INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO SUL
CAMPUS RESTINGA
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PYBOT
FRAMEWORK DE AUTOMAÇÃO EM BROWSER COM SELENIUM E PYTHON

FELIPE DOS SANTOS VIEGAS

PORTO ALEGRE
2017

FELIPE DOS SANTOS VIEGAS

PYBOT
FRAMEWORK DE AUTOMAÇÃO EM BROWSER COM SELENIUM E PYTHON

Trabalho de Conclusão de Curso apresentada como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas da Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - IFRS, Campus Restinga.

Orientador: Prof. Me. Roben Castagna Lunardi

Co-orientador:

Porto Alegre

2017

FELIPE DOS SANTOS VIEGAS

PYBOT
FRAMEWORK DE AUTOMAÇÃO EM BROWSER COM SELENIUM E PYTHON

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - IFRS, Campus Restinga.

Data de Aprovação: DD/MM/20AA

Banca Examinadora

_____

Prof. Me. Roben Castagna Lunardi - IFRS - Campus Restinga
Orientador

_____

Prof. Mestre dos Magos- IFRS- Campus Restinga
Membro da Banca

_____

Prof. Me. Mestre Splinter- IFRS- Campus Restinga
Membro da Banca

Dedico esse trabalho a todos aqueles que me ajudaram do fim ao começo!

RESUMO

Em muitas empresas temos uma certa carência quando o assunto é automação de testes ou processos web em navegadores. A necessidade de testes de regressão, testes de funcionalidades e ou automação de processos cresce junto com o sistema, porem a pratica dessas atividades só tem força quando aparece aguma necessidade ou problema.

Esse novo framework pretende trazer aos usuarios uma ferramenta de facil uso e com recursos uteis para o desenvolvimento dessas tarefas, contando com a facilidade e versatilidade da linguagem python e a integração com navegadores com framework selenium.

O conjuntos de ferarmentas que o framework dispõe são: gerenciamento automatico dos controladores de navegadores(drivers), modulo de relatorios e logs para controle de atividades executadas, padronização de criação de elementos de tela utilizando o padrão PageObject e a identificação de alteração de layout.

Palavras-chave: Automação, Selenium, Testes.

ABSTRACT

In many companies we have a certain lack when the subject is automation of tests or web processes in browsers. The need for regression testing, functionality testing, and / or process automation grows along with the system, but the practice of these activities is only strong when a need or problem appears.

This new framework aims to bring users an easy-to-use tool with useful resources for the development of these tasks, with the ease and versatility of the python language and the integration with browsers with selenium framework.

The sets of tools that the framework has are: automatic management of the drivers of navigators (drivers), module of reports and logs to control activities performed, standardization of creation of screen elements using the standard PageObject and identification of layout change.

Palavras-chave: Automation, Selenium, Tests.

LISTA DE FIGURAS

Figura 1 – Diagrama de Componentes 16
Figura 2 – Estrutura pybot.ini 17
Figura 3 – Lista de Seletores 18
Figura 4 – Uso padrão Selenium 19
Figura 5 – Uso padrão Selenium com modulo 20
Figura 6 – Uso padrão Pybot 21

SUMÁRIO

1 INTRODUÇÃO 14
2 FRAMEWORK PROPOSTO 15
2.1 Tecnologias Utilizadas 15
2.1.1 Python 15
2.1.2 Selenium WebDriver 15
2.1.3 Git e GitHub 15
2.2 Modulos 16
2.2.1 Core 16
2.2.1.1 Manager 16
2.2.1.2 Configuration 16
2.2.2 Component 17
2.2.2.1 WebElement 17
2.2.2.2 PageElement e PageElements 17
2.2.2.3 PageObject 18
2.2.3 Report 18
2.2.3.1 Logger 18
2.2.3.2 CsvHandler 18
2.2.4 Driloader 18
3 METODO DE DESENVOLVIMENTO PROPOSTO 19
REFERÊNCIAS 22

1 INTRODUÇÃO

O ciclo de vida de software tem diveras etapas, de um modo geral elas são: Análise de requisitos, Concepção do Projeto, Desenvolvimento, Implantação e por fim Manutenção. Nas etapas de Desenvolvimento e a Manutenção é onde a criação ou a codificação do software em questão mais acontece, e na concepção de um projeto a necessidade da criação de um processo de testes que cresça junto do sistema não tem a sua devida importação.

