

# Testes de unidade na prática

Por quê? Quando? Como?

Tiago Furtado <[tfurtado@gmail.com](mailto:tfurtado@gmail.com)>  
Full-stack developer, Meritt  
XVII SECCOM





# Tiago Furtado

Mestre em Ciência da Computação  
Universidade Federal de Viçosa (UFV)



unopar

 **meritt**<sup>®</sup>  
cada aluno é unico



**Vamos falar de coisa boa?**



# Testes de unidade

#FALLONTONIGHT

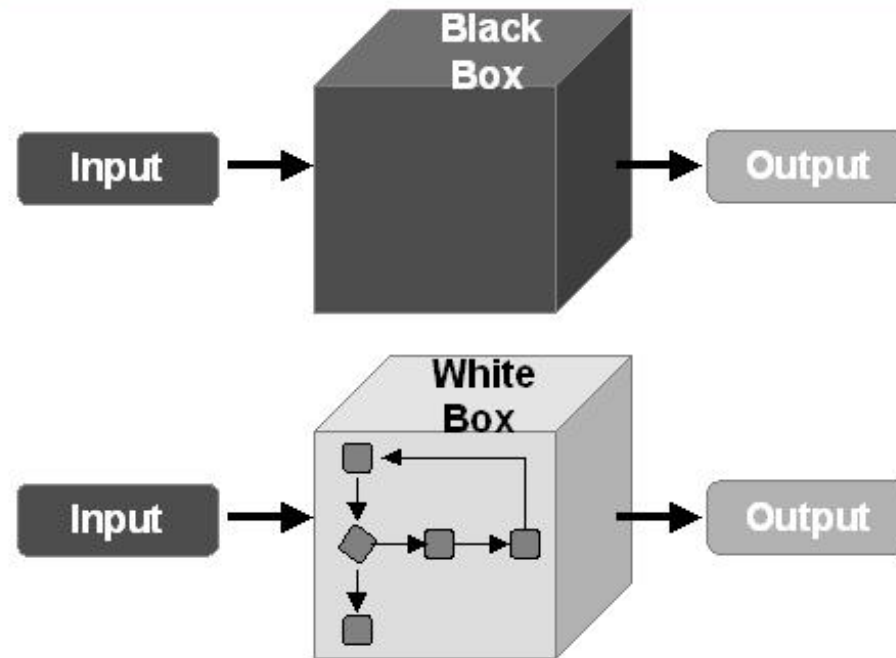
...WHAT?



# Testes de ~~unidade~~ software

# Testes de software

## Comparison among Black-Box & White-Box Tests

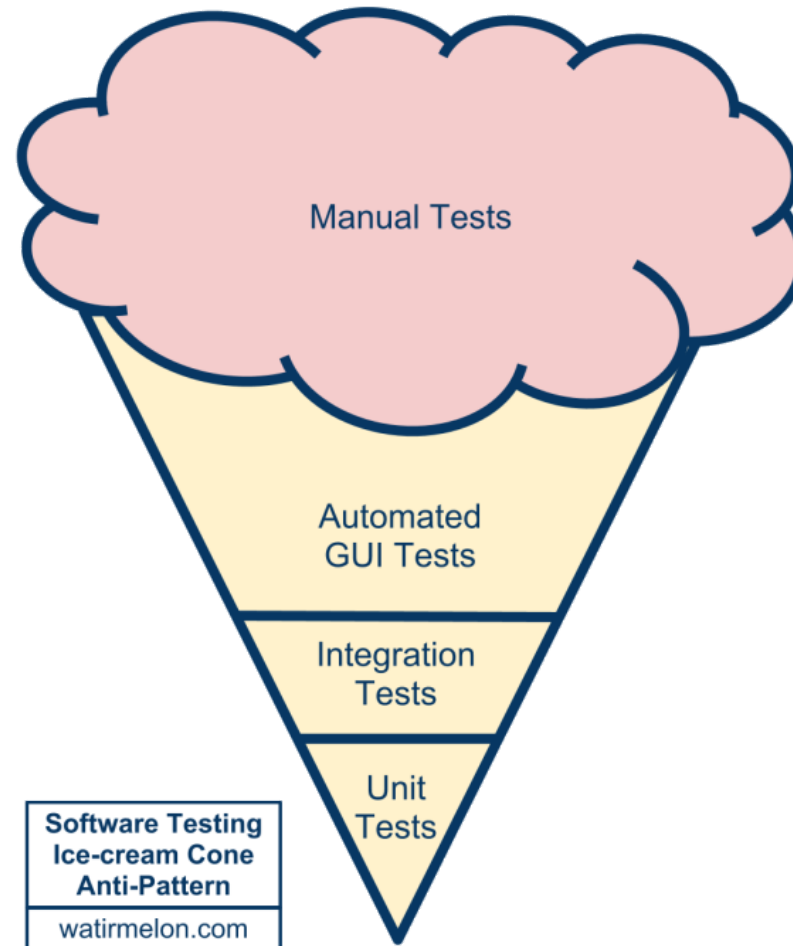


[www.softwaretestinggenius.com](http://www.softwaretestinggenius.com)

