



*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

*Arquitetura  
do projeto*

4

*Cucumber  
e  
Capybara*

5

*Pry*

6

*Cucumber  
Tags*

**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...



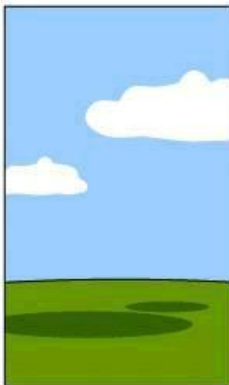
Como o analista projetou...



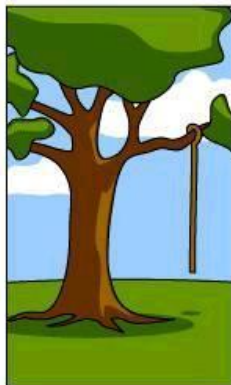
Como o programador construiu...



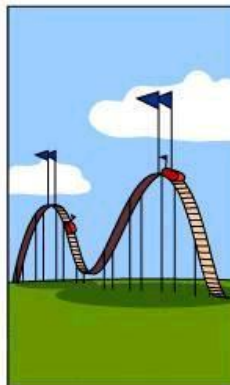
Como o Consultor de Negócios descreveu...



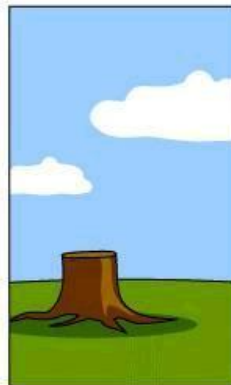
Como o projeto foi documentado...



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** / **E**) 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: User Registration
```

```
As a new user
```

```
I'd like to register myself
```

```
Scenario: Register a new User
```

```
Given I am on the Registration screen
```

```
And I do the registration with valid data
```

```
Then the registration is successfully finished
```

```
#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
```

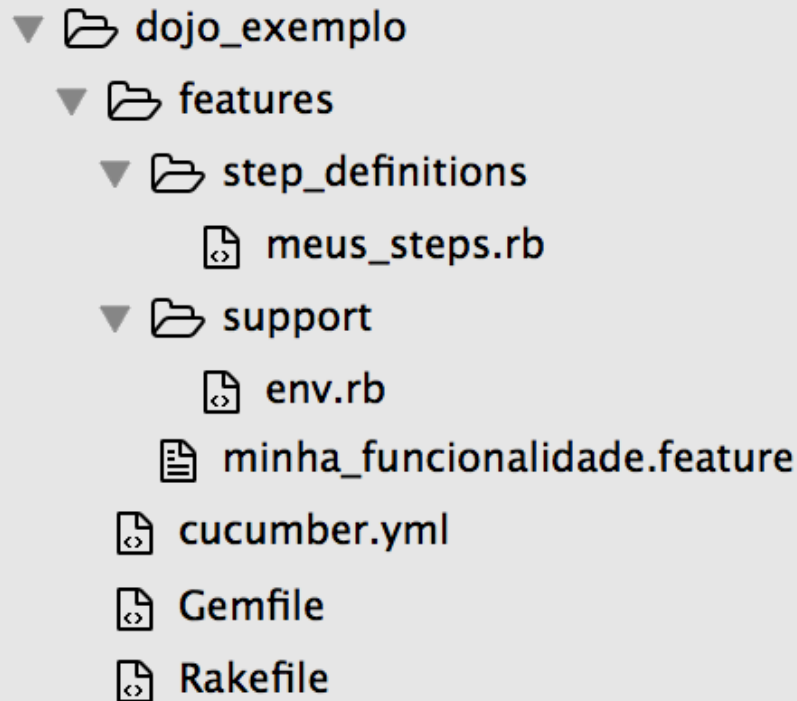
**3**

## *Arquitetura do Projeto*


***in***










# Arquitetura do projeto

## Entendendo a Estrutura Básica do Projeto



```
graph TD; dojo_exemplo[dojo_exemplo] --> features[features]; dojo_exemplo --> cucumber[cucumber.yml]; dojo_exemplo --> Gemfile[Gemfile]; dojo_exemplo --> Rakefile[Rakefile]; features --> step_definitions[step_definitions]; features --> support[support]; 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

