



Apaixonados por Eficiência

in|metrics



Automação com BDD em Ruby + Capybara

DOJO NIVEL I



in|metrics

Agenda

1

*Dinâmica
Dojo*

2

*Conhecendo
o BDD*

3

*Cucumber
x
Capybara*

4

*Arquitetura
do projeto*

5

*Vamos
Praticar?*

in |

1

Dinâmica Dojo

in

Dojo (pronuncia-se Dojô)

Valores

- Cooperação
- Participação
- Coragem
- Respeito
- Simplicidade

Objetivo

- Praticar
- Aprender
- Ensinar
- Discutir com base no código

É uma palavra de origem japonesa e significa “local de treinamento”. Portanto, o **Coding Dojo** nada mais é que do um “local de treinamento de código”, ou “local de treinamento de programação”.



2

Conhecendo o BDD

in



Como o cliente explicou...



Como o líder de projeto entendeu...



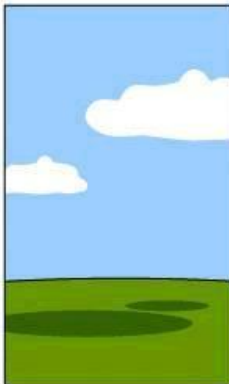
Como o analista projetou...



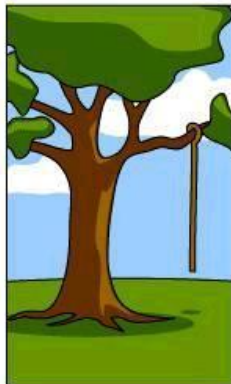
Como o programador construiu...



Como o Consultor de Negócios descreveu...



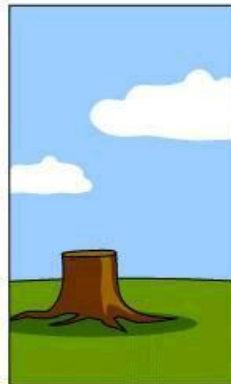
Como o projeto foi documentado...



Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...

Técnica de desenvolvimento **Ágil** que encoraja **colaboração** entre desenvolvedores, setores de qualidade e pessoas não-técnicas ou de negócios num projeto de software.

BDD

Tradução BDD: **Desenvolvimento Guiado por Comportamento**

Traduz o comportamento do usuário da funcionalidade para cada cenário com seu respectivo resultado.

Não é criado para o teste e sim para valor de negócio, documentação e comunicação.

É possível ser criada antes mesmo do desenvolvimento da funcionalidade.



Representa os critérios de aceite de uma funcionalidade.

Criando uma funcionalidade

Funcionalidade: É um comportamento ou ação que o sistema oferece ao seu usuário baseada em uma entrada e uma saída, que contém um ou mais cenários diferentes. O nome é determinado pela combinação de um verbo e um substantivo. Exemplo:

Funcionalidade: Buscar Agências

Eu como cliente do banco

Quero procurar uma agência dentro do Brasil

Para saber suas informações de contato

Criando um cenário

Cenário: Descreve um conjunto ordenado de comportamentos baseado em uma entrada para alcançar um resultado específico de uma funcionalidade.

As palavras chaves do cenário (**Dado** / **Quando** / **Então**) são base para o funcionamento do teste.

Cenário: Buscar agência por CEP válido

Dado que esteja na home do site do banco

Quando buscar uma agência pelo CEP

Então apresentará as agências disponíveis

Exemplo de uma Funcionalidade Cucumber

```
Feature: Viewer visits the home page
  In order to read the page
  As a viewer
  I want to see the home page of my app

Scenario: View home page # features/home
  Given I am on the home page # features/step
  Then I should see "This is the home page." # features/step

Scenario: Find heading on home page # features
  Given I am on the home page # features
  Then I should see "MY APP" in the selector "h1" # features

Scenario: Find the link to the form
  Given I am on the home page
  Then I should see "Sign up for our newsletter." in a link
```

```
#language: pt
#encode: UTF-8
```

Funcionalidade: Cadastro de usuario

Eu como novo usuario
Desejo me cadastrar no Site

Cenario: Cadastrar usuario

Dado que eu esteja na tela de cadastro
E efetue o cadastro com dados validos
Então o cadastro efetuado com sucesso