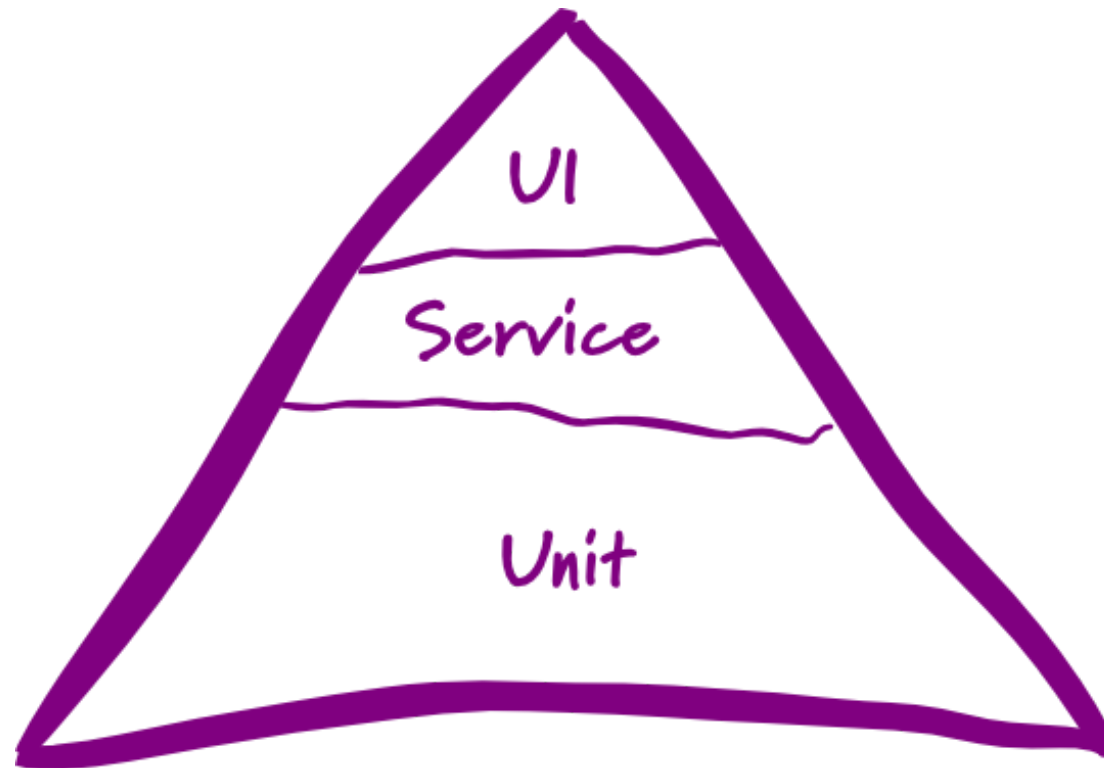




# Testes de software



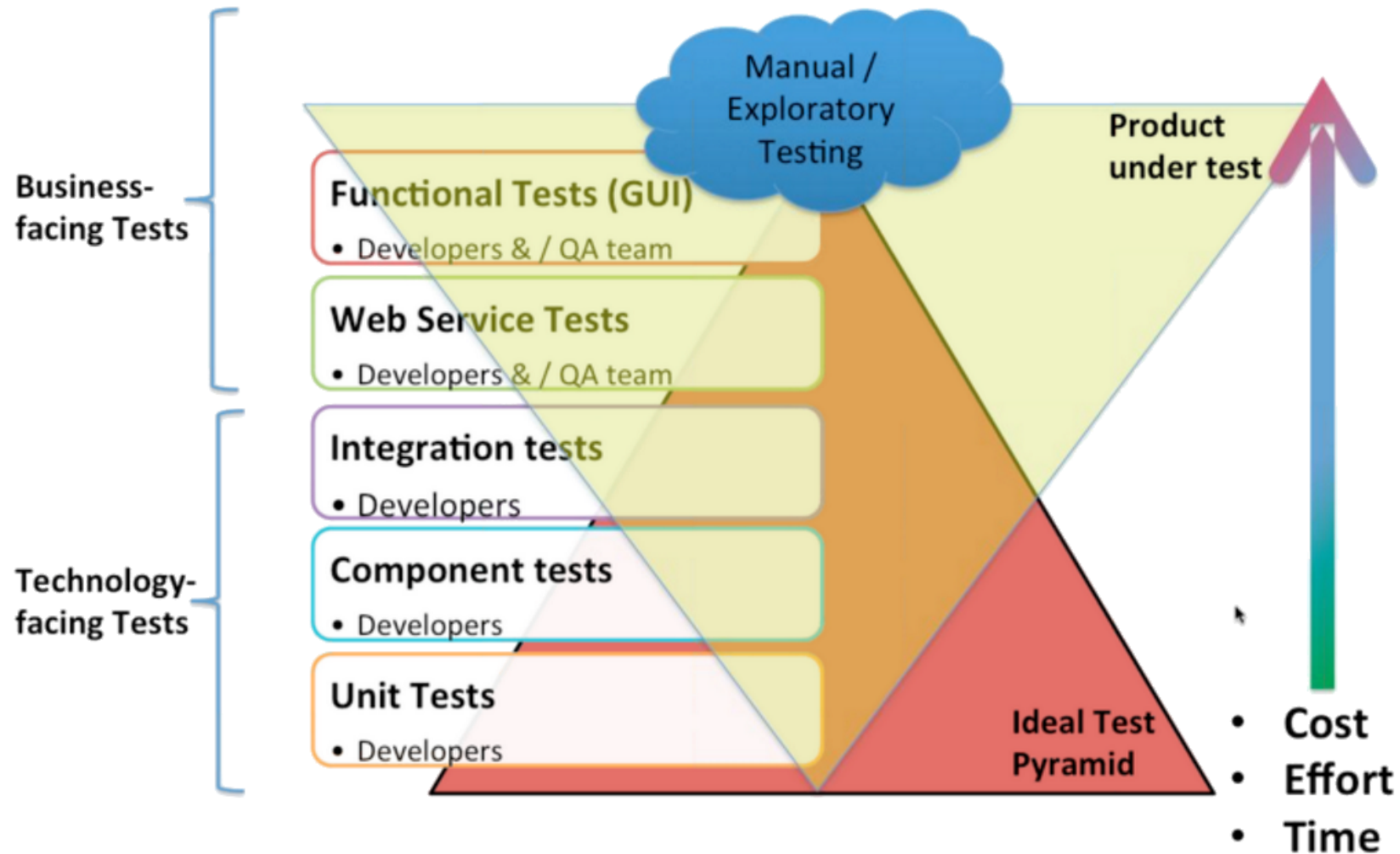
# Testes de software



Martin Fowler. [martinfowler.com/bliki/TestPyramid.html](http://martinfowler.com/bliki/TestPyramid.html), 2012.



# Testes de software





# Testes de unidade

O que são? ~~Onde vivem?~~ ~~De que se alimentam?~~

# Testes de unidade

PHP

```
interface CalculatorInterface {
    public function sum($first, $second);
    public function subtract($first, $second);
}

class Calculator implements CalculatorInterface { /* ... */ }

class CalculatorTest {
    /** @test */
    public function sumShouldReturnTheSumOfTwoNumbers() {
        $calculator = new Calculator();
        $result = $calculator->sum(1, 2);
        if ($result !== 3) {
            throw new Exception("Something wrong's not right `\_(\ツ)\_/'");
        }
    }
}
```



# Testes de unidade

Por quê?

- Software livre de defeitos **deveria ser** sempre entregue
- Mas...
  - software é muito complexo
  - pessoas são imperfeitas! ><"



**Happens**



“O todo é maior do que a simples soma das suas partes.”

Aristóteles



# Testes de unidade

Por quê?

Garantir a correção das partes

PHP

```
interface PersonInterface {  