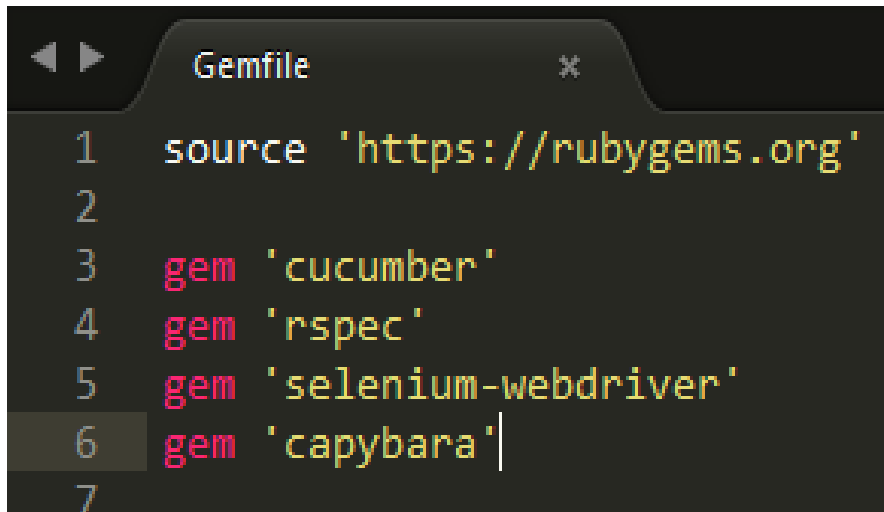
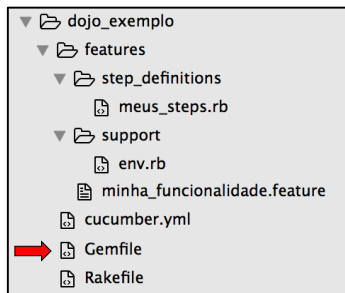
## Montando seu ambiente de testes

```
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 showing a 'Gemfile'. The file has a tab labeled 'Gemfile' with a close button. The content is as follows:

```
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|
```



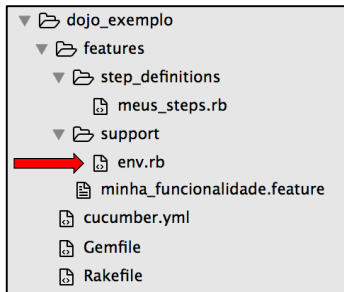
# Arquitetura do projeto - env.rb

## 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

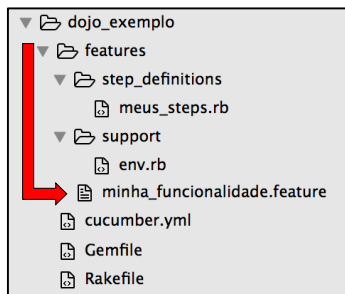
## Escrevendo um Cenário BDD / Criando uma feature

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**”:

1. No **Sublime**, dentro da pasta “**step\_definitions**”, crie um arquivo “**<nome\_arquivo>\_steps.rb**”

\*o nome do arquivo não precisa ser o mesmo da feature

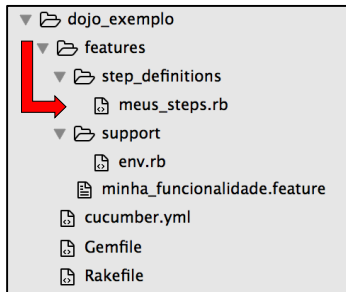
2. Verifique no **terminal** se o caminho apresentado é o do projeto criado pelo **testgen**. Se não for o mesmo, digite o comando abaixo para acessar a pasta:

**cd dojo\_ddmmaa**

3. Ainda no terminal, execute o comando para o Cucumber gerar os snippets não implementados:

**cucumber**

4. Copie os steps em **amarelo**, cole no arquivo “**\_steps.rb**” e salve



```
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
```

A screenshot of a code editor showing the contents of the 'buscar\_agencias.feature' file. The code is highlighted in yellow, indicating it has been copied. The code is as follows:

```
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
```

\* **Boas praticas:** palavras em minúsculos, underline no lugar dos espaços e **\_steps** no final

# 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?')

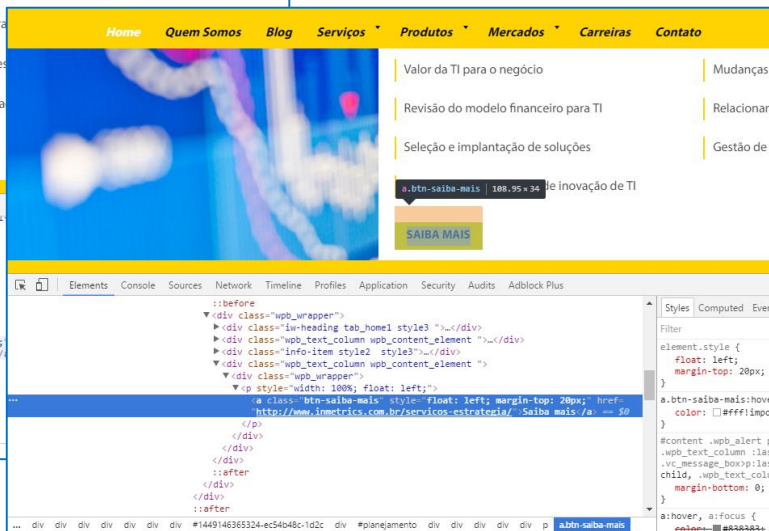
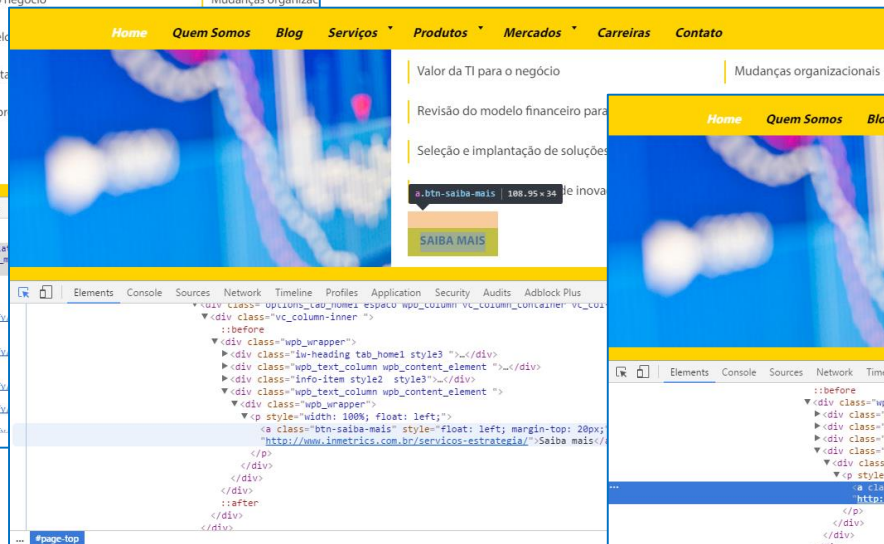
should have\_xpath('deve\_existir\_xpath\_do\_elemento')

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



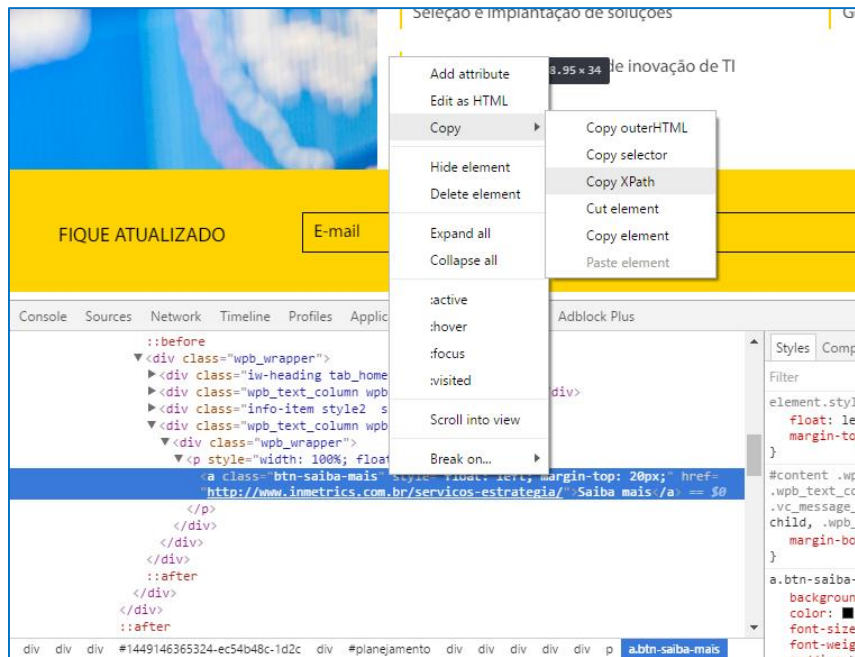
- *Apertar F12*

- 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?*



***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, se caso for necessário!*

# 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](https://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!!!**





\_\_\_\_ GET TO THE CODE \_\_\_\_

**5**

***Pry***

***in***



# Pry

## Debug no Ruby

O **Pry** é uma gem do Ruby que nos permite fazer o **debug** do nosso código.

Este **debug** é feito no **IRB** pelo terminal.

**\*Debug**, é o ato de depurar o código. Olhar passo a passo cada ação para identificar possíveis erros no fluxo

**\*IRB**, significa **Interactive Ruby Shell**.  
Basicamente ele habilitará todos os comandos ruby em seu terminal, onde você poderá executar e ver os resultados em tempo real.

# Pry

## Debug no Ruby

### Instalando o pry

- Incluir a gem no arquivo *Gemfile* `gem 'pry'`
- Efetuar o require da gem no *ENV.rb* `require 'pry'`
- Dentro da pasta do projeto, executar o comando no terminal: *bundle* `x bundle`

### Usando o pry

*O pry pode ser utilizado nos steps ou métodos da automação. Para tal, basta inserir o seguinte comando:*