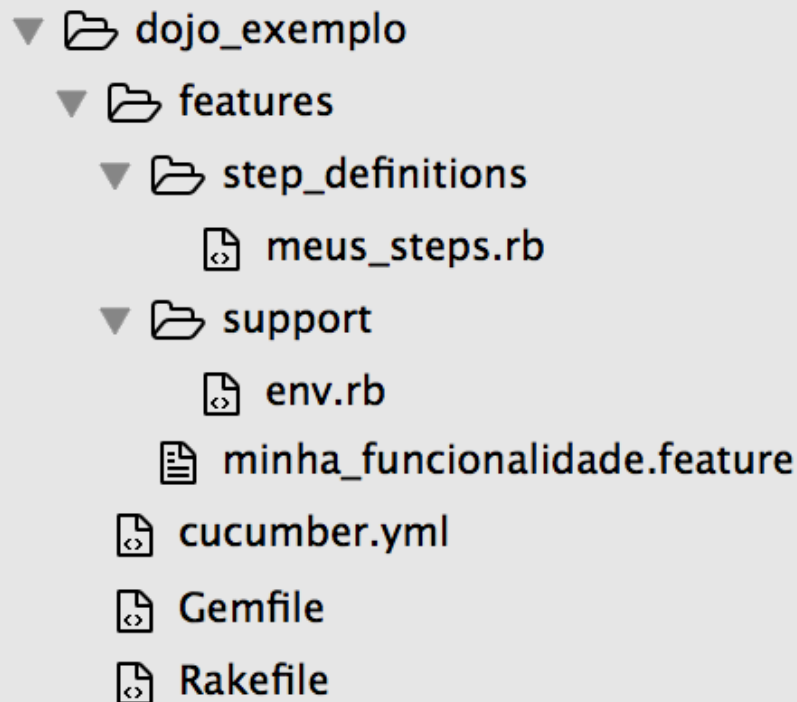
3

Arquitetura do Projeto


in










Arquitetura do projeto

Entendendo a Estrutura Básica do Projeto



```
graph TD; dojo_exemplo[dojo_exemplo] --> features[features]; dojo_exemplo --> support[support]; dojo_exemplo --> cucumber[cucumber.yml]; dojo_exemplo --> gemfile[Gemfile]; dojo_exemplo --> rakefile[Rakefile]; features --> step_definitions[step_definitions]; step_definitions --> meus_steps[meus_steps.rb]; support --> env[env.rb]; support --> minha_funcionalidade[minha_funcionalidade.feature];
```

▼  dojo_exemplo

- ▼  features
 - ▼  step_definitions
 -  meus_steps.rb
 - ▼  support
 -  env.rb
 -  minha_funcionalidade.feature
-  cucumber.yml
-  Gemfile
-  Rakefile

Arquitetura do projeto

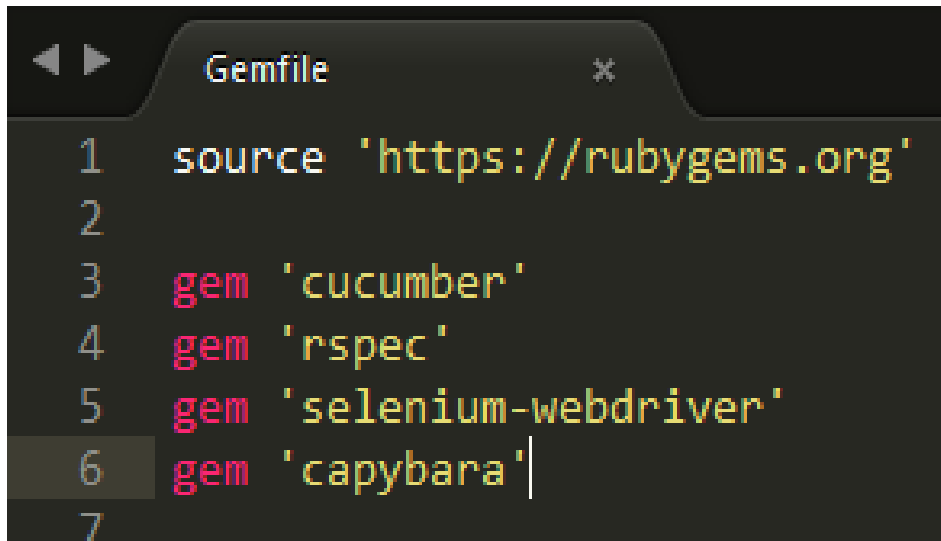
Montando seu ambiente de testes

```
c:\Projects
λ testgen project Dojo
  create Dojo
  create Dojo/cucumber.yml
  create Dojo/Gemfile
  create Dojo/Rakefile
  create Dojo/features
  create Dojo/features/support
  create Dojo/features/step_definitions
  create Dojo/features/support/env.rb
```

