Behaviour-Driven Development (BDD) em Python

Hugo Lopes Tavares

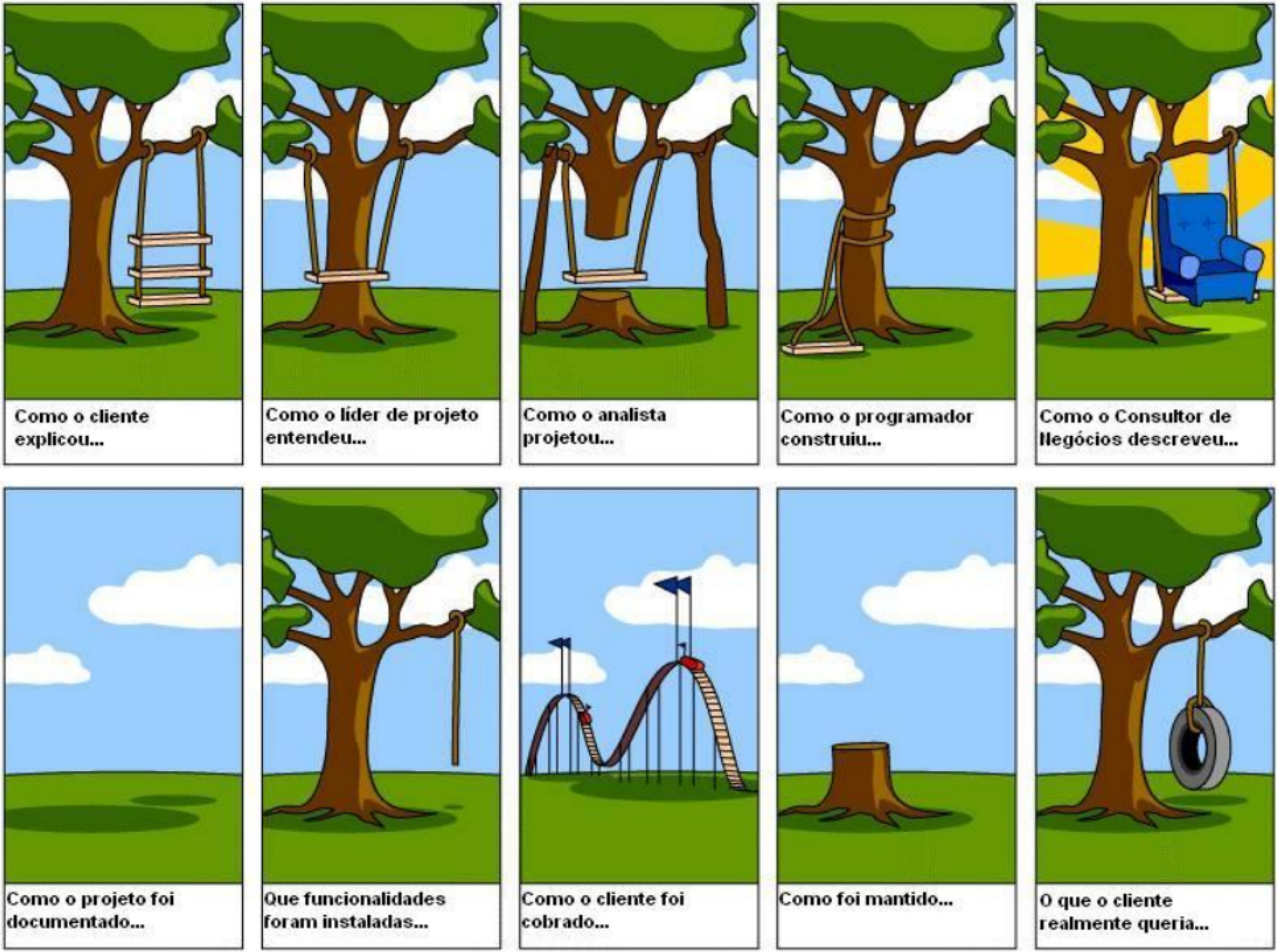

NÃO ENCHE O SACO
TO
DE FÉRIAS!
NATAL - RN / BRASIL
the Simpsons
Red Nose
PYTHON
Powered

