



<http://cukes.info/>

Táise Dias da Silva
taisedias@gmail.com

Agenda

- Por que Cucumber?
- Estrutura do Cucumber (testing stack)
- Feature, Scenario & Step Definitions
- Melhores práticas
- Referências



Por que Cucumber?

Software - Problemas

- Software: o que pode acontecer de errado?
 - Defeito de implementação
 - Programa não faz o que pretendia
 - Corrigido em desenvolvimento/QA
 - Defeito no requisito
 - Programa faz o que pretendia
 - O requisito não foi compreendido corretamente

Software - Problemas

- 50-60% dos problemas encontrados por testadores são causados por problemas nos requisitos
- 100-200% mais caro para se corrigir do que os outros defeitos porque o código já estará escrito

Como Cucumber pode ajudar?

- Cucumber é uma ferramenta baseada em Behavior Driven Development (BDD)
- Stakeholders focam em especificação baseada no valor do negócio
- Especificações escritas em linguagem natural (qualquer um consegue ler)

Como Cucumber pode ajudar?

- Especificações viram testes de aceitação, descrevendo exemplos do comportamento do software antes da implementação (feedback cedo)
- Testes são usados na regressão durante a evolução do software

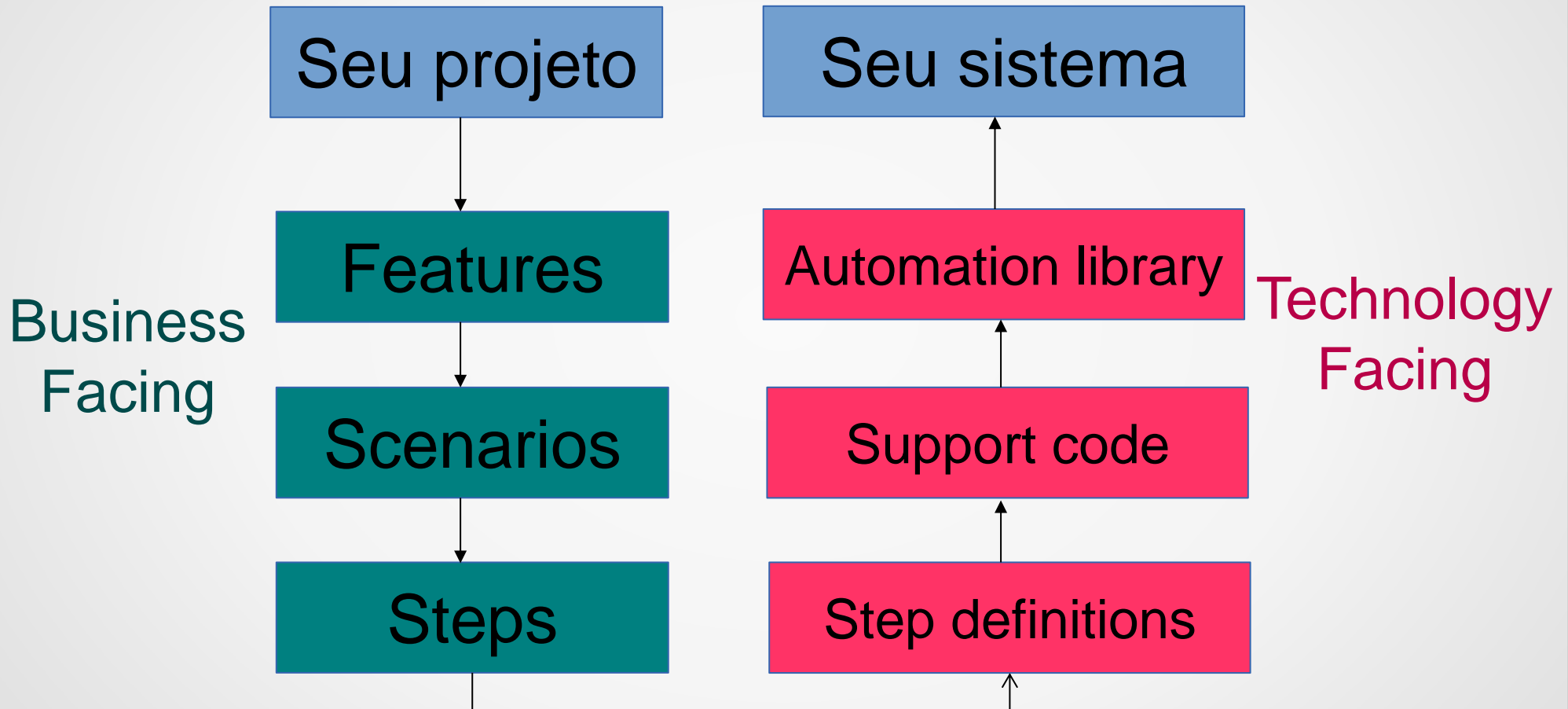
O que é Cucumber?