- No terminal, instale o gerador de estrutura do projeto para automação:
 - **gem install testgen**
- Crie o projeto de automação: Acesso ao diretório onde o projeto deverá ser criado.
 - **testgen project dojo_ddmmaa**
 - **cd dojo_ddmmaa**
 - **Abrir o Projeto criado na ferramenta Sublime**

Arquitetura do projeto - GEMFILE

O que é gem?

A screenshot of a code editor window titled 'Gemfile'. The window shows a list of Ruby gems to be installed. The first line is 'source 'https://rubygems.org''. The next four lines are 'gem 'cucumber'', 'gem 'rspec'', 'gem 'selenium-webdriver'', and 'gem 'capybara''. The line 'gem 'capybara'' is currently selected, and a cursor is at the end of the line. The line numbers 1 through 7 are visible on the left side of the editor.

```
1 source 'https://rubygems.org'
2
3 gem 'cucumber'
4 gem 'rspec'
5 gem 'selenium-webdriver'
6 gem 'capybara'
7
```

RubyGems é um gerenciador de pacotes para a linguagem de programação Ruby que provê um formato padrão para a distribuição de programas Ruby e bibliotecas em um formato auto-suficiente.

Arquitetura do projeto - Gem Bundler

Utilizando o Bundler

O que é o Bundler?

Bundler é um gerenciador de dependências para projetos Ruby. Com ele instalado, todas as dependências do projeto especificadas no arquivo Gemfile serão baixadas. Isso ajuda a manter o controle das gems usadas no projeto.

Como usar?

Adicione as gem necessárias no arquivo Gemfile.
Com o arquivo Gemfile configurado, basta executar o **comando para efetuar a** instalação das gem e suas dependências.

Instalar o bundler:

```
C:\dojo_ddmmaa  
λ gem install bundler
```

Instalar as gem e suas dependências:

```
C:\dojo_ddmmaa  
bundle|
```


Arquitetura do projeto

Aprendendo a usar o env.rb

- O env.rb inicializa configurações do teste, tal como o navegador que deve ser utilizado.
- O require sobe em memória os arquivos das Gems necessárias.
- Segue um exemplo de env.rb para um projeto de automação web.

```
env.rb
1 #sobe em memória as Gems informadas
2 require 'rspec'
3 require 'cucumber'
4 require 'selenium-webdriver'
5 require 'capybara'
6 require 'capybara/cucumber'
7
8
9 #Configurando o driver Capybara
10 Capybara.register_driver :selenium do |app|
11   Capybara::Selenium::Driver.new(app, :browser => :chrome)
12 end
13
14 #Setando a configuração do driver como padrão
15 Capybara.default_driver = :selenium
16
17 #timeout padrão na execução
18 Capybara.default_max_wait_time = 15
19
20 #Maximizar a tela ao iniciar o teste
21 Capybara.page.driver.browser.manage.window.maximize
```

4

Cucumber e Capybara

in

Cucumber

Escrevendo um Cenário BDD / Criando uma feature

O arquivo "<nome_funcionalidade>.feature" contém o BDD a ser testado.

Esse arquivo deve ser salvo na estrutura do projeto dentro da pasta "Features"

Esse arquivo é interpretado pelo Cucumber para chamar os *steps* implementados.