Este Trabalho de Conclusão de Curso tem como objetivo criar um framework para auxiliar nas tarefas de criação de testes e ou automatização de processos de sistemas executados em navegadores. Levando o nome de PyBot, união das palavras Python, liguagem utilizada para criação do projeto, e Bot, que em inglês quer dizer robô, essa ferramenta propõe prover para os usuarios uma serie de ferramentas para auxiliar a criação e implantação desses processos, contando uma estrutura de criação dos scripts de teste no padrão PageObject e PageElement, geração de registros de logs para controle de tarefas e passos executados, verificação de alteração de interfaces e layout dos sites e o gerenciamento automatico de drivers de navegadores com a ferramenta Driloader.

2 Framework Proposto

Neste capítulo irei abordar as tecnologias de utilizadas para a codificação do framework, detalhando os modulos e suas classes expostas para o usuario e as ferramentas utilizadas para controle de versão e publicação do framework.

2.1 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do framework foi utilizado apenas como linguagem para desenvolvimento o Python e para a manipulação e integração com o browser a biblioteca em python do Selenium Webdriver. Por ser um projeto que visa ser o mais simples e leve possivel apenas os modulos padrões do python estão sendo utilizado para o desenvolvimento desta ferramenta.

2.1.1 Python

A escolha do Python (PYTHON, 2017) foi devida porque ele trata-se de uma linguagem de programação fácil de aprender e poderosa. Possuindo uma estruturas dados de alto nível e uma abordagem simples, mas eficaz, para a programação orientada a objetos. Contendo uma Sintaxe elegante e tipagem dinâmica, juntamente com uma interpretação natural, tornam a linguagem ideal para criação dos scripts do pybot.

2.1.2 Selenium WebDriver

Selenium Webdriver (WEBDRIVER, 2017) é um framework utilizado para se comunicar e enviar comandos para os browser em conjunto com um controlador de cada browser especifico. Em comparação com seu antecessor, Selenium RC, o Selenium Webdriver não precisa de um server para enviar os comandos para o browser. Utilizando comando nativos do sistema operacional ao invés de comando javascript, usados pelo Selenium RC, deixam o Selenium Webdriver uma excelente ferramenta para integração com diversos browser.

