Appium

Dojo para automação Mobile

Dinâmica Dojo

Agenda

- Relembrando BDD
- Sobre o Appium
- Gem appium_lib
- Appium Server
- Arquitetura do Projeto
- Comandos básicos
- Appium Desktop

BDD

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

Tradução BDD: **Desenvolvimento Guiado por Comportamento**

Cucumber

- 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.

Appium

Framework

- Appium é um framework de automação de teste open source para aplicativos móveis **nativos e híbridos.**

Appium

Framework

- Appium é um framework de automação de teste open source para aplicativos móveis **nativos e híbridos.**

- O Appium é multi-plataforma, ou seja, permite que você escreva testes nas plataformas (**iOS e Android**) usando a mesma API. Isso permite a reutilização de código entre os seus testes.

Appium

Framework

- Appium é um framework de automação de teste open source para aplicativos móveis **nativos e híbridos.**

- O Appium é multi-plataforma, ou seja, permite que você escreva testes nas plataformas (**iOS e Android**) usando a mesma API. Isso permite a reutilização de código entre os seus testes.

- Você pode escrever testes com suas ferramentas dev favoritas usando qualquer linguagem compatível com WebDriver como **Java, Objective-C, JavaScript (Node), PHP, Python, Ruby e C#**.

Appium Lib

Helper Methods

- Biblioteca com comandos e interações mobile

Appium Lib

Helper Methods

- Biblioteca com comandos e interações mobile

- Disponível para diversas linguagens de programação

Appium Lib

Helper Methods

- Biblioteca com comandos e interações mobile
- Disponível para diversas linguagens de programação
- Comunica-se com o appium server

Appium Server

Core

- Recebe os comandos da Appium lib

Appium Server

Core

- Recebe os comandos da Appium lib

- É responsável por comunicar-se e realizar as interações com o(s) dispositivo(s)

Appium

Desired Capabilities

São um conjunto de **chave/valor** com os parâmetros desejados, que são enviados ao Appium server para iniciar uma sessão. É através do Desired Capabilities, por exemplo, que informamos em qual dispositivo queremos executar, path do aplicativo e quais as configurações iniciais.

Capabilities para Android

```
{
    platformName: "Android"
    deviceName: "Nexus_5_API_23_mars"
    app: "/Users/texugo/docu/apk/nome.apk"

}
```

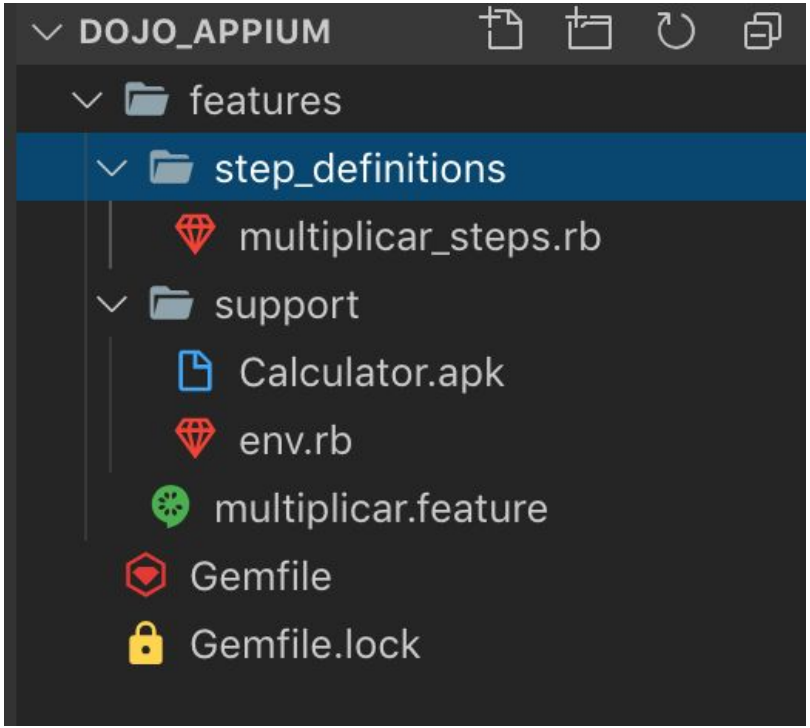
Capabilities para iOS

```
{
    platformName: "iOS"
    platformVersion: "10.3"
    deviceName: "iPhone 6 Plus"
    app: "/Users/texugo/docu/ios/nome.app"

}
```