Necessário especificar o idioma que será utilizado. Exemplo: "#language: pt"

```
1 #language: pt
2 #utf-8
3
4 Funcionalidade: Buscar Agencias
5   Eu como cliente do banco
6   Quero procurar uma agencia dentro do Brasil
7   Para saber suas informacoes de contato
8
9   Cenario: Buscar agencia por CEP valido
10      Dado que esteja na home do site do banco
11      Quando buscar uma agencia pelo CEP
12      Entao apresentara as agencias disponiveis
13
```

Boas praticas: nomes minúsculos e underline no lugar dos espaços.

Cucumber

Gerando os Steps

Após salvar o arquivo `.feature`, acesse o terminal e execute:

1. Acessar a pasta do projeto:

```
cd <pasta_projeto>
```

2. Gere os steps:

```
cucumber
```

3. Copie os steps gerados em amarelo e salve em um arquivo `.rb` dentro da pasta `"step_definitions"`.

*o nome do arquivo não precisa se o mesmo da feature.

Boas praticas: palavras em minúsculos, underline no lugar dos espaços e `_steps` no final.

```
Dado(/^que esteja na home do site do banco$/) do
  pending # Write code here that turns the phrase above into concrete actions
end

Quando(/^buscar uma agencia pelo CEP$/) do
  pending # Write code here that turns the phrase above into concrete actions
end

Entao(/^apresentara as agencias disponiveis$/) do
  pending # Write code here that turns the phrase above into concrete actions
end
```

```
1 Dado(/^que esteja na home do site do banco$/) do
2   pending # Write code here that turns the phrase above into concrete actions
3 end
4
5 Quando(/^buscar uma agencia pelo CEP$/) do
6   pending # Write code here that turns the phrase above into concrete actions
7 end
8
9 Entao(/^apresentara as agencias disponiveis$/) do
10  pending # Write code here that turns the phrase above into concrete actions
11 end
12
```

Capybara

Implementando os steps

- *Responsável por simular as iterações do usuário com o navegador pela interface web da aplicação.*
- *Os comandos do Capybara devem ser inseridas dentro dos **steps** gerados pelo Cucumber.*

```
1 | Dado(/^que esteja na home do site do banco$/) do
2 |   visit "http://www.banco.com.br"
3 | end
4 |
5 | Quando(/^buscar uma agencia pelo CEP$/) do
6 |   fill_in "refCep", with => "04534011"
7 |   find("OperacaoBuscaAgencia").click
8 | end
9 |
10 | Entao(/^apresentara as agencias disponiveis$/) do
11 |   assert_text('1744 - SELECT ITAIM BIBI_SP')
12 | end
```

Capybara - Comandos Básicos

Implementando os steps

Navegação

visit '<https://site.com.br>'

Clique links e botões por id, texto ou nome

click_link('id-do-link')

click_link('Texto do Link')

click_link('nome_d_link')

Clica em um botão por id, texto ou nome

click_button('id-do-botao')

click_button('Texto do botao')

click_button('nome_do_botao')

Interagindo com Formulários

fill_in('nome_do_elemento', :with => 'valor')

choose('nome_do_radio_button')

check('nome_do_checkbox')

unchecked('nome_do_checkbox')

select('opção', :from => 'nome_do_combobox')

Buscar um elemento na página

find('#id')

find('nome_do_elemento')

find('.class')

find(:id, 'id_do_elemento')

find(:xpath, 'xpath_do_elemento')

find(:css, 'css_do_elemento')

Validações

assert_text('texto_que_deve_existir')

assert_no_text('texto_que_não_deve_existir')

has_xpath?('existe_xpath_do_elemento?')

has_css?('existe_css?')

has_content?('existe_conteúdo?')

has_link?('existe_link?')