```
binding.pry
```

# Pry - Exemplo

## Debug no Ruby

```
13 Quando(/^eu buscar agência no bairro$/) do
14     @app.agency.click_something("Agência")
15     binding.pry
16     @app.agency.click_something("Clique aqui")
17 end
18
```

# Pry - Exemplo

## Debug no Ruby

```
Cenário: Buscar agência por CEP válido
  Dado que eu esteja na home do site Santander
  Quando eu buscar agência pelo CEP
  Então aparecerá as agências disponíveis do CEP

Cenário: Buscar agência por bairro
  Dado que eu esteja na home do site Santander

From: /Users/vtrmartinez/Documents/SantanderSite/SantanderSitePrism/features/step_definitions/agency_steps.rb @ line 15 :

10:   end
11: end
12:
13: Quando(/^eu buscar agência no bairro$/) do
14:   @app.agency.click_something("Agência")
=> 15:   binding.pry
16:   @app.agency.click_something("Clique aqui")
17:
18:   @app.agency.last_window
19:   within_frame(@app.searchAgency.iframe) do
20:     @app.searchAgency.neighborhood.neighborhoodTab.click
```

# Pry - Comandos

## Debug no Ruby

### Navigating pry

<code>!pry</code>	Start a pry session on current self.
<code>disable-pry</code>	Stops all future calls to pry and exits the current session.
<code>exit</code>	Pop the previous binding.
<code>exit-all</code>	End the current pry session.
<code>exit-program</code>	End the current program.
<code>jump-to</code>	Jump to a binding further up the stack.
<code>nesting</code>	Show nesting information.
<code>switch-to</code>	Start a new subsession on a binding in the current stack.

6

## *Cucumber Tags*

*in*

# Cucumber Tags

## Organizando seus testes!

**Tag** é uma ótima maneira de organizar a execução de suas funcionalidades e cenários:

```
@regressao
Funcionalidade: Registro de usuários

  @importante
  Cenário: Cadastrar cliente

  Cenário: Alterar cadastro do cliente
```

Uma funcionalidade ou cenário pode ter várias **tags**. Basta separar por ‘espaço’:

```
@regressao @diario @smoke
Funcionalidade: Registro de usuários
```

# Cucumber Tags

## Organizando seus testes!

