deloreanVelocity;  
  
    public FluxCapacitor(Double mrFusionEnergy, Integer deloreanVelocity) {  
        this.mrFusionEnergy = mrFusionEnergy;  
        this.deloreanVelocity = deloreanVelocity;  
    }  
  
    public Double getMrFusionEnergy() {  
        return mrFusionEnergy;  
    }  
  
    public void setMrFusionEnergy(Double mrFusionEnergy) {  
        this.mrFusionEnergy = mrFusionEnergy;  
    }  
  
    public Integer getDeloreanVelocity() {  
        return deloreanVelocity;  
    }  
  
    public void setDeloreanVelocity(Integer deloreanVelocity) {  
        this.deloreanVelocity = deloreanVelocity;  
    }  
}
```

© FluxCapacitor 60% methods, 60% lines covered



```
public class FluxCapacitorTest {  
  
    @DisplayName("Building flux capacitor")  
    @Test  
    public void fluxCapacitorBuild() {  
        FluxCapacitor fluxCapacitor = new FluxCapacitor(0.0, 0);  
        fluxCapacitor.setMrFusionEnergy(1.21);  
        assertEquals(fluxCapacitor.getMrFusionEnergy(), 1.21);  
        fluxCapacitor.setMrFusionEnergy(88);  
        assertEquals(fluxCapacitor.getDeloreanVelocity(), 88);  
    }  
}
```