Arquitetura do Projeto

Arquitetura do projeto

Entendendo a Estrutura Básica do Projeto



Arquitetura do projeto

Env = Ambiente.rb

multiplicar.feature | env.rb

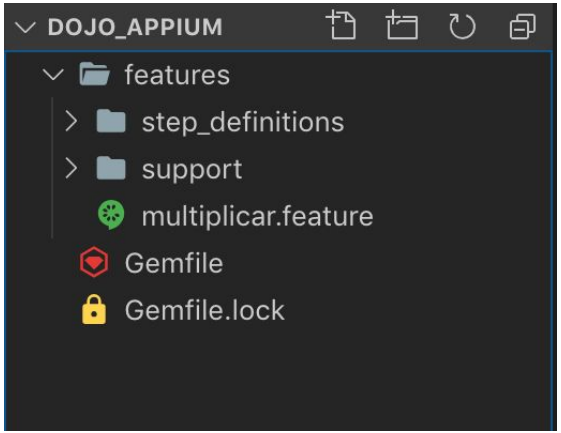
features > support > env.rb > ...

```ruby
require 'appium_lib'
require 'cucumber'
require 'rspec/expectations'

def caps
  { caps: { deviceName: 'testeCalculadora',
            platformName: 'Android',
            app: File.join(File.dirname(__FILE__), 'Calculator.apk')
          }
  }
end

Appium::Driver.new(caps, true)
Appium.promote_appium_methods Object

Before { start_driver }
After { driver_quit }

```

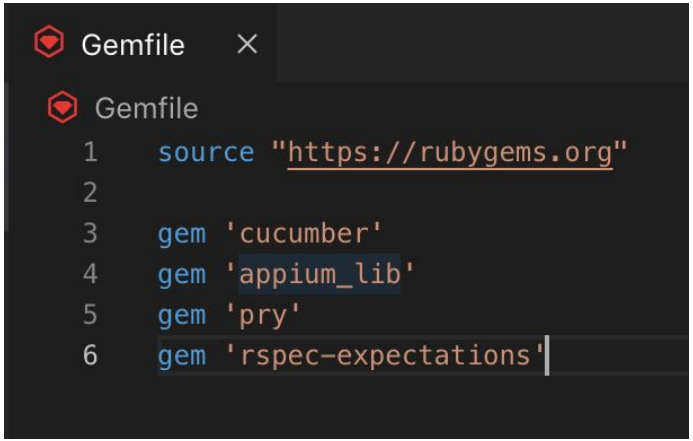
- Require
- Método caps
- Criação de um novo Driver
- Before
- After

Arquitetura do projeto

O que é uma Gem? O que é o Gemfile?



Gem é uma biblioteca contendo códigos que você pode reaproveitar importando em outros projetos.



Gemfile é um arquivo de gerenciamento das gems utilizadas no seu projeto Ruby.

Arquitetura do projeto

Montando seu ambiente de testes

```
marcosmanoel@MacBook-Pro-de-Marcos ~/Desktop/dojo_appium
$ cd ~/Desktop/
marcosmanoel@MacBook-Pro-de-Marcos ~/Desktop
$ mkdir dojos
marcosmanoel@MacBook-Pro-de-Marcos ~/Desktop
$ cd ~/desktop/dojos/
marcosmanoel@MacBook-Pro-de-Marcos ~/desktop/dojos
$ mkdir dojo_appium
marcosmanoel@MacBook-Pro-de-Marcos ~/desktop/dojos
$ bundle init
Writing new Gemfile to /Users/marcosmanoel/Desktop/dojos/Gemfile
marcosmanoel@MacBook-Pro-de-Marcos ~/desktop/dojos
$ cucumber --init
   create   features
   create   features/step_definitions
   create   features/support
   create   features/support/env.rb
marcosmanoel@MacBook-Pro-de-Marcos ~/desktop/dojos
$
```

