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 alguma necessidade ou problema.

Esse novo framework pretende trazer aos usurários uma ferramenta de fácil 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 ferramentas que o framework dispõe são: gerenciamento automático dos controladores de navegadores(*drivers*), modulo de relatórios 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

SUMÁRIO

1 INTRODUÇÃO

O ciclo de vida de software tem diversas 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.

Atualmente possuem diversas ferramentas que possibilitam a criação de testes automatizados, desdes testes mais simples como a simples verificação de um campo de texto ou titulo, como para testes mais complicados como a verificação de uma funcionalidade de cadastro, por exemplo, ou até mesmo um teste de regressão, onde todas as funcionalidades e requisitos do software são testadas novamente para garantir que uma atualização do software não impacte em outras partes. Porem para fazer uso delas é necessário se preocupar com diversas coisas, e o custo inicial para começar a utiliza-las pode ser um pouco custoso.

Portanto, 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, contando com uma forma de utilização fácil e trazendo para si algumas das preocupações básicas que os usurários enfrentam ao utilizar de outras ferramentas. Levando o nome de PyBot, união das palavras Python, linguagem utilizada para criação do projeto, e Bot, que em inglês quer dizer robô, essa ferramenta propõe prover para os usurários uma serie de ferramentas para auxiliar a criação e implantação desses processos, contando com 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 automático de *drivers* de navegadores com a ferramenta Driloader.

2 TRABALHOS RELACIONADOS

Quando pesquisamos sobre automação de testes e processos em browser a primeira solução abordada é o *Selenium* (SELENIUM, 2017), trata-se de uma ferramenta com diversos projetos tais como *Selenium Grid*, *Selenium IDE*, *Selenium Remote Control* e o *Selenium WebDriver*, cada um com suas determinadas funcionalidades. Dentre eles o que mais se assemelham ao projeto proposto é o *Selenium WebDriver*, este por sua vez é também utilizado como uma das dependências do *Pybot* (3.1.2). E indo mais a fundo nas pesquisas e adicionamos junto do selenium o python, direcionando assim ela para o escopo do projeto, outra ferramenta que apresenta é o *robotframework* (ROBOTFRAMEWORK, 2017), este por sua vez é uma ferramenta bem mais complexa e abrangente.

O *Pybot* é para ser ferramenta simples e de fácil uso, portanto, para exemplificar mais o lugar onde ele se encontra foi feita uma análise melhor das funcionalidades e usos das ferramentas faladas anteriormente junto de um comparativo de alguns pontos fortes e fracos de cada uma delas.

2.1 SELENIUM WEBDRIVER

Selenium Webdriver (WEBDRIVER, 2017) trata-se de um framework onde ele disponibiliza para usuário uma API para integração com o browser. Com essa API é possível enviar diversos para o navegador tipos de comando tais como verificação e iteração qualquer tipo de elemento presente no DOM, geração de *prints* de tela e manipulação do próprio navegador.

Possui suporte as seguintes linguagens de programação: *Java*, *Csharp*, *Python*, *Ruby*, *Php*, *Perl* e *Javascript*, na sua maioria o suporte e funcionalidade são os mesmos para todas linguagens, porem o maior suporte é dado para a linguagem *Java*.

2.2 ROBOTFRAMEWORK

Robotframework (ROBOTFRAMEWORK, 2017) é um dos frameworks mais abrangentes que temos em *Python* para testes, consiste de uma vasta variedade de módulos separados do projetos, sendo assim possível optar por incluir eles ou não no projeto, dando também suporte a *Java* na maioria de seus módulos.

Sua arquitetura foi feita para executar testes utilizando a pratica de TDD possibilitando também uso de BDD e DDT, podendo criar marcadores para cada teste dando a versatilidade para o usuário executar determinados testes em especificas situações.

Conta também com:

- geração de relatórios de execução, tanto de sucesso como em falhas
- interface por linha de comando
- resposta de execução por XML

2.3 COMPARATIVO

Ambos frameworks utilizam em sua base o próprio *Selenium Webdriver*, portando todas as funcionalidades de integração com os navegadores estarão presentes neles.