    public function canBeArrested();  
}
```

```
class Person implements PersonInterface {  
    public function __construct($age);  
    /*...*/  
}
```

```
(new Person(17))->canBeArrested();  
(new Person(18))->canBeArrested();  
(new Person(19))->canBeArrested();  
(new Person(-1))->canBeArrested();  
(new Person(0))->canBeArrested();  
(new Person("RUKM?"))->canBeArrested();
```



# Testes de unidade

Por quê?

Ampliar a capacidade de análise do todo

PHP

```
interface PersonInterface { /*...*/ }
class Person implements PersonInterface {
    public function __construct($age);
    /*...*/
}

interface PoliceInterface {
    public function arrest(PersonInterface $person);
}
class Police implements PoliceInterface { /*...*/ }

$police = new Police();
$police->arrest(new Person(15));
$police->arrest(new Person(19));
```

# Testes de unidade

Como?

Manualmente

```
function sum($a, $b) {  
    return $a + 0;  
}
```

PHP

```
printf("sum(%d, %d) = %d [%d]\n", 0, 0, sum(0, 0), 0);  
printf("sum(%d, %d) = %d [%d]\n", 1, 0, sum(1, 0), 1);  
printf("sum(%d, %d) = %d [%d]\n", 0, 1, sum(0, 1), 1);
```

```
sum(0, 0) = 0 [0]  
sum(1, 0) = 1 [1]  
sum(0, 1) = 0 [1]
```



# Testes de unidade

Como?