Parte 1: Conceitos

TestDox

```
Public class FooTest extends TestCase {
    public void testIsASingleton() {}
    public void testAReallyLongNameIsAGoodThing() {}
}
```

Foo
- is a singleton
- a really long name is a good thing


Como o cliente explicou...
Como o líder de projeto entendeu...
Como o analista projetou...
Como o programador construiu...
Como o Consultor de Negócios descreveu...
Como o projeto foi documentado...
Que funcionalidades foram instaladas...
Como o cliente foi cobrado...
Como foi mantido...
O que o cliente realmente queria...


palavras
pensamentos

“Getting the words right”

Palavra mágica:

should

```
public class CustomerLookupTest {
    @Test
    public void shouldFindCustomerById() { }
    @Test
    public void shouldFailForDuplicatedCustomers() { }
}
```

CustomerLookup

- should find customer by id

- should fail for duplicated customers

Ubiquitous Language para análise de software

Histórias
e
Cenários

História: Calculadora

Como uma criança inocente

Eu quero ter uma calculadora

Para que eu aprenda a fazer

operações matemáticas

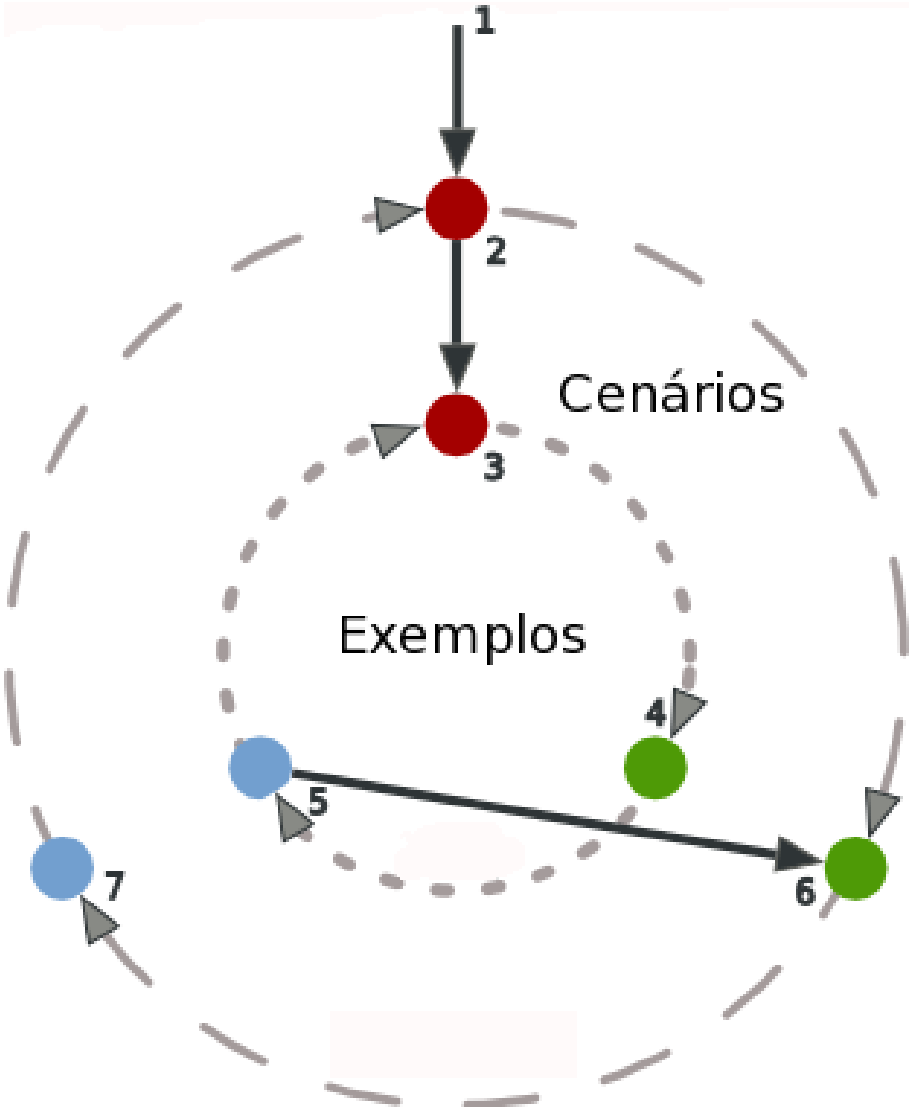
Cenário: Soma
Dado que eu tenho uma calculadora
Quando eu entro com 1 e 2
E aperto =
Então eu vejo 3 na tela

“It's all behaviour”

histórias e cenários → **aplicação**

exemplos → **código**

Outside-In Development



O que precisamos sempre manter em mente?