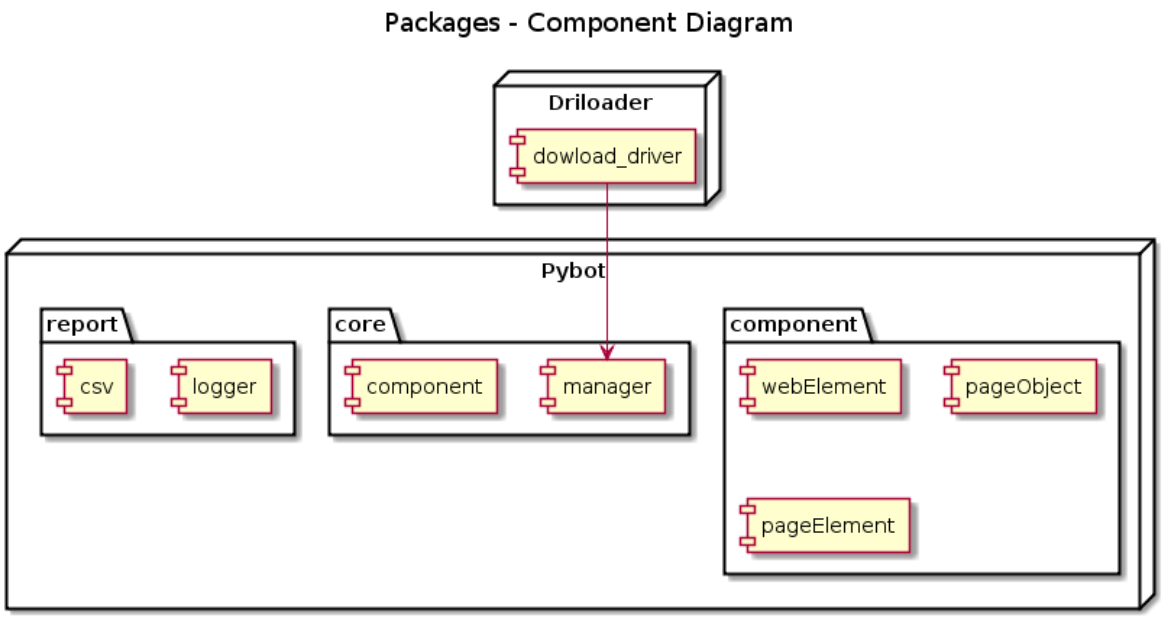
2.1.3 Git e GitHub

Para o controle de versões e alterações do codigo fonte do framework e scripts de exemplo foi utilizado a ferramenta Git (GIT, 2017) em conjunto com os servidores do Github (GITHUB, 2017) para hospedagem e gerenciamento. Com eles foi possivel fazer alterações dos codigos fontes em qualquer computador e gerenciar os erros e melhorias do framework.

2.2 MODULOS

O framework consiste em alguns modulos basicos, cada um com suas devidas utilidades e funções. A separação dos de cada modulos foi dada com base em suas caracteristicas e funcionalidades,

Figura 1 – Diagrama de Componentes



2.2.1 Core

Este modulo contem as funcionalidades basicas para a operação do framework com o Selenium Webdriver e o gerenciamento dos parâmetros de execuções de cada script.

2.2.1.1 Manager

Manager server para abstrair o uso do Selenium Webdriver criando uma camada de metodos propios fazendo com que caso alguma atualiação da API do Selenium Webdriver altere os scripts criados não sejam impactados. Fazendo uso do Driloader mencionado na subseção 2.2.4 ele verifica a necessidade do download do driver para poder executar o Selenium Webdriver.

2.2.1.2 Configuration

Responsavel por gerar as configurações basicas para o framework e disponibiliza-las no arquivo *pybot.ini*. Este arquivo é criado para cada script do usuario e nele arquivo é possivel adicionar quaisquer tipo de configurações ou parâmetros necessarias para o usuario, ape-

nas sendo necessario seguir os padrões descrito na imagem 2 e utilizando com o comando *configuration.getConfig('Seção', 'variavel')*

Figura 2 – Estrutura pybot.ini

```
1
2   [Seção]
3       variavel1 = 'valor'
4       variavel2 = 1
5       variavel3 = True
6
```

2.2.2 Component

Modulo criado para seguir os padrões de *PageObject* e *PageElement*, contendo abstração para os tipos de inputs do html.

2.2.2.1 WebElement

Serve para abstrair o uso da classe WebElement do proprio Selenium Webdriver. Contendo uma classe para cada tipo de campo dos html, ele dispõe de algumas funcionalidades basica, como a atribuição de uma valor para um elemento do tipo *input text* irá excrever valor dentro do campo, *select* irá selecionar a opção cujo texto seja igual ao valor informado, *radio* irá selecionar o a opção que tenha o *value* do valor informado e para o tipo *checkbox* irá marcar ou desmarcar as opções se o valor for verdadeiro(True) ou falso(False)

2.2.2.2 PageElement e PageElements

Essas classes servem para controlar os elementos mapeados das telas. Sempre quando serão acessadas a classe faz novamente a pesquisa do elemento em tela, previnindo assim uma das excessões mais comum do Selenium Webdriver que é a *StaleElementReferenceException*, que é quando o elemento em questão não existe mais no DOM ou a referencia que tinha não é mais a mesma. Conta com uma lista de seletores que facilitam para o usuario buscar os elementos e deixam o codigo mais legivel. Usa-se a classe PageElements quando quiser pegar mais de um elemento com o mesmo seletor.

Figura 3 – Lista de Seletores

```
class TestPage(PageObject):
    e01 = PageElement(css='#rso > div:nth-child(1) > div > div > div > h3 > a')
    e02 = PageElement(id_='id123')
    e03 = PageElement(name='name1')
    e04 = PageElement(xpath='//*[@id="palavras"]')
    e05 = PageElement(link_text='Texto')
    e06 = PageElement(partial_link_text='IFRS-Re')
    e07 = PageElement(tag_name='input')
    e08 = PageElement(class_name='class01')
```

2.2.2.3 PageObject

Classe simbolica, serve apenas para poder juntar diversos PageElement descritos pelo usuario em uma classe para melhor legibilidade e componentização das paginas mapeadas.