- No terminal, instale a gem do bundler:
  - gem install bundler
- Crie a pasta para o seu projeto:
  - mkdir dojo_ddmmaa
  - cd dojo_ddmmaa
  - bundle init
- Abra o projeto no seu editor de texto preferido.
- Preencha o Gemfile com a gem do cucumber
- Crie a estrutura (pastas) para o seu projeto:
  - cucumber --init

Cucumber

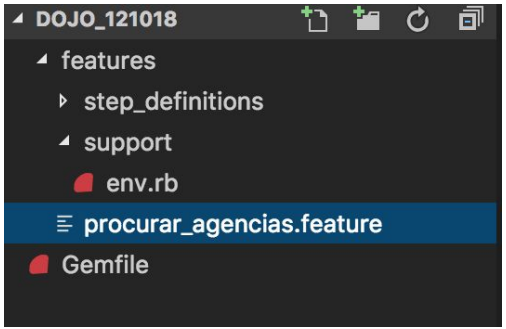
Escrevendo um Cenário BDD / Criando uma feature

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”.



procurar_agencias.feature

```
1   #language: pt
2
3   Funcionalidade: Procurar agências
4   Eu como cliente do banco
5   Quero procurar uma agência dentro do Brasil
6   Para saber sua localização
7
8   Cenário: Procurar agência mais próxima de um CEP
9       Dado que esteja na página do banco
10      Quando procurar uma agência pelo CEP
11      Então apresentará as agências mais próximas
```

* Boas práticas: nomes minúsculos e underline no lugar dos espaços.

Cucumber

Gerando os Steps

```
You can implement step definitions for undefined steps with these snippets:

Dado("que possuo um novo caminhao") do
  pending # Write code here that turns the phrase above into concrete actions
end

Quando("crio um novo frete") do
  pending # Write code here that turns the phrase above into concrete actions
end

Entao("efetuo o agendamento do frete") do
  pending # Write code here that turns the phrase above into concrete actions
end
```

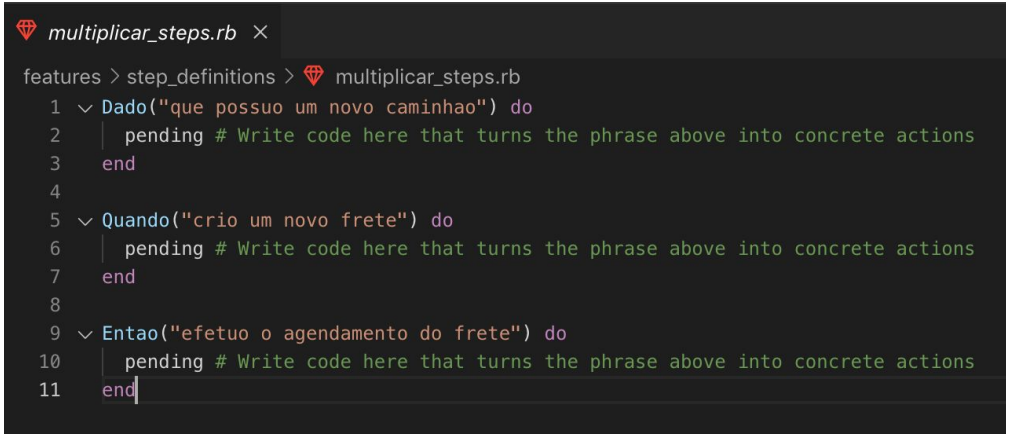
Após salvar o arquivo “**.feature**”:

1. Dentro da pasta “**step_definitions**”, crie um arquivo “**<nome_arquivo>_steps.rb**”.

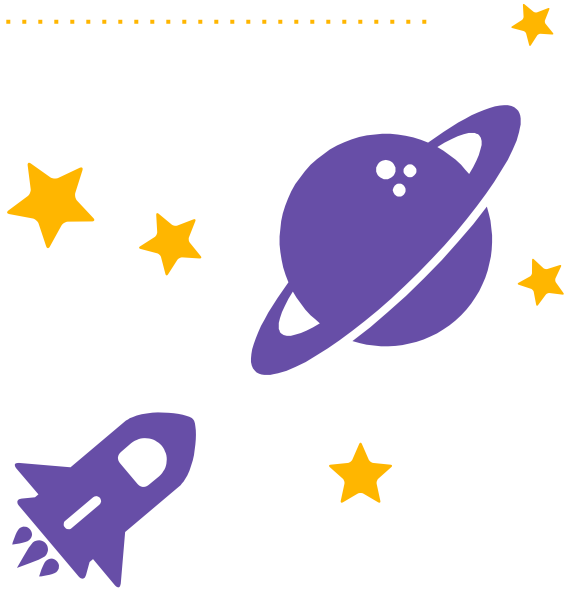
2. Ainda no terminal, execute o comando para o Cucumber gerar os snippets não implementados:

   cucumber

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



Comandos base



Appium - Comandos Básicos

Implementando os steps

Busca de elementos

find_element(id:'<id_elemento> ') = Busca um elemento pelo id
find_element(class:'<classe_elemento> ') = Busca um elemento pela classe
find('<trecho de um texto>') = Busca um elemento que contenha a string desejada
find_exact('<texto>') = Busca um elemento pelo texto

Navegação

find_element(id:'<id_elemento> ').**text** = retorna o texto do elemento
find_element(id:'<id_elemento> ').**name** = retorna o texto do elemento
find_element(id:'<id_elemento> ').**click** = toca no elemento
find_element(id:'<id_elemento> ').**clear** = limpa o campo
find_element(id:'<id_elemento> ').**displayed?** = retorna true ou falso
find_element(id:'<id_elemento> ').**send_keys**("Hello World!") = Escreve em um campo

Appium - Comandos Básicos

Implementando os steps

Asserts => Usamos o Rspec

expect(actual).to eq(expected)
expect(actual).to be(expected)
expect(actual).to be > expected
expect(actual).to be >= expected
expect(actual).to be true

Exemplo: expect(find_element(id: 'title').**text**).to eq("texto esperado")

Interações Celular

back
rotate :landscape
rotate :portrait
hide_keyboard
open_notifications

“

Appium Desktop

Appium Desktop

Mapeando Elementos

Iremos utilizar o **Appium Desktop.**

Abra uma nova sessão do Appium clicando em **New Session Window**

Appium Desktop

Mapeando Elementos

Capabilities:

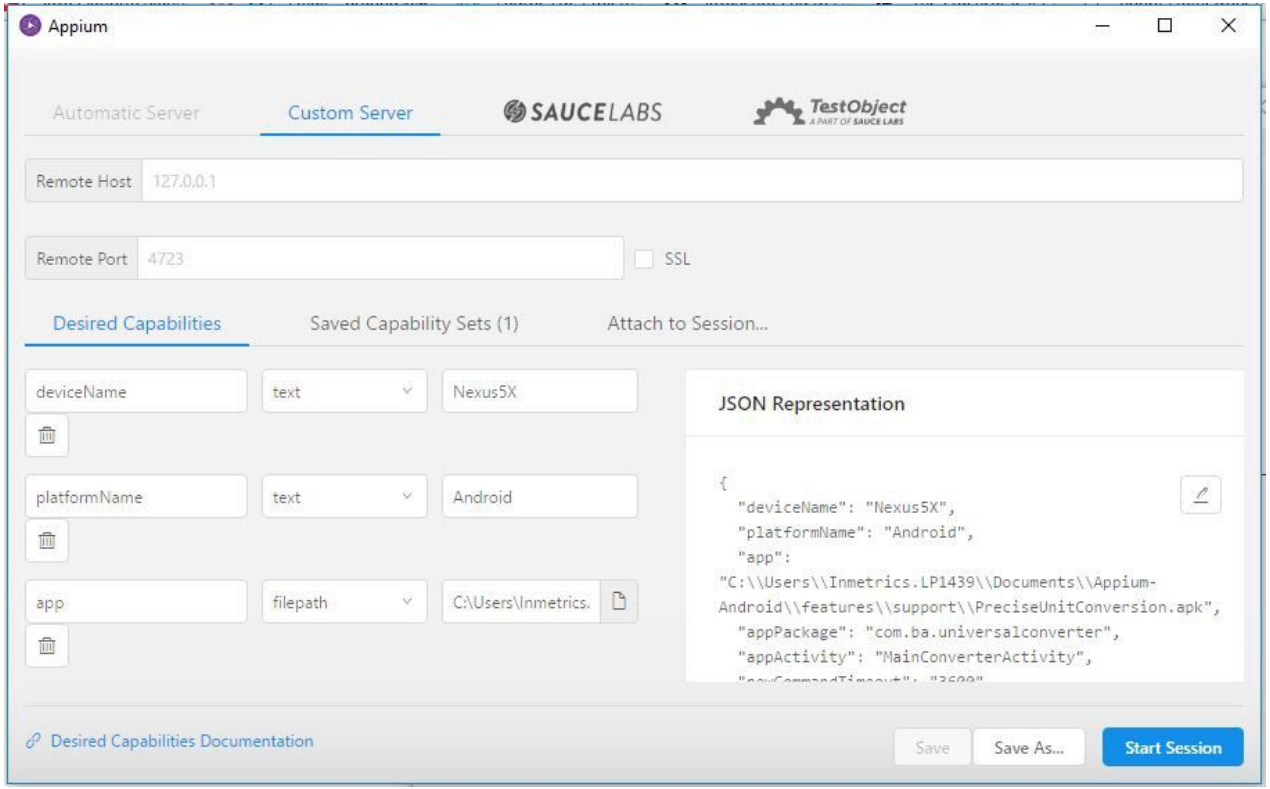
{

deviceName: Nome do seu celular/device utilizado na automação

platformName: qual a plataforma que vou usar na minha automação. Ex.: iOS/Android