- **Cucumber** é uma **ferramenta** que executa descrições funcionais em **texto puro** como **testes automatizados**
- Texto puro: escrito em Gherkin
- Automação: escrita em alguma linguagem de programação



Cucumber Testing Stack

Cucumber testing stack





Features, Scenarios & Step Definitions

Feature

- Testes em Cucumber são agrupados em **features**

```
Feature: Acess app Dinosaurs  
<<description>>
```

- Cada teste em Cucumber chama-se **scenario**

```
Scenario: List dinos names
```

Scenario & Steps

- Cada *scenario* contém *steps* que diz ao Cucumber o **que** fazer

```
Given I am in Dino app  
When I choose List  
Then I see the list of dinosaurs  
And I should see filter message in  
the screen
```

Step Definitions

- Para que Cucumber saiba **como** executar um *scenario* de uma *feature*, precisamos escrever *step definition*.

```
Given(/^I am in Dino app$/) do
  pending # express the regexp above
  with the code you wish you had
end
```

Step Definitions

- Para implementar *step definition*:
 - Substitua comentário por código
 - Código delega para o ***support code***, específico do domínio da sua app
 - O support code usa uma ***automation library*** para interagir com o sistema.

```
Given(/^I am in Dino app$/) do
  @page = Index.new
  @page.load
  @diff_group = ""
  @filter_value = ""
end
```

Step Definitions

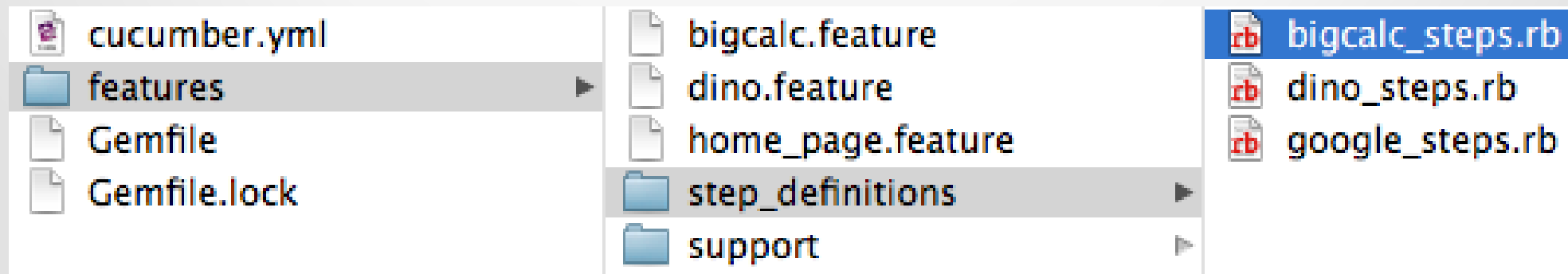
```
When (/^I choose List$/) do
  @page.list_link.click
end

Then (/^I see the list of dinosaurs$/) do
  @page.should have_dino_list
  @page.should_not have_dino_image
end

And (/^I should see filter message in the
screen$/) do
  @page.dino_results.text.should ==
  "Results for \""+@filter_value+"\"
  and DD Group \""+@diff_group+"\"
end
```