Manualmente

```
function sum($a, $b) {  
    return $a + $b;  
}
```

PHP

```
printf("sum(%d, %d) = %d [%d]\n", 0, 0, sum(0, 0), 0);  
printf("sum(%d, %d) = %d [%d]\n", 1, 0, sum(1, 0), 1);  
printf("sum(%d, %d) = %d [%d]\n", 0, 1, sum(0, 1), 1);
```

```
sum(0, 0) = 0 [0]  
sum(1, 0) = 1 [1]  
sum(0, 1) = 1 [1]
```





# Testes de unidade

PHP

```
1 <?php
2 function sum($a, $b) {
3     return 0 + $b;
4 }
5
6 function sumTest($a, $b, $c) {
7     if (sum($a, $b) === $c) { return; }
8     throw new Exception(/*...*/);
9 }
10 sumTest(0, 0, 0);
11 sumTest(1, 0, 1);
12 sumTest(0, 1, 1);
```

PHP Fatal error: Uncaught exception 'Exception' in sumTest.php:8

Stack trace:

#0 sumTest.php(11): sumTest(1, 0, 1)

#1 {main}

thrown in sumTest.php on line 8

# Testes de unidade

Como?

Usando um framework de testes

PHP

```
class Calculator {  
    public function sum($a, $b) {  
        return $a + $b;  
    }  
}  
  
class CalculatorTest extends PHPUnit_Framework_TestCase {  
    /** @test */  
    public function sumTest() {  
        $this->assertSame(0, sum(0, 0));  
        $this->assertSame(1, sum(1, 0));  
        $this->assertSame(1, sum(0, 1));  
    }  
}
```

# Testes de unidade

Como?

Manualmente



# Testes de unidade

Como?

~Automático~





# Testes de unidade

Como?

Usando um framework de testes



E quando um objeto interage com outro?



# Objetos duplês!



# Testes de unidade

PHP

```
class Police implements PoliceInterface {  
    public function arrest(PersonInterface $person) {  
        if (! $person->canBeArrested()) {  
            throw new Exception(/*...*/);  
        }  
    }  
}
```



# Testes de unidade

PHP

```
class PoliceTest extends PHPUnit_Framework_TestCase {  
    /** @test */  
    public function arrestShouldThrowExceptionIfPersonCantBeArrested() {  
        $person = $this->getMockForAbstractClass(PersonInterface::class);  
        $person->expects($this->once())  
            ->method('canBeArrested')  
            ->willReturn(false);  
  
        $this->setExpectedException(Exception::class);  
  
        $police = new Police();  
        $police->arrest($person);  
    }  
}
```



Quando escrevê-los?!