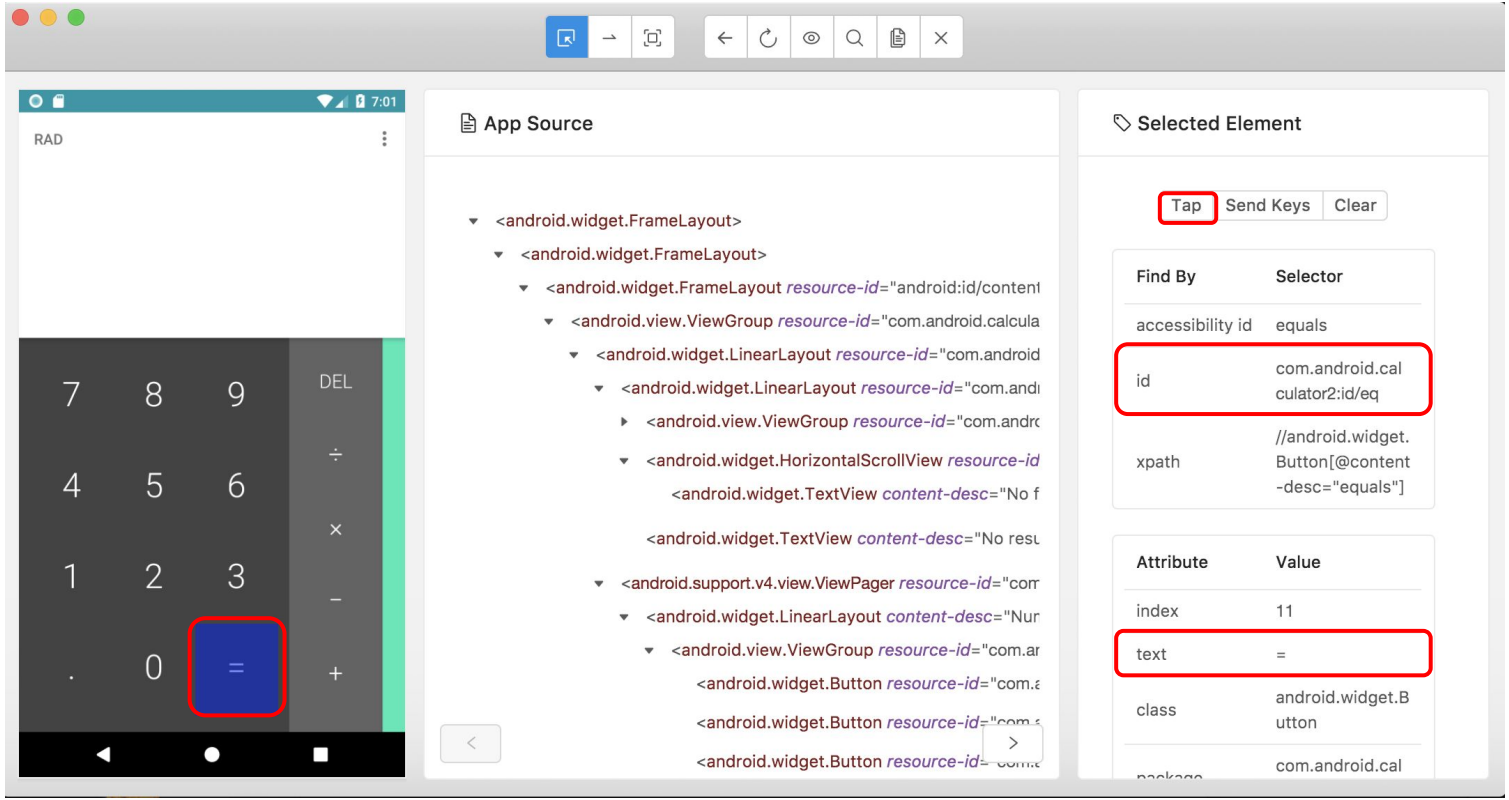
app: qual é o caminho/diretório do seu app.

}



Appium Desktop

Mapeando Elementos



Relembrando...

Relembrando

Automatizando na prática!

Crie a pasta do seu projeto.

Crie e configure o *Gemfile* com as gems básicas para a execução.

Execute o comando para instalar as gems.

Relembrando

Automatizando na prática!

Configure o arquivo **env.rb**

Escreva o BDD

Gere os steps

Implemente as ações nos steps

Execute o seu teste com "cucumber"

Vamos Praticar?

Vamos Praticar?

Automatizando na prática

Escreva um cenário (BDD) que o resultado da divisão entre os valores 8 por 4 seja 2.

Escreva um cenário (BDD) valide o erro: "Não é possível dividir por 0" quando houver a divisão de 5 por 0.

Dica:

***Leia** e **entenda** o enunciado antes de iniciar o projeto.*

Execute o cenário manualmente!!!

- Todos os cenários estão verdes?

Parabéns!!!

Retrospectiva

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

Links Importantes

Automatizando na prática!

Wiki do Cucumber:
https://docs.cucumber.io/guides/

Curso Lógica de Programação com Ruby:
https://www.codecademy.com/learn/ruby

Comandos Terminal:
https://www.devmedia.com.br/comandos-importantes-linux/23893

Doc Appium:
https://github.com/appium/ruby_lib

Git - guia prático :
http://rogerdudler.github.io/git-guide/index.pt_BR.html

Obrigado!