Specification,
not Verification

todo comportamento precisa de exemplos


WORLD'S
BEST
BOSS
EXIT

Benefícios

foco


Backspace
Insert
Delete
End
RUN!
Shift

documentação **correta**
em sincronia com o
código

Cenários:

testes de aceitação
e de regressão

Exemplos:

testes unitários

e

documentação

YAGNI

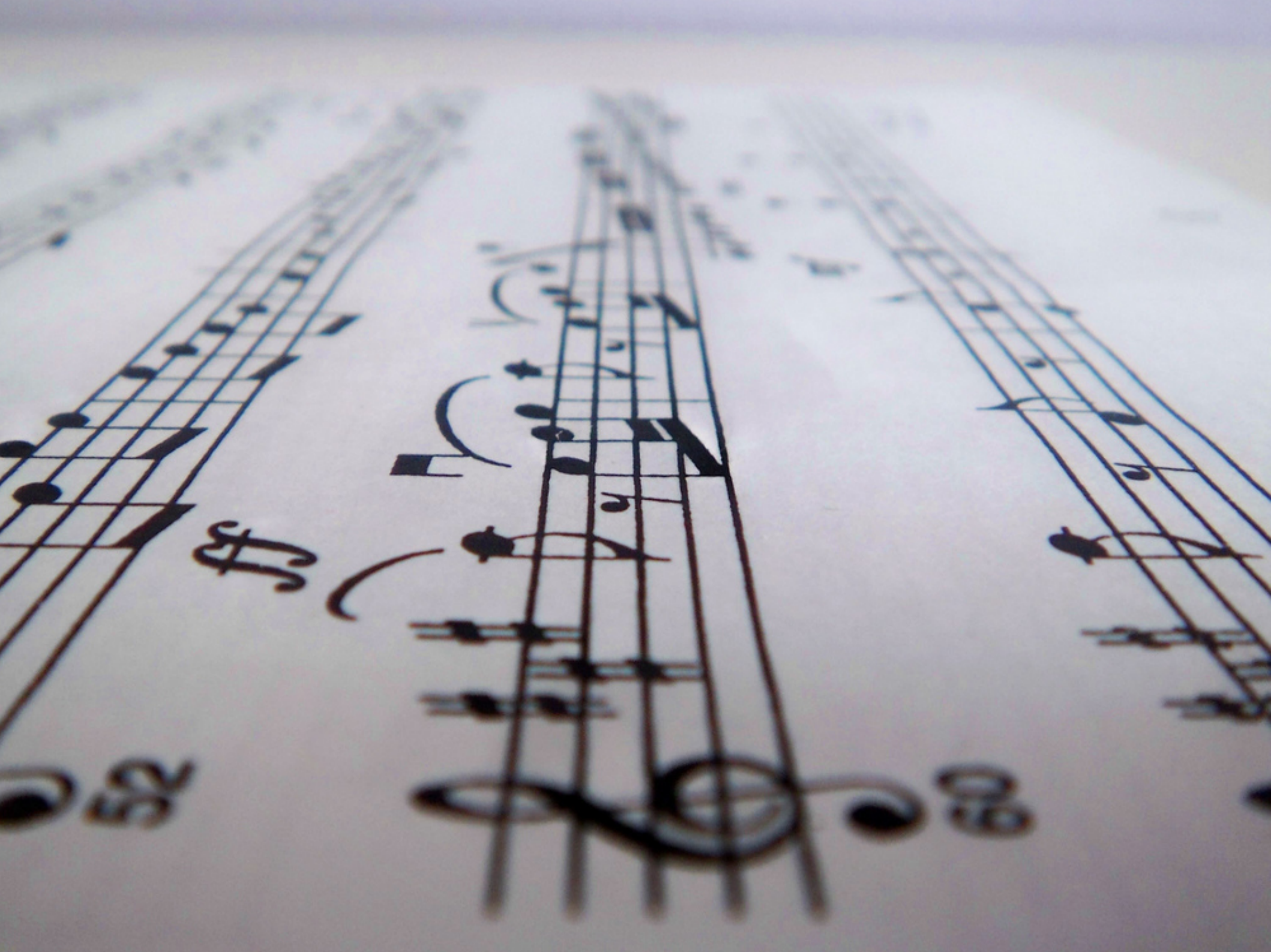

Mock
Objects

Como fazer BDD no dia-a-dia?


Não Iniciado
Em Andamento
Finalizado
10 Anos
Congresso Fluminense de
Iniciação Científica e Tecnológica