TDD

Test-driven development



# Test-driven development (TDD)

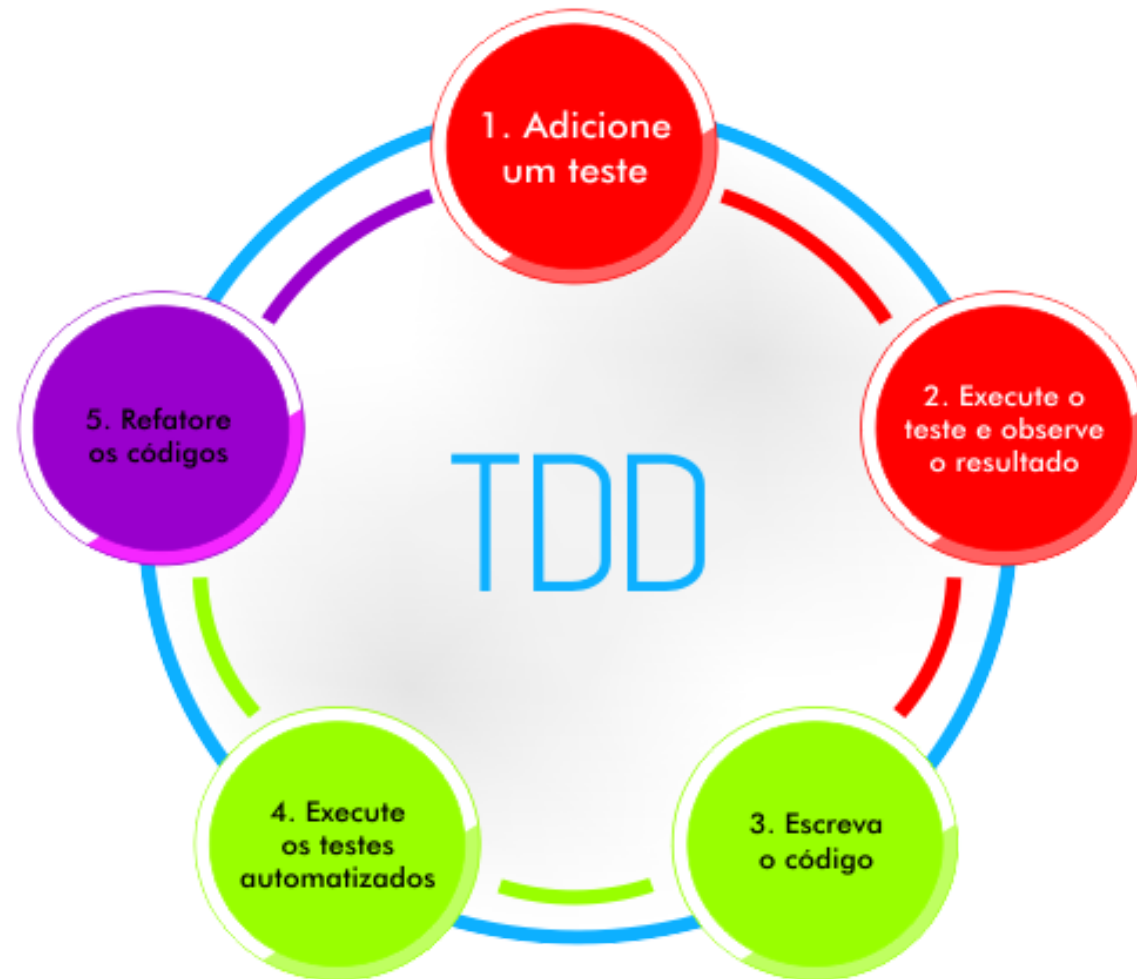
## Princípios

KISS: Keep It Simple, Stupid

YAGNI: You Aren't Gonna Need It



# Test-driven development (TDD)



# Test-driven development (TDD)

## 1. Escrever um teste

```
class CalculatorTest extends PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function sumTest()
    {
    }
}
```

PHP



# Test-driven development (TDD)

## 2. Garantir que o novo teste falha

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

```
.
```

```
Time: 75 ms, Memory: 2.75Mb
```

```
OK (1 test, 0 assertions)
```

wrong



# Test-driven development (TDD)

## 2. Garantir que o novo teste falha

```
class CalculatorTest extends PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function sumTest()
    {
        $calculator = new Calculator();
    }
}
```

PHP



# Test-driven development (TDD)

## 2. Garantir que o novo teste falha

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

PHP Fatal error:

```
Class 'TFurtado\UnitTestSample\Calculator' not found  
in ../test/CalculatorTest.php on line 19
```

PHP Stack trace:

```
PHP    1. {main}() ./vendor/phpunit/phpunit/phpunit:0  
    ...
```





**TÁ SERTO**

GERA

# Test-driven development (TDD)

## 3. Escrever código

```
class Calculator  
{  
  
}
```

PHP





# Test-driven development (TDD)

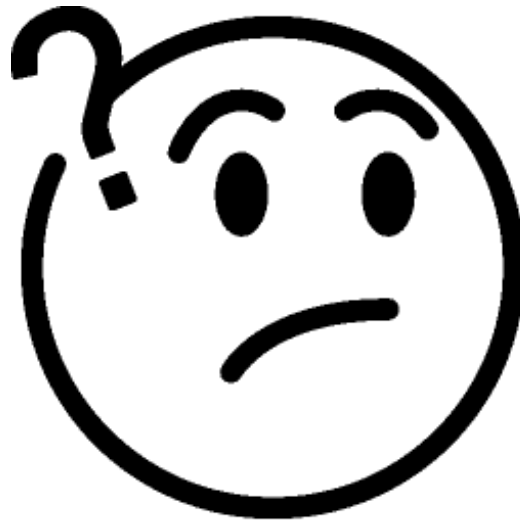
## 4. Garantir que os testes passam

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.  
  
.  
  