© FluxCapacitor 100% methods, 100% lines covered



**HOW DO WE FIX
THIS?**

Testes
mutantes em
ação!

Porque testes
mutantes?

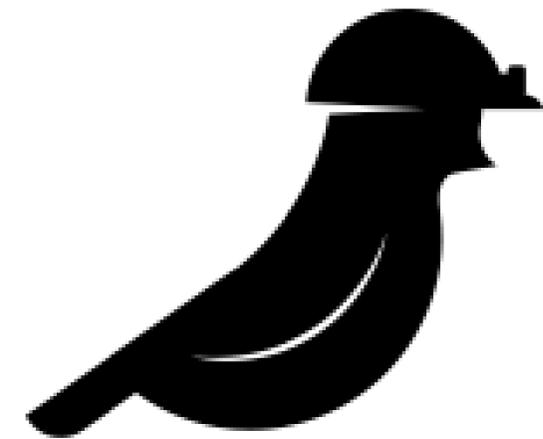


O que são testes mutantes?

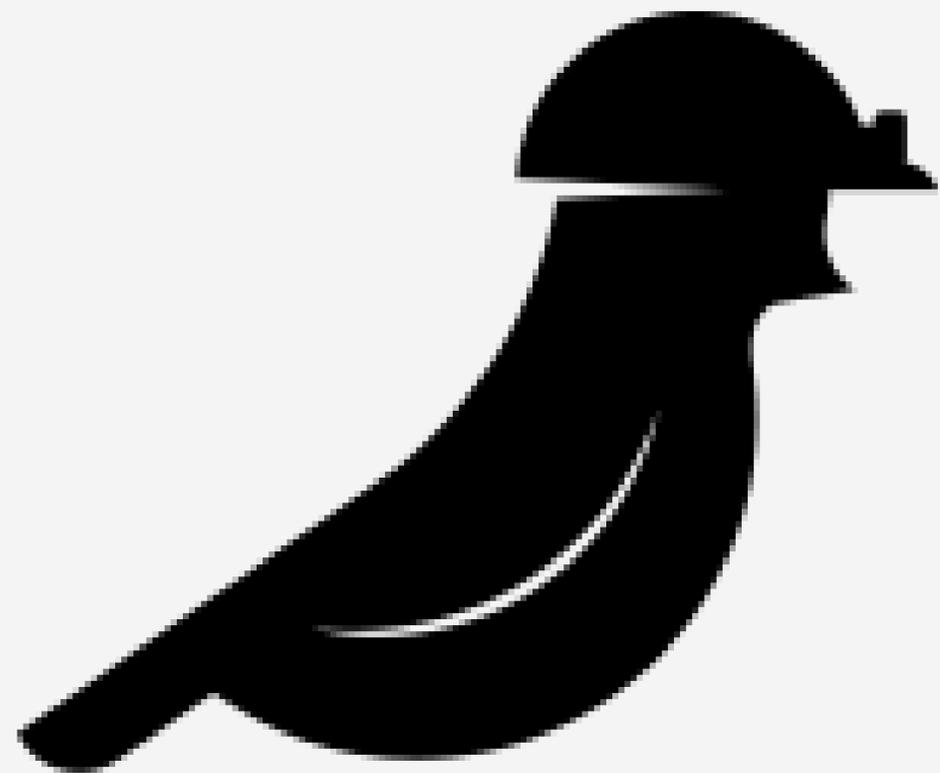


- 🔍 Mutantes são inseridos no código
- 🔍 Testes executados em cima do código mutado
- 🔍 Se o teste falhar o mutante foi morto
- 🔍 Se um mutante sobreviver, significa que você precisa melhorar seus testes unitários :)

Ferramentas



pitest.org



pitest.org

PITest pitest.org

- Versão 1.6.9
- Release em 2014
- Henry Coles
- Apache 2.0
- Maven, Gradle, Ant
- Java 8 ou + (1.4.0)
- Junit4 e Junit5*
- TestNG 6.1.1
- JMock, EasyMock, Mockito, PowerMock and JMockit.
- Eclipse, IntelliJ



CODE

Adventure time!



KILL
ALL
COVERAGE
TESTS!

**Teste mutantes
não substitui a
cobertura de
código!**



É um complemento!

Desvantagens:

Poder Computacional & Tempo de Execução

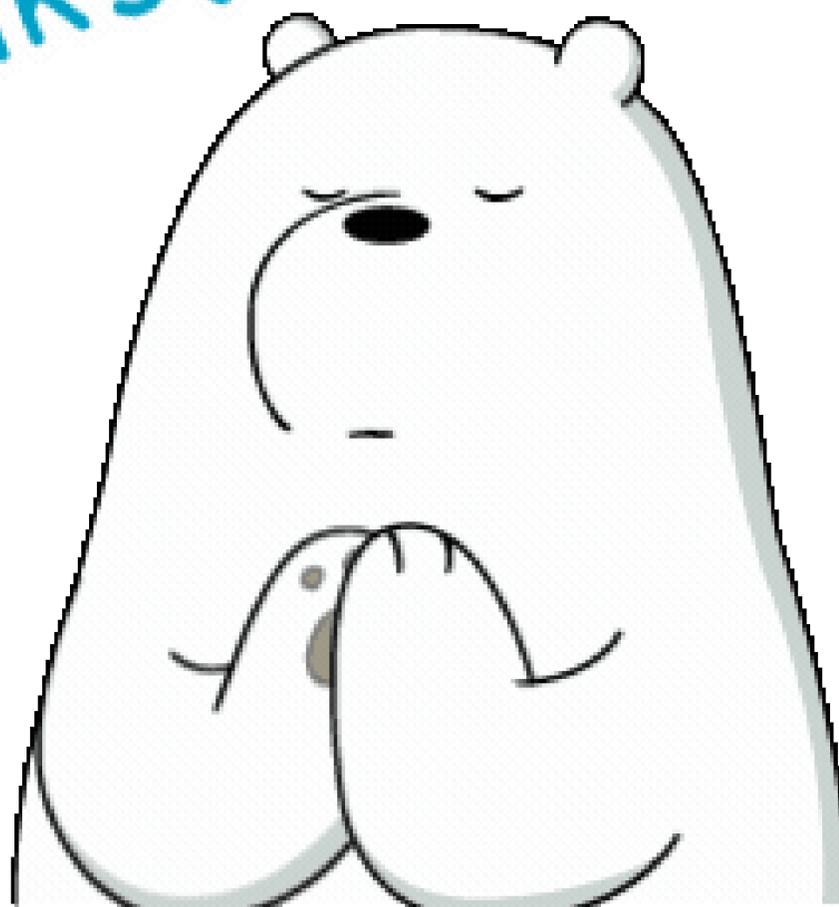


**State of Mutation Testing at Google:
<https://research.google/pubs/pub46584>**



**Qualidade dos meus testes
mutantes depende da
qualidade dos casos de testes
escritos!**

Thanks!



@Rossalli

@YabeDaniela