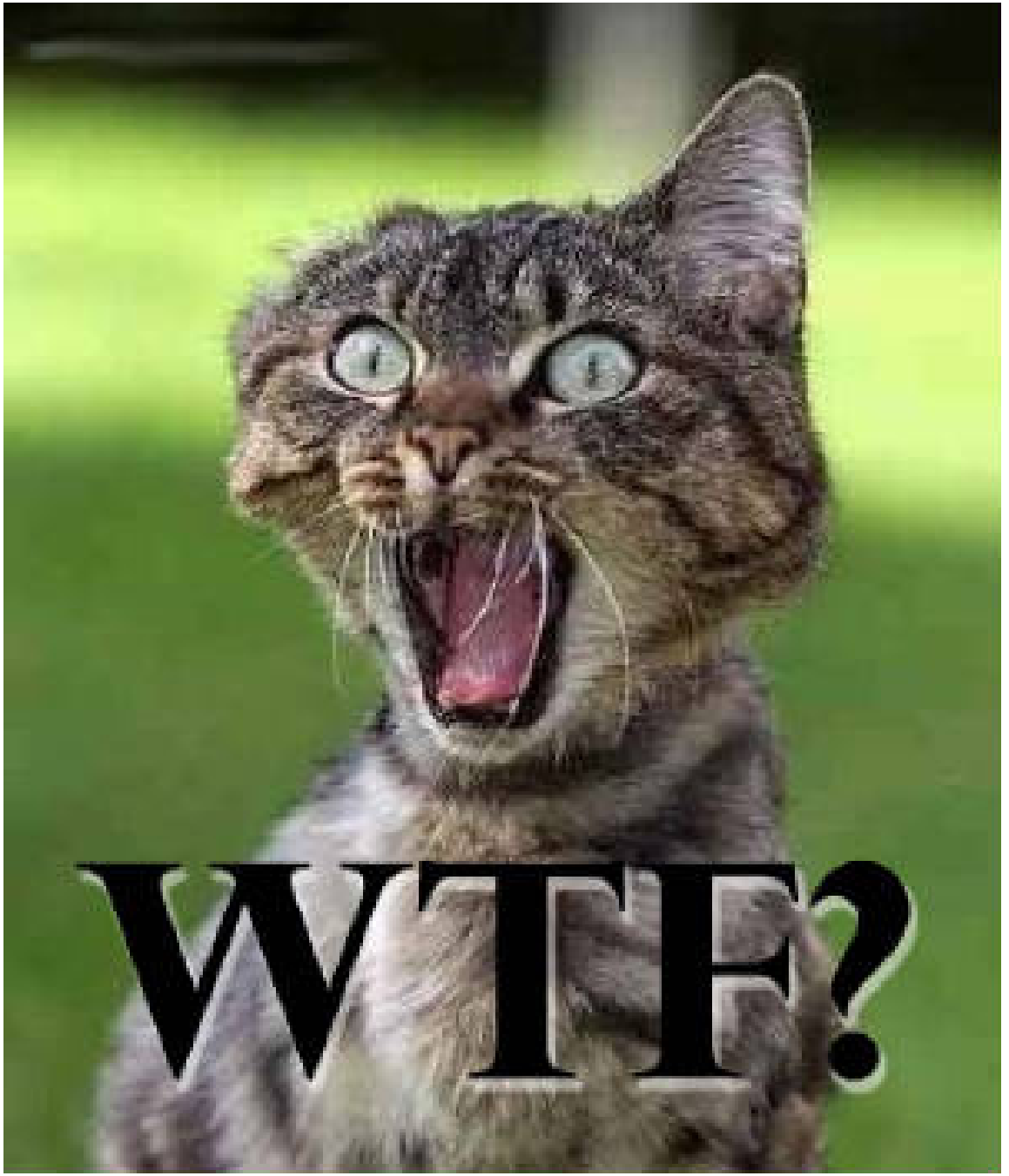

python brasil [5]

Todas as práticas em BDD são novas?



Beer-Driven Development


SPATEN


WTF?

The Pragmatic Programmers

The RSpec Book

Behaviour Driven Development with RSpec, Cucumber, and Friends

David Chelimsky
with Dave Astels, Zach Dennis, Aslak Hellesøy, Bryan Helmkamp, and Dan North

Edited by Jacquelyn Carter

The Facets of Ruby Series

Parte 2: Ferramentas em Python

Mock Frameworks

Mockito