```
@consulta
Funcionalidade: Pesquisa de termos
  Eu como usuário do sistema
  Quero consultar um termo
  Para saber mais informações referentes ao termo

  Cenário: Pesquisar termos com filtros ativados
    Dado que eu esteja na area de consulta
    Quando aplico o filtros "em estoque" e "pronta entrega"
    E eu pesquiso o termo "café"
    Então eu vejo detalhes sobre o termo
```

### Alguns exemplos de tags:

- Momentos em que os testes devem rodar  
*@hourly, @daily, @diariamente*
- De acordo com suas dependências externas  
*@local, @database, @network*
- Nível  
*@functional, @system, @smoke*
- Sistemas/Produtos  
*@home, @carrinho*



# Cucumber Tags - Exemplos de Execução

## Organizando seus testes!

```
cucumber --tags @pendente # Executa os cenários/funcionalidades com a tag @pendente
cucumber --tags @smoke # Executa os cenários/funcionalidades com a tag @smoke
cucumber --tags ~@pendente # Executa somente os que NÃO tenham a tag @pendente
cucumber --tags @pendente --tags @smoke # Executa os que tenham a tag @pendente E @smoke
cucumber --tags @pendente,@smoke # Executa os que tenham a tag @pendente OU @smoke
cucumber -t @pendente # -t é um atalho para --tags
```

No terminal, quando quisermos rodar os cenários que estão nomeados com tags, basta utilizarmos o seguinte comando:

**cucumber --tags @nome\_da\_tag**

ou

**cucumber -t @nome\_da\_tag**

# Vamos Praticar?

*Automatizando na prática*

## Exercício 2:

### Escreva

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

*\*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, se caso for necessário!*

**Vamos Praticar?**

*Automatizando na prática!*

*Todos os  
cenários estão  
verdes?*

**Parabéns!!!**



# *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>

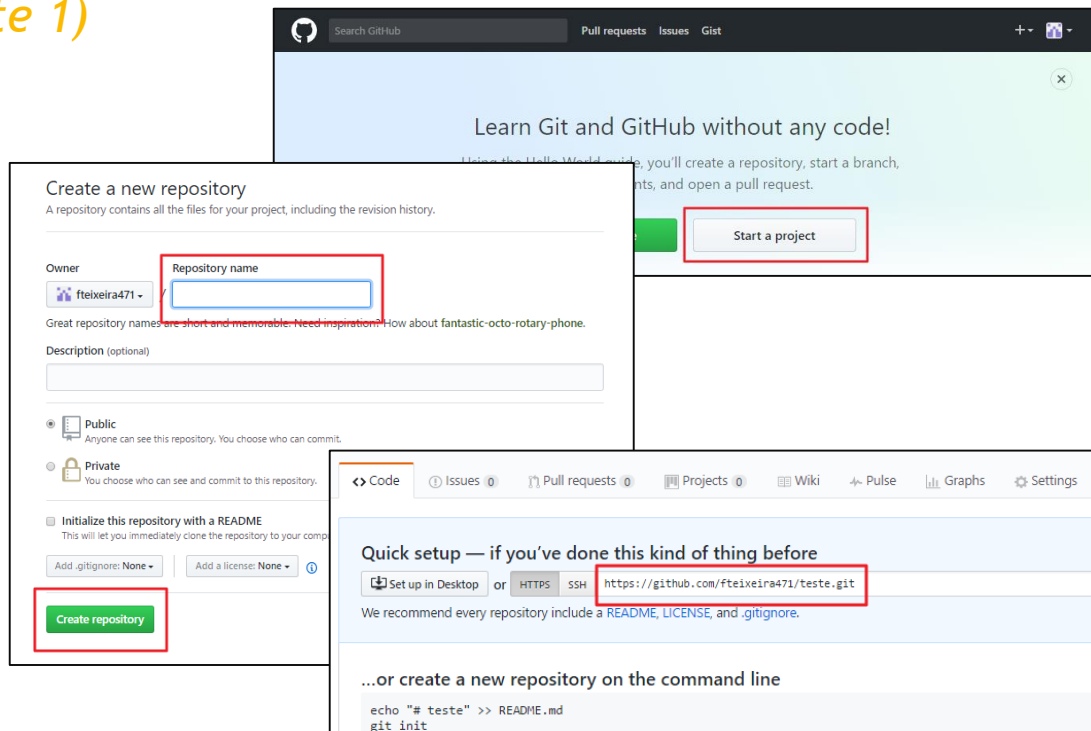
Concluído o desafio, coloque o projeto no **GitHub**  
e envie o link para [agile\\_itau@inmetrics.com.br](mailto:agile_itau@inmetrics.com.br)

# GitHub

## Versionando seus projetos! (Parte 1)

Para se cadastrar e criar um repositório para o seu projeto no GitHub:

1. Acesse o site <https://github.com> e crie sua conta em **“Sign Up”**
2. Logue com sua conta e, na página principal, clique em **“Start a project”**
3. Insira o nome do repositório e clique em **“Create repository”**
4. O link gerado será utilizado para você subir os arquivos do projeto



# GitHub

## Versionando seus projetos! (Parte 2)

Depois de criar o repositório, você pode subir seu projeto:

Se for a primeira vez que utiliza GitHub, efetue esses passos para a configuração inicial dentro do **terminal**:

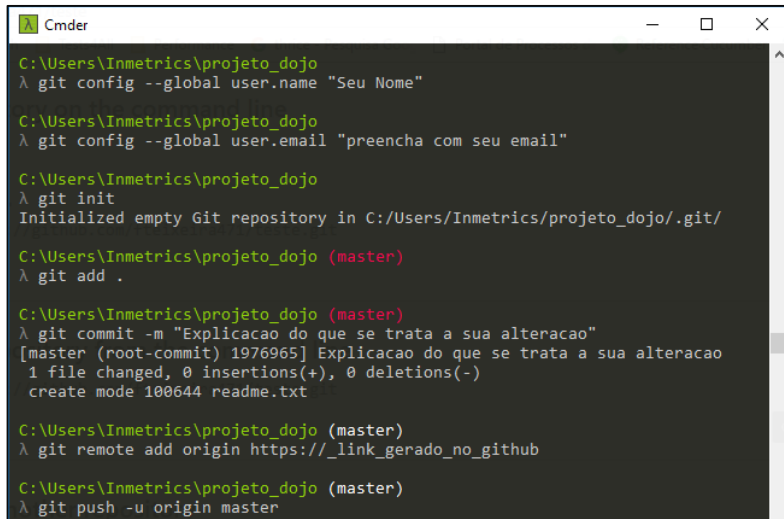
```
git config --global user.name "Seu Nome"
git config --global user.email "preencher com email" (sem aspas)
```

No **terminal**, dentro da **pasta de seu projeto**, digite os comandos abaixo para inicializar o projeto e subir os arquivos:

```
git init
git add .
git commit -m "breve explicação sobre as alterações feitas"
git remote add origin "url gerada no GitHub" (sem aspas)
git push -u origin master
```

Após, se fizer novas alterações, execute:

```
git add . (ou, ao invés do "." o caminho do arquivo/pasta que alterou)
git commit -m "mensagem"
git push
```



```
C:\Users\Inmetrics\projeto_dojo
λ git config --global user.name "Seu Nome"

C:\Users\Inmetrics\projeto_dojo
λ git config --global user.email "preencha com seu email"

C:\Users\Inmetrics\projeto_dojo
λ git init
Initialized empty Git repository in C:/Users/Inmetrics/projeto_dojo/.git/

C:\Users\Inmetrics\projeto_dojo (master)
λ git add .

C:\Users\Inmetrics\projeto_dojo (master)
λ git commit -m "Explicacao do que se trata a sua alteracao"
[master (root-commit) 1976965] Explicacao do que se trata a sua alteracao
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 readme.txt

C:\Users\Inmetrics\projeto_dojo (master)
λ git remote add origin https://_link_gerado_no_github

C:\Users\Inmetrics\projeto_dojo (master)
λ git push -u origin master
```

# Links Importantes

*Automatizando na prática!*

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

**Wiki do Cucumber:** <https://github.com/cucumber/cucumber/wiki/A-Table-Of-Content>

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

**Capybara:** <https://github.com/teamcapybara/capybara>

**Git (Comandos úteis):** <https://zeroturnaround.com/rebellabs/git-commands-and-best-practices-cheat-sheet/>

**Pratique Git:** <https://try.github.io>

**Boas práticas SitePrism:** [https://github.com/natritmeyer/site\\_prism](https://github.com/natritmeyer/site_prism)





Obrigado pela presença!!!

Dúvidas? [agile\\_itau@inmetrics.com.br](mailto:agile_itau@inmetrics.com.br)



**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