2.2.3 Report

Estes modulo está destinado para geração de logs de execuções internas do framework, criação e controle de logs definidos pelos usuario e a criação de planilhas análiticas de dados extraidos das paginas.

2.2.3.1 Logger

Classe de geração dos Logs de execução do framework.

2.2.3.2 CsvHandler

Utilizado para geração de planilhas com dados extraidos das paginas para análise posterior do usuario.

2.2.4 Driloader

Driloader é o responsavel pelo download dos driver de cada browser,suportanto download dos driver do Internet Explorer, Firefox e Chrome, sendo possivel para o usuario selecionar uma versão especifica, a ultima versão ou detectar automaticamente qual a versão adequada para o browser instalado do usuario. Como para utilizar do Selenium Webdriver é necessario um driver especifico de cada browser foi tomada a decisão da criação desse projeto, inicialmente o Driloader era um modulo do framework mas pela autonomia e praticidade que ele proporciona aos usuarios do Selenium Webdriver foi feita a separação dele do pybot.

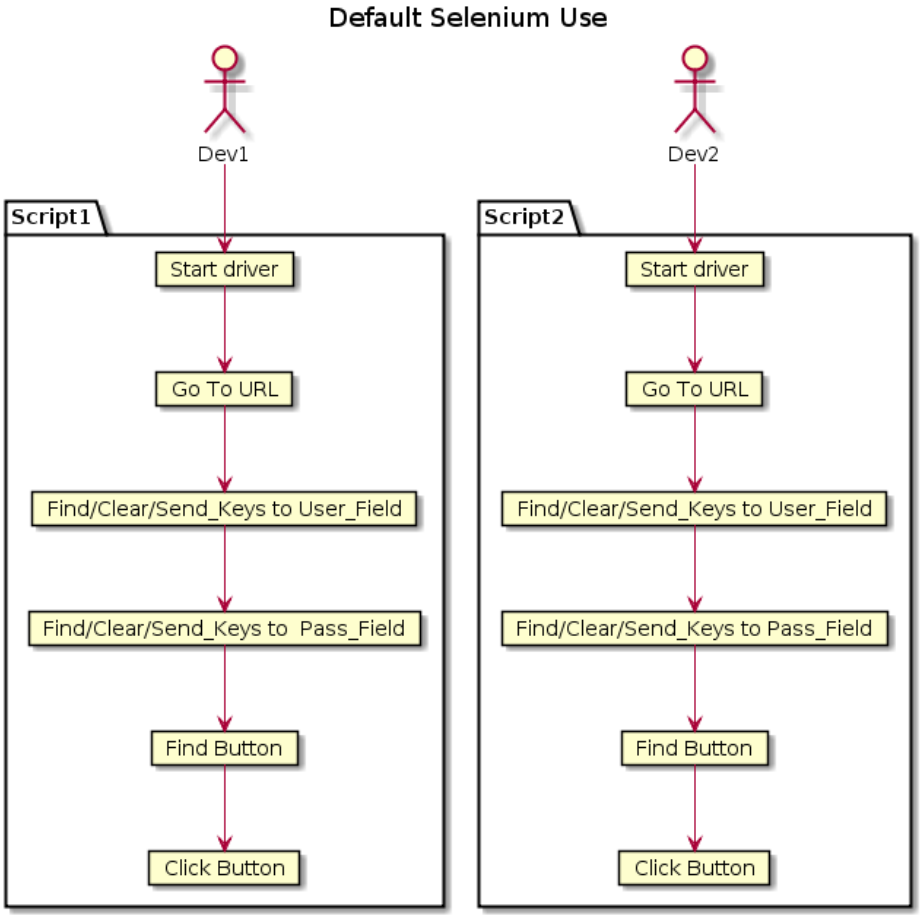
3 Metodo de Desenvolvimento Proposto

O framework é baseado no conceito de PageObject, onde todas as paginas web são tratadas como objetos e cada componente dela, que seja necessario para automação, um *input*, *span*, etc, é um atributo desse objeto.

Para melhor exemplificar o uso do conceito de PageObject usarei como exemplo um simples login para 2 usuarios, onde temos uma pagina que possue dois campos de texto, campo de usuario e outro de senha, e um botão para submeter o login.