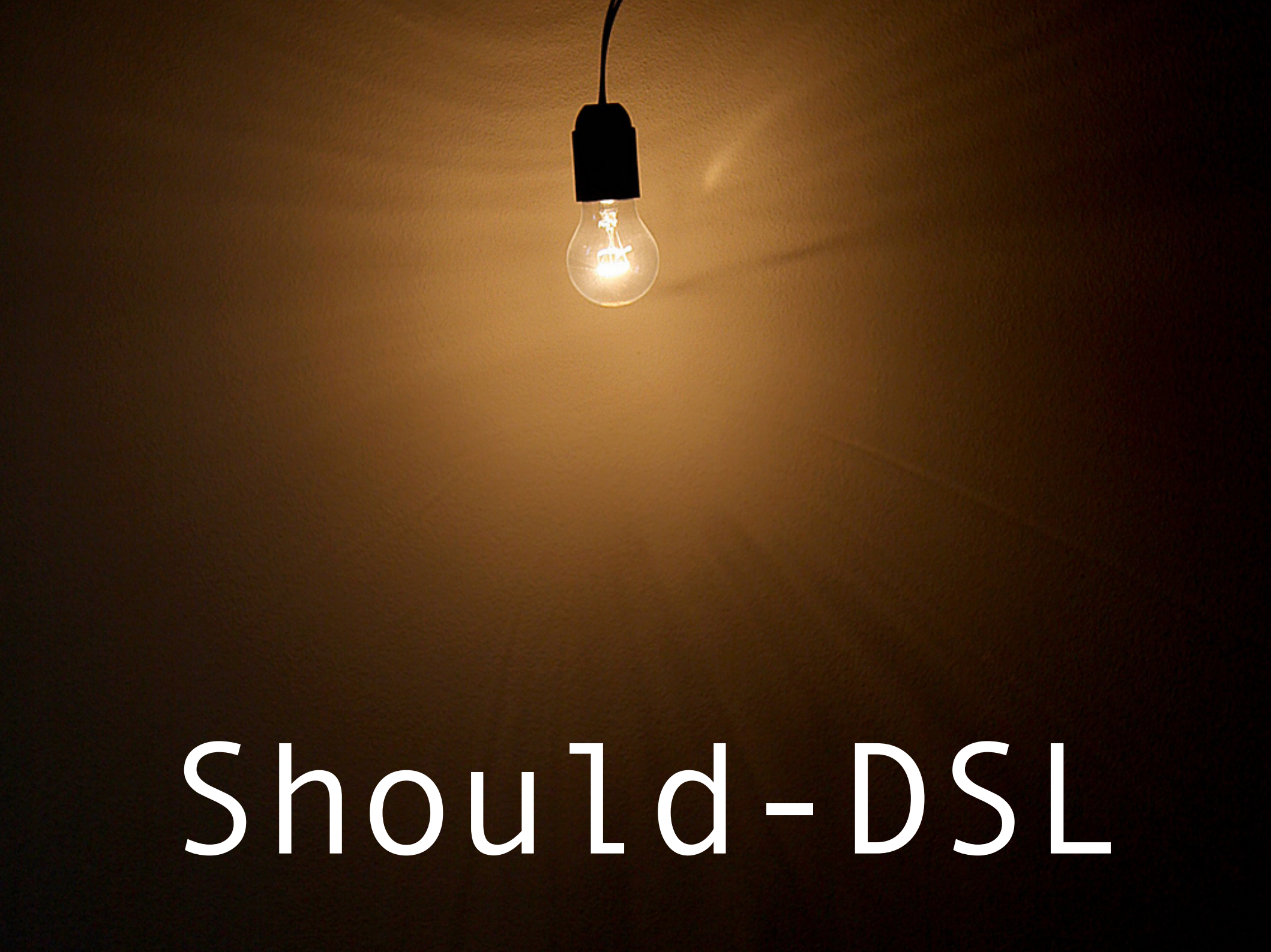
```
from mockito import Mock, when, verify

class MaquinaDeSaque(object):
    @staticmethod
    def transferir(valor, conta1, conta2):
        if conta2.pegar_saldo() >= valor:
            conta2.sacar(valor)
            conta1.depositar(valor)

conta2 = Mock()
conta1 = Mock()
when(conta2).pegar_saldo().thenReturn(200)

MaquinaDeSaque.transferir(100, conta1, conta2)

verify(conta1).depositar(100)
verify(conta2).sacar(100)
```