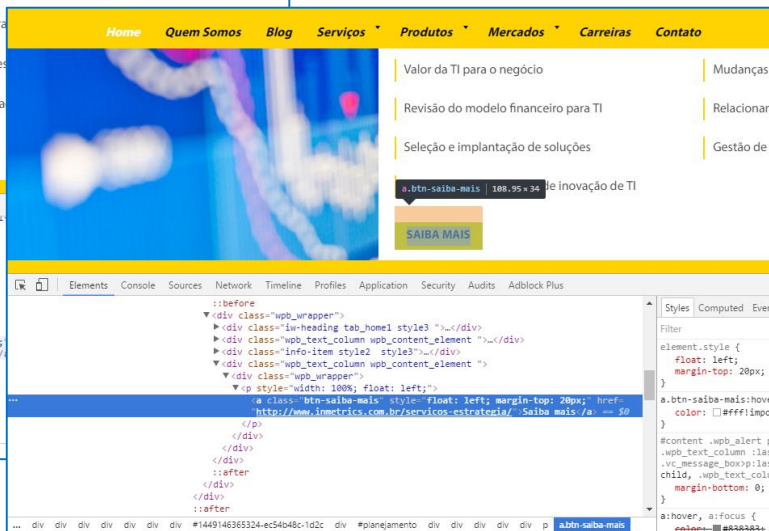
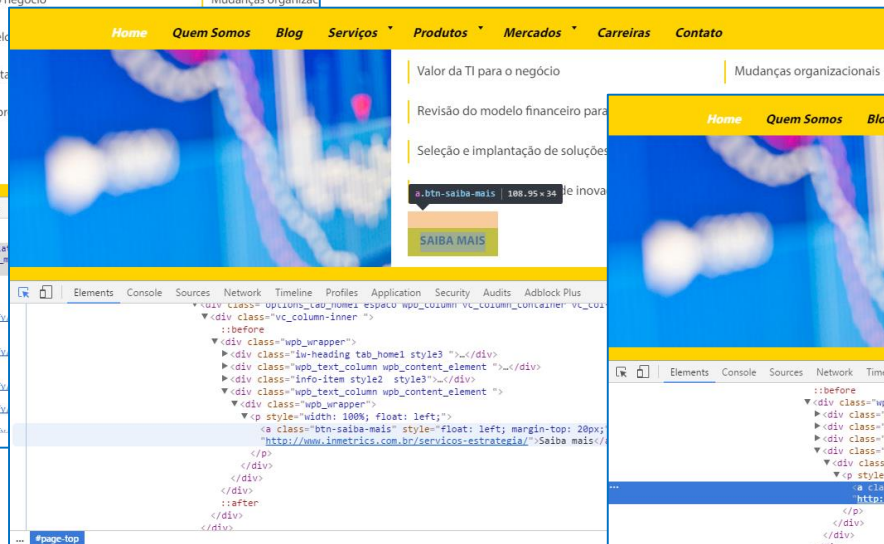
should have_xpath('deve_existir_xpath_do_elemento')

should have_css('deve_existir_css')

should have_content('deve_existir_conteúdo')

should have_no_content('não_deve_existir_conteúdo')

Localizando elemento por id, class, name



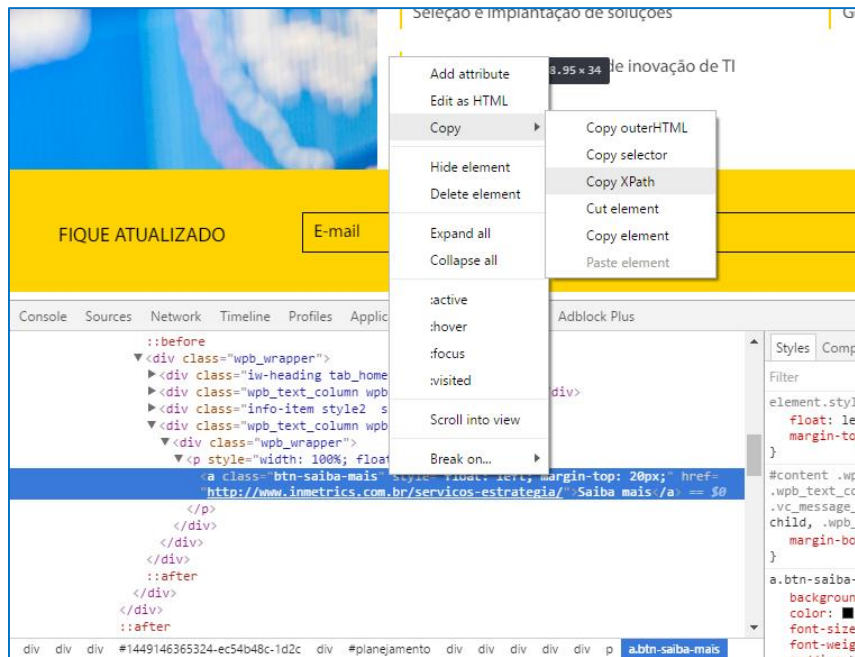
- *Apertar F12*

- Clicar no ícone do cursor

- Selecionar e clicar sobre o elemento para identificar o id/class/name dele

Inspecionando Elementos de um Site

Copiando xpath do elemento



Para copiar o xpath:

- Clique com o botão direito do mouse sobre código do elemento
- Copy
- Copy XPath

*um exemplo de xpath

`//*[@id="planejamento"]/div[2]/div/div/div[4]/div/p/a`

Vamos Praticar?