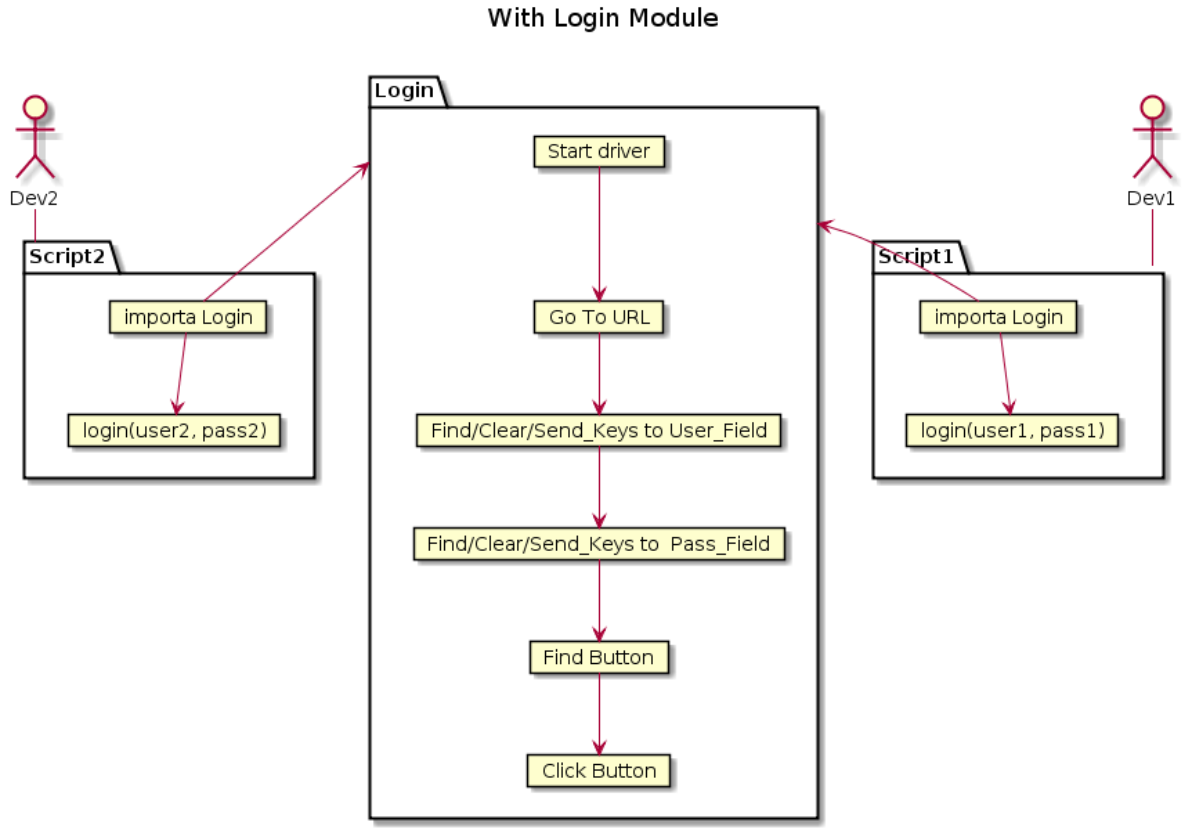
Primeiro temos um exemplo básico de como o Selenium propôe o desenvolvimento mostrado na figura 4. Primeiro é iniciado o driver do navegador, navegar para a URL, depois seguimos de 3 passos para cada um dos campos de texto, procurar ele, limpar o conteudo (porque não se sabe se ele já possui algum texto dentro) e enviar os caracteres necessario para cada campo e por final, procurar e clicar no campo de submeter. Não é uma metodo muito viavel, pois no caso temos 2 logins e o *script* deverá ser duplicado para atender ambas necessidades.