Ludíbrio

```
from __future__ import with_statement
from ludibrio import Mock

class MaquinaDeSaque(object):
    @staticmethod
    def transferir(valor, conta1, conta2):
        if conta2.pegar_saldo() >= valor:
            conta2.sacar(valor)
            conta1.depositar(valor)

with Mock() as conta1:
    conta1.depositar(100)

with Mock() as conta2:
    conta2.pegar_saldo() << 200
    conta2.sacar(100)

MaquinaDeSaque.transferir(100, conta1, conta2)
```

http://bitbucket.org/szczepiq/mockito-python

http://github.com/nsigustavo/ludibrio


Should-DSL

```
2 |should_be.greater_than| 1

'hugo' |should_not_be.equal_to| 'Hugo'

5 |should_be.less_than| 10

['python', 'C'] |should_have| 'python'

'C' |should_be.into| ['python', 'C']
```

```
assert 2 > 1

assert 'hugo' == 'Hugo'

assert 5 < 10

assert 'C' in '[python', 'C']
```

```
>>> from should_dsl import matcher, should_be

>>> @matcher
... def abs_of():
...     return lambda x, y: x == abs(y),\
...                 '%s is%s the abs of %s'
...

>>> 1 |should_be.abs_of| -1
True
```

```
>>> from math import sqrt

>>> deve_ser = should_be

>>> @matcher
... def a_raiz_quadrada_de():
...     return lambda x,y: x == sqrt(y),\
...                "%s %se' a raiz quadrada de %s"
...

>>> 2 |deve_ser.a_raiz_quadrada_de| 4
True
```

http://github.com/hugobr/should-dsl

Example Frameworks


XUnit ?

PySpec

behaviour_spec.py:

```
import stack          # import production code
from pyspec import * # import pyspec framework

class Behavior_Stack(object):
  @context
  def new_stack(self):
    self.stack = stack.Stack()

  @spec
  def should_be_empty(self):
    About(self.stack).should_be_empty()

if __name__ == "__main__":
  run_test()
```

stack.py:

```
class Stack(object):
  def __len__(self):
    return 0
```

```
$ python behavior_stack.py

new stack
    should be empty ... OK

----------------------------------
Ran 1 spec in 0.048s
OK
```

Behaviour

```
class verifyUserSpecification(behaviour.Behaviour):

    def setUp(self):
        self.user = User("Mark Dancer")

    def verifyInitialUserNameIsNameInConstructor(self):
        self.shouldBeEqual(self.user.name, "Mark Dancer")

    def verifyInitialUserHasNoLanguages(self):
        self.shouldBeEmpty(self.user.languages)
```