Arquivos & Diretórios

- Features
 - Cada feature é salva como arquivo `.feature`
 - Features são salvas no diretório *features*
- Step definitions
 - step definitions são salvos com extensão `.rb`
 - Step definitions são salvos no diretório *step_definitions*





Melhores Práticas

Features Declarativas

Feature: Dinosaurs app access

Scenario: List dinos names

Given I go to the home page

When I click the link "List"

Then the page should contain a list of Dinosaurs



Feature: Dinosaurs app access

Scenario: List dinos names

Given I am in Dino app

When I list dinos names

Then I see a list of Dinosaurs



Features Narrativas

Feature: Dinosaurs app access

Scenario: List dinos names

Given I am in Dino app

When I choose List

Then I see list of Dinosaurs



Feature: Dinosaurs app access

In order to identify dinosaurs

As a bones keeper

I want to access information about dinosaurs

Scenario: List dinos names

Given I am in Dino app

When I choose List

Then I see list of Dinosaurs




Evitar steps conjuntivos

```
# Scenario 1
...
When I compose an email to "john@john.com" and send it
...
# step definition
When(/^I compose an email to "(.*)" and send it$/) do |email_address|
  email = Email.new recipient: email_address
  email.send
end
```



```
# Scenario 2
...
When I compose an email to "john@john.com"
And I add "mary@mary.com" as recipient
And I send the email
...
# step definition
When(/^I compose an email to "(.*)"$/) do |email_address|
  @email = Email.new recipient: email_address
end
When(/^I send an email$/) do
  @email.send
end
```



Reusar step definitions

```
# feature
...
When I compose an informal email to "friend@gap.com"
And I send an email
...
# step definition
When(/^I compose an email to "(.*)" and send it$/) do
  |email_address|
    @email = Email.new recipient: email_address
end
...
When(/^I compose an informal email to "(.*)"$/) do
  |email_address|
    Step %[I compose an e-mail to "#{email_address}"]
    @email.set_greeting "Ahoi mate"
end
```



Não abusar do uso do Background

Feature: Using the background

In order to ...

Background:

Given I am signed up as "qa@it.com"

Scenario: Accessing a build

When I go to the dashboard

Then I should be able to access the
build

Scenario: Restarting a build

When I go to the dashboard

Then I should be able to restart the
build



Não abusar do uso do Background

Feature: Using the background

In order to ...

Background:

Given I am signed up as "qa@it.com"

When I go to the dashboard

Scenario: Acessing a build

Then I should be able to access the
build

Scenario: Restarting a build

Then I should be able to restart the
build





Resumo

Resumo

- Cucumber ajuda o time de desenvolvimento de software a:
 - Compreender corretamente os requisitos através de exemplos
 - Usar o mesmo vocabulário para fala sobre o software
 - Ter uma documentação viva (executável) e de fácil leitura em linguagem natural

Resumo

- Estrutura do Cucumber:
 - Business facing: Features, Scenarios & Steps
 - Technology Facing: Step Definitions, Support Code & Automation Library

Resumo

- Melhores práticas:
 - Escrever features declarativas
 - Inserir narrative nas features
 - Evitar steps conjuntivos
 - Reusar step definitions
 - Não abusar do uso de backgrounds

References

- Test automation with Cucumber-JVM:
http://pt.slideshare.net/alan_parkinson/test-automation-with-cucumberjvm
- Code Centric:
<https://blog.codecentric.de/en/2013/08/cucumber-setup-basics/>
- Code Centric:
<http://blog.codeship.io/2013/05/21/Testing-Tuesday-6-Top-5-Cucumber-best-practices.html>
- The Cucumber Book: Behaviour-Driven Development for Testers and Developers, by Matt Wynne and Aslak Hellesøy