Figura 4 – Uso padrão Selenium



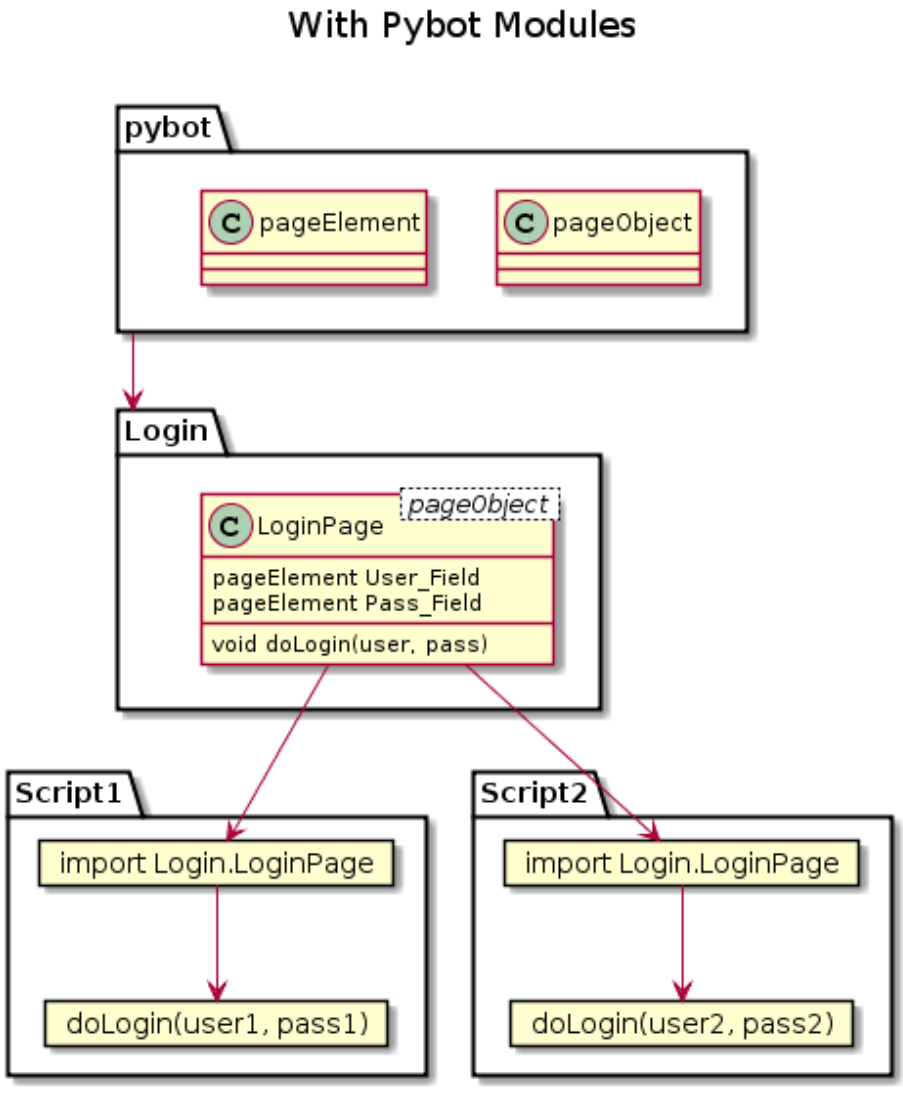
Num segundo exemplo poderiamos separar o *script* de login e criar um modulo separado para ele. Desse jeito os *scripts* podem fazer uso do mesmo codigo e caso uma terceira pessoa precise dele não seria um problema. Porem temos todo o mapeamento dessa pagina preso num modulo e caso seja necessario a criação de outro modulo que use esses campos ainda assim teremos que duplicar mais codigo.

Figura 5 – Uso padrão Selenium com modulo



Chegando num terceiro exemplo onde agora fazemos uso do framework *Pybot*, onde utilizando-se do modulo Component(2.2.2) podemos separar todos os componentes da tela em atributos da nossa pagina e criar um metodo onde precisando de dois parâmetros ele faz o processo de login, e ainda assim, caso necessario podemos ainda utilizar os campos mapeados para fazer algum outro metodo sem impactar o login. E caso alguma referencia dos campos mapeados mude, será necessario alterar apenas um local e nenhum *script* será impactado.

Figura 6 – Uso padrão Pybot



REFERÊNCIAS

GIT. **Git**. 2017. Disponível em: <https://www.git-scm.com>.

GITHUB. **Github**. 2017. Disponível em: <http://www.github.com>.

PYTHON. **The Python Tutorial**. 2017. Disponível em: <https://docs.python.org/3.6/tutorial/>.

WEBDRIVER, S. **The Python Tutorial**. 2017. Disponível em: <http://www.seleniumhq.org/docs/03_webdriver.jsp>.