Time: 65 ms, Memory: 2.75Mb  
  
OK (1 test, 0 assertions)
```



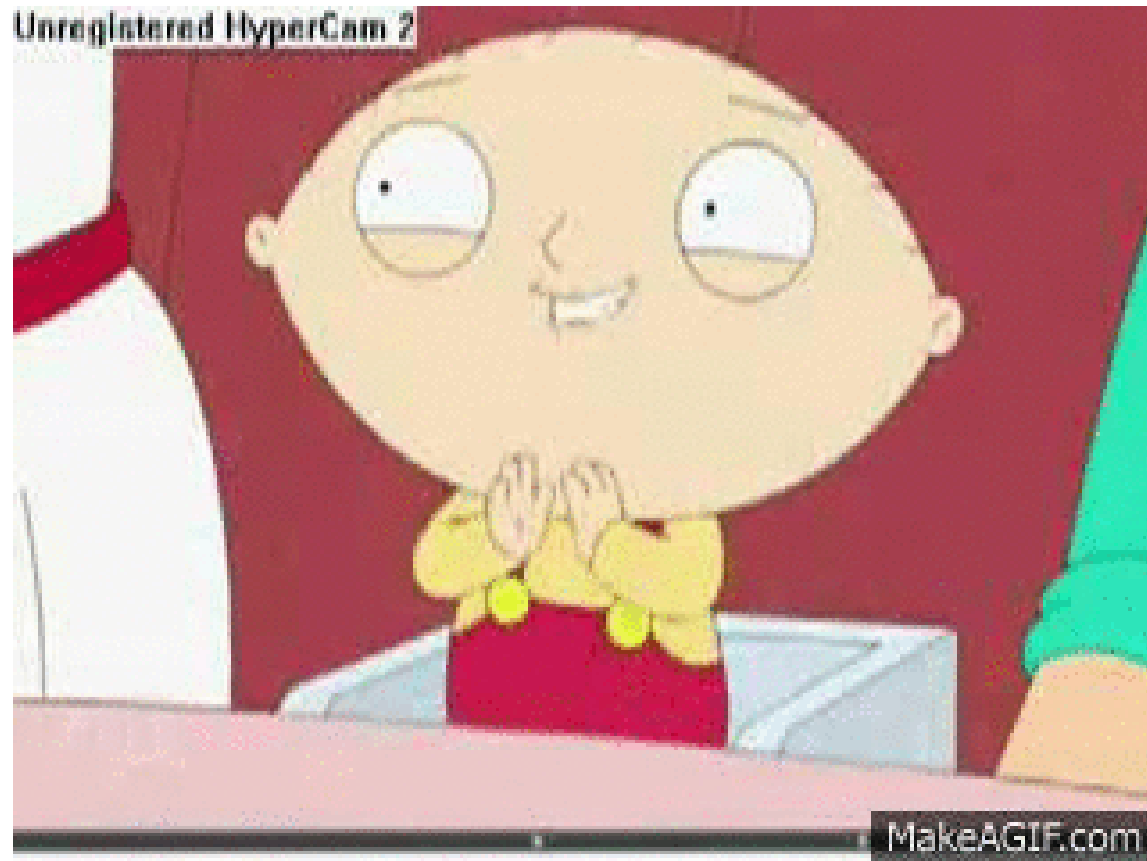
# Test-driven development (TDD)

## 5. Refatorar o código



# Test-driven development (TDD)

Repetir...



# Test-driven development (TDD)

## 1. Escrever um teste

```
class CalculatorTest extends PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function sumTest()
    {
        $calculator = new Calculator();
        $this->assertSame(3, $calculator->sum(1, 2));
    }
}
```

PHP



# Test-driven development (TDD)

## 2. Garantir que o novo teste falha

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

```
PHP Fatal error: Call to undefined method  
    TFurtado\UnitTestSample\Calculator::sum()  
    in .../test/CalculatorTest.php on line 20
```

```
PHP Stack trace:
```

```
PHP    1. {main}() ./vendor/phpunit/phpunit/phpunit:0  
    ...
```



# Test-driven development (TDD)

## 3. Escrever código

```
class Calculator
{
    public function sum($a, $b)
    {
        $sum = 3;
        return $sum;
    }
}
```

PHP



# Test-driven development (TDD)

## 4. Garantir que os testes passam

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

```
.
```

Time: 77 ms, Memory: 2.75Mb

OK (1 test, 1 assertion)



# Test-driven development (TDD)

## 5. Refatorar o código

```
class Calculator
{
    public function sum($a, $b)
    {
        $sum = 3;
        return $sum;
        return 3;
    }
}
```

PHP





# Test-driven development (TDD)

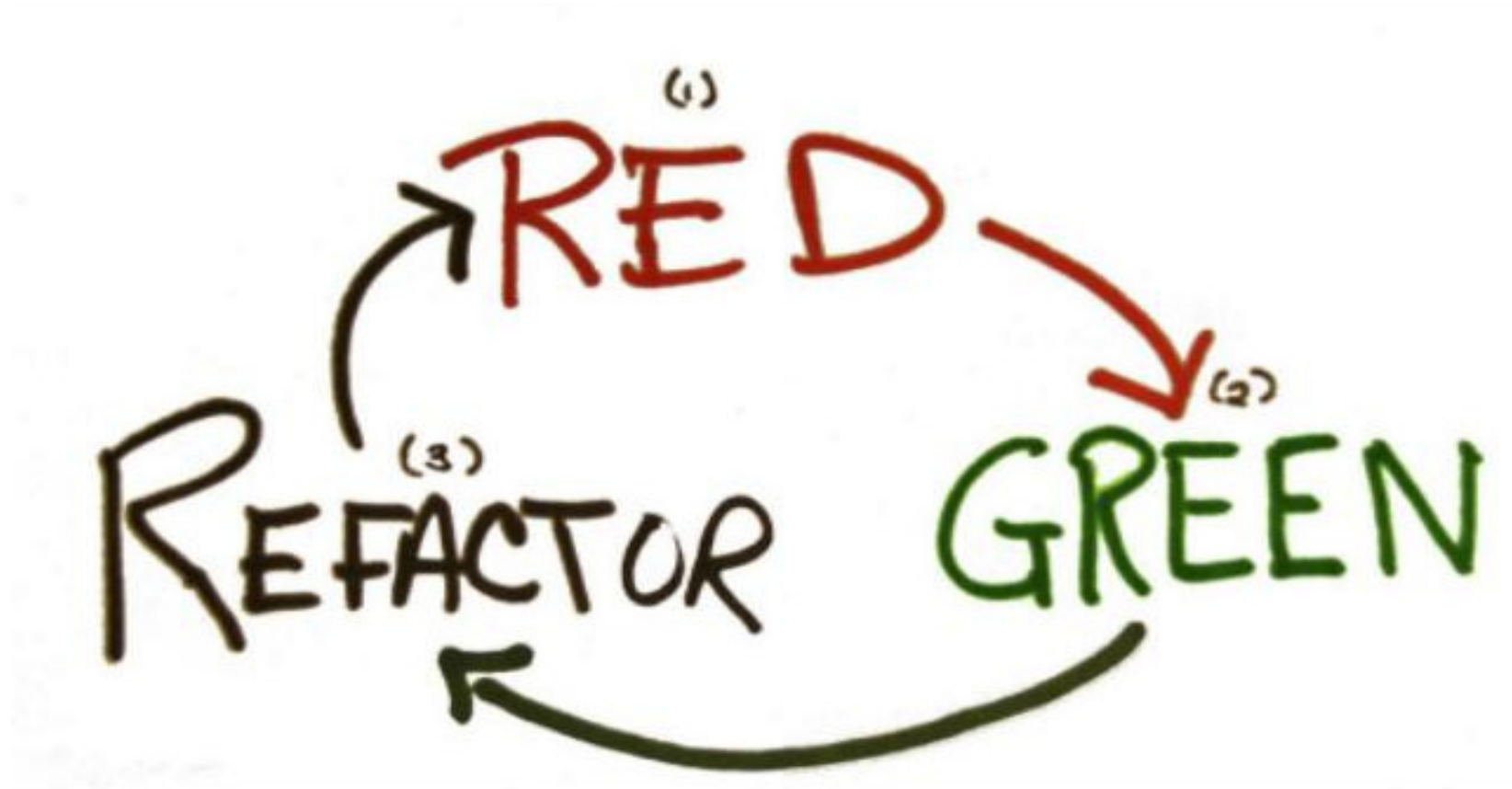
## 5. Refatorar o código

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.  
  
.  
  
Time: 87 ms, Memory: 2.75Mb  
  
OK (1 test, 1 assertion)
```



# Test-driven development (TDD)

Repetir...



# Test-driven development (TDD)

## 1. Escrever um teste

```
class CalculatorTest extends PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function sumTest()
    {
        $calculator = new Calculator();
        $this->assertSame(3, $calculator->sum(1, 2));
        $this->assertSame(4, $calculator->sum(2, 2));
    }
}
```

PHP



# Test-driven development (TDD)

## 2. Garantir que o novo teste falha

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

F

Time: 76 ms, Memory: 3.00Mb

There was 1 failure:

1) TFurtado\UnitTestSample\CalculatorTest::sumTest  
Failed asserting that 3 is identical to 4.

.../test/CalculatorTest.php:21

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.



# Test-driven development (TDD)

## 3. Escrever código

```
class Calculator
{
    public function sum($a, $b)
    {
        return 3;
        $sum = $a + $b;
        return $sum;
    }
}
```

PHP



# Test-driven development (TDD)

## 4. Garantir que os testes passam

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php
    test/CalculatorTest.php
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.

.