5



in

Vamos Praticar?

Automatizando na prática

Exercício 1:

Escreva

Cenário (BDD) que a partir do site da Inmetrics valide o texto: "mais de 15 anos de atuação" dentro da página "Quem Somos".

***Dica:**

Leia e entenda o enunciado antes de iniciar o processo de escrita do cenário em BDD e da automação dele.

Execute o cenário manualmente!

Antes de começar.

Configurando sua máquina

- **Ruby for Windows:** linguagem de programação utilizada nos testes.
- **cmdr for Windows:** Sistema que trás as funcionalidades bash (Terminal) para o Windows.
- **DevKit:** Kit de ferramentas que o sistema operacional precisa para que o desenvolvimento funcione.
- **Chromedriver:** Driver do navegador que será utilizado na automação. Disponível no site do seleniumhq.org
- **SublimeText:** Editor de texto com funções úteis para escrever o código da automação de testes.

Relembrando...



Relembrando

Automatizando na prática!

Crie o projeto com a gem: **TESTGEN**

Configure no *Gemfile* as gems básicas para a execução

Execute o comando para instalar as gems

Relembrando

Automatizando na prática!

Configure o arquivo **env.rb**

Gere os steps

Implemente as ações nos steps

Vamos Praticar?

Automatizando na prática!

- Todos os cenários estão **verdes**?
Parabéns!!!



Vamos Praticar?

Automatizando na prática

Exercício 2:

Escreva

Cenário (BDD) em que o usuário efetue um cadastro no site <http://demoqa.com/> .

Retrospectiva

- O que foi bom?
- O que podemos melhorar?
- O que não foi bom?

Desafio!!!

Automatizando na prática!

Automatize os seguintes cenários, seguindo as definições que aprendeu nesse dojo:

1. Logar no site **ORANGEHRM** (<https://enterprise-demo.orangehrmlive.com/>) com perfil **ADMIN** e cadastrar um novo empregado <Add Employee>
2. Logar no site **ORANGEHRM** com perfil **ADMIN** e editar um empregado existente na **Lista de Empregado** <Employee List>

Lembre-se:

Dúvidas?

dojo_automacao@lnmetrics.com.br :D

Links Importantes

Automatizando na prática!

Curso de Ruby: <https://www.codecademy.com/learn/ruby>

Boas práticas Cucumber: <https://github.com/strongqa/howitzer/wiki/Cucumber-Best-Practices>

Boas práticas SitePrism: https://github.com/natritmeyer/site_prism

Gherkin: <https://github.com/cucumber/cucumber/tree/master/gherkin>

GitHub: <https://github.com>

Step by Step: <https://help.github.com/articles/create-a-repo/>



Barueri . +55 11 3303-3200

Av. Tamboré, 267 - 21º andar
Torre Norte, Tamboré, Barueri - SP - Brasil
CEP: 06460-000

São Paulo . +55 11 3303-3200

Av. Eng. Luiz Carlos Berrini, 105 - 16º andar
Sala 1607, Brooklin Novo, São Paulo - SP - Brasil
CEP: 04571-010

Rio de Janeiro . +55 21 3232-2611

Rua Visconde de Inhaúma, 134 - 20º andar
Sala 2016/2017, Centro, Rio de Janeiro - RJ - Brasil
CEP: 20091-007

Chile . +56 2 3203-9507

Cerro El Plomo, 5420
Oficina 1503, Las Condes, Santiago - Chile
Código Postal: 7560742

Colômbia . +57 1 646-9642

Carrera 19A #90-13
Oficina 304, Bogotá - Colômbia
Código Postal: 110221