Tabela 1 – My caption

	Selenium	Pybot	Robotframework
Integração com browser	Sim	Sim	Sim
Gerenciamento de drivers dos browser	Não	Sim	Não
Linguagens Suportadas	Java, Csharp, Python, Ruby, Php, Perl e Javascript	Python	Python e Java
Suporte à Técnicas de Desenvolvimento	Nenhuma	Nenhuma	TDD,BDD e DDT
Relatorios de Execução	Nenhum	Erro	Sucesso e Erro
Arquivo de Configuração	Não	Sim	Sim
Modular	Não	Não	Sim
Código Aberto	Sim	Sim	Sim

Fonte: Felipe Viegas, 2017.

3 Framework Proposto

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

3.1 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 possível apenas os módulos padrões do Python estão sendo utilizado para o desenvolvimento desta ferramenta.

3.1.1 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.

3.1.2 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* específico. 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*.

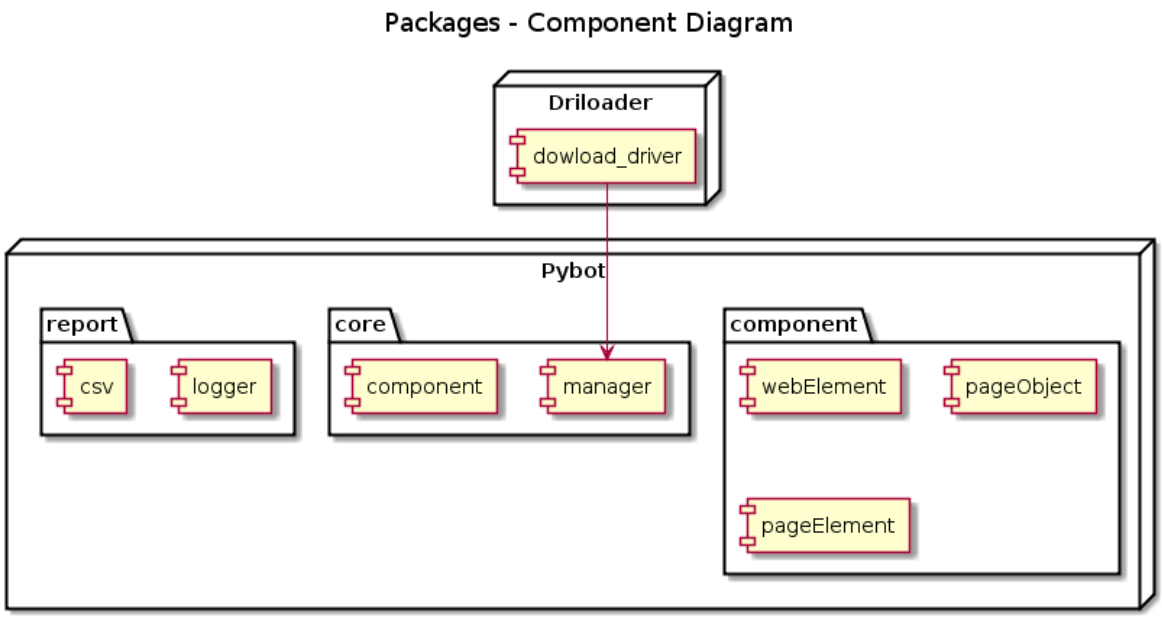
3.1.3 Git e GitHub

Para o controle de versões e alterações do código 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 possível fazer alterações dos códigos fontes em qualquer computador e gerenciar os erros e melhorias do framework.

3.2 MÓDULOS

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

Figura 1 – Diagrama de Componentes



3.2.1 Core

Este módulo contém as funcionalidades básicas para a operação do framework com o Selenium *Webdriver* e o gerenciamento dos parâmetros de execuções de cada script.

3.2.1.1 Manager

Manager server para abstrair o uso do