Yeti

```
# coding: spec
from should_dsl import should_be

class Bowling:
    score = 0
    def hit(self, v):
        self.score += v

describe Bowling:
    def before_each:
        self._bowling = Bowling()

    it "should score 0 for gutter game":
        self._bowling.hit(0)
        self._bowling.score |should_be.equal_to| 0

    it "should score 1 for just one hit":
        self._bowling.hit(1)
        self._bowling.score |should_be.equal_to| 1
```

http://www.codeplex.com/pyspec

http://pypi.python.org/pypi/Behaviour

http://github.com/fmeyer/yeti

Story/Feature Frameworks

Pyccuracy

```
As a Google User
I want to search Google
So that I can test Pyccuracy

Scenario 1 - Searching for Hello World
Given
  I go to "http://www.google.com"
When
  I fill "q" textbox with "Hello World"
  And I click "btnG" button
  And I wait for 2 seconds
Then
  I see "Hello World - Pesquisa Google" title
```

Pyhistorian

```
from should_dsl import *
from pyhistorian import *

class Calculator(object):
    def sum(self, n1, n2):
        return n1+n2


class SpecifyingMyNewCalculator(Story):
    """As a lazy mathematician
       I want to use a calculator
       So that I don't waste my time thinking"""
    colored = True
    template_color = 'yellow'
    scenarios = ['SumScenario'] # optional
```

Pyhistorian

```
class SumScenario(Scenario):
    @Given('I have a calculator')
    def set_my_calculator(self):
        self.calculator = Calculator()

    @When('I enter with 1 + 1')
    def sum_one_to_one(self):
        self.result = self.calculator.sum(1, 1)

    @Then('I have $value as result', 2)
    def get_result(self, value):
        self.result |should_be| value


if __name__ == '__main__':
    SpecifyingMyNewCalculator.run()
```

PyCukes

```
Story: Bowling Game
  As a bowling player
  I want to play bowling online
  So that I can play with everyone in the world

    Scenario 1: Gutter Game
      Given I am playing a bowling game
      When I hit no balls
      Then I have 0 points
```

PyCukes

```
from pycukes import *

class BowlingGame(object):
    score = 1
    def hit(self, balls):
        pass


@Given('I am playing a bowling game')
def start_game(context):
    context._bowling_game = BowlingGame()

@When('I hit no balls')
def hit_no_balls(context):
    context._bowling_game.hit(0)

@Then('I have 0 points')
def i_have_zero_points(context):
    assert context._bowling_game.score == 0
```

Freshen

```
Feature: Division
  In order to avoid silly mistakes
  Cashiers must be able to calculate a fraction

  Scenario: Regular numbers
    Given I have entered 3 into the calculator
    And I have entered 2 into the calculator
    When I press divide
    Then the result should be 1.5 on the screen
```

Freshen

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario Outline: Add two numbers
    Given I have entered <input_1> into the calculator
    And I have entered <input_2> into the calculator
    When I press <button>
    Then the result should be <output> on the screen

  Examples:
    | input_1 | input_2 | button | output |
    | 20      | 30      | add    | 50     |
    | 2       | 5       | add    | 7      |
    | 0       | 40      | add    | 40     |
```

Freshen

```
from freshen import *
from freshen.checks import *

import calculator

@Before
def before(sc):
    scc.calc = calculator.Calculator()
    scc.result = None

@Given("I have entered (\d+) into the calculator")
def enter(num):
    scc.calc.push(int(num))

@When("I press (\w+)")
def press(button):
    op = getattr(scc.calc, button)
    scc.result = op()

@Then("the result should be (.*) on the screen")
def check_result(value):
    assert_equal(str(scc.result), value)
```

http://github.com/heynemann/pyccuracy

http://github.com/hugobr/pyhistorian

http://github.com/hugobr/pycukes

http://github.com/rlisagor/freshen

Referências

http://dannorth.net/introducing-bdd

http://blog.daveastels.com/files/BDD_Intro.pdf

http://behaviour-driven.org/

http://pt.wikipedia.org/wiki/Behaviour_Driven_Development

http://en.wikipedia.org/wiki/Behaviour_Driven_Development

http://agiledox.sourceforge.net/

http://cukes.info

http://rspec.info

Obrigado!

hltbra@gmail.com
@hltbra
http://hugolt.wordpress.com