Time: 90 ms, Memory: 2.75Mb

OK (1 test, 2 assertions)
```



# Test-driven development (TDD)

## 5. Refatorar o código

```
class Calculator
{
    public function sum($a, $b)
    {
        $sum = $a + $b;
        return $sum;
        return $a + $b;
    }
}
```

PHP



# Test-driven development (TDD)

## 5. Refatorar o código

```
$ ./vendor/bin/phpunit --bootstrap ./vendor/autoload.php  
    test/CalculatorTest.php  
PHPUnit 4.5-gf75e6b2 by Sebastian Bergmann.
```

```
.
```

Time: 87 ms, Memory: 2.75Mb

OK (1 test, 1 assertion)





# Test-driven development (TDD)

Repetir...





# Cobertura de testes

# Quanto do software deve estar coberto por testes de unidade?

100%!

TODO!

O software INTEIRO!

Deu pra entender, né?



Como garantir 100% de cobertura de testes?







# Testes de unidade na Meritt

# Testes de unidade na Meritt

## Status

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total	<div></div>	100.00%	1040 / 1040	<div></div>	100.00%	378 / 378	<div></div>	100.00% 63 / 63
 Domain	<div></div>	100.00%	589 / 589	<div></div>	100.00%	246 / 246	<div></div>	100.00% 22 / 22
 Infra	<div></div>	100.00%	451 / 451	<div></div>	100.00%	132 / 132	<div></div>	100.00% 41 / 41

QEdu > test-ci:

```
..... 63 / 571 ( 11%)
..... 126 / 571 ( 22%)
..... 189 / 571 ( 33%)
..... 252 / 571 ( 44%)
..... 315 / 571 ( 55%)
..... 378 / 571 ( 66%)
..... 441 / 571 ( 77%)
..... 504 / 571 ( 88%)
..... 567 / 571 ( 99%)
.....
```




Time: 29.61 seconds, Memory: 32.00Mb

OK (571 tests, 793 assertions)



# Testes de unidade na Meritt

## Status

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	<div><div></div></div>	76.93%	2237 / 2908	<div><div></div></div>	80.99%	443 / 547	<div><div></div></div>	80.67%	121 / 150
 Application	<div><div></div></div>	88.21%	756 / 857	<div><div></div></div>	93.16%	109 / 117	<div><div></div></div>	87.88%	29 / 33
 Infra	<div><div></div></div>	100.00%	175 / 175	<div><div></div></div>	100.00%	42 / 42	<div><div></div></div>	100.00%	12 / 12
 UI	<div><div></div></div>	69.62%	1306 / 1876	<div><div></div></div>	75.26%	292 / 388	<div><div></div></div>	76.19%	80 / 105

QEdu > test-ci:

```
..... 63 / 627 ( 10%)
..... 126 / 627 ( 20%)
..... 189 / 627 ( 30%)
..... 252 / 627 ( 40%)
..... 315 / 627 ( 50%)
..... 378 / 627 ( 60%)
..... 441 / 627 ( 70%)
..... 504 / 627 ( 80%)
..... 567 / 627 ( 90%)
.....
```

Time: 34.99 seconds, Memory: 42.25Mb

OK (627 tests, 1076 assertions)







# Considerações finais

# Considerações finais

Testes de unidade são...

- ... essenciais para garantir a manutenibilidade do software
- ... instrumentos de simples implementação
- ... muito valiosos para criar um produto de qualidade
- ... uma importante forma de documentação técnica
- ... divertidos! Por que não?



# Considerações finais

Testes de unidade **não** são...

- ... a abordagem definitiva para testes de software
- ... uma garantia de qualidade do produto final
- ... a solução mágica para software mal arquitetado



# Temos vagas!

## Desenvolvedor(a) PHP

Você tem experiência com desenvolvimento PHP, gosta de trabalhar com bancos de dados, não tem medo da linha de comando e faz mais commits do que toma água?

Estamos procurando por você!  
[meritt.com.br/jobs](http://meritt.com.br/jobs)



## Front-end UI

Você adora desenvolvimento web? Gosta muito de design e sempre pensa em como transformar telas em sistemas amigáveis, bonitos e incríveis?

Estamos procurando por você!  
[meritt.com.br/jobs](http://meritt.com.br/jobs)



## Back-end Engineer PHP

Você tem vasta experiência em desenvolvimento back-end, seus sonhos seguem design patterns e você escreve testes até para seus posts de facebook?

Estamos procurando por você!  
[meritt.com.br/jobs](http://meritt.com.br/jobs)



## Front-end Engineer

Você é uma pessoa apaixonada por HTML5, Javascript, arquiteturas MV\*, testes unitários e gostaria que o mundo fosse reescrito com uma API REST através de um Single Page App?

Estamos procurando por você!  
[meritt.com.br/jobs](http://meritt.com.br/jobs)



[meritt.recruiterbox.com](http://meritt.recruiterbox.com)

“Pay attention to zeros. If there is a zero,  
someone will divide by it.”

Cem Kaner





# Muito obrigado!

tfurtado@gmail.com  
speakerdeck.com/tfurtado  
github.com/tfurtado  
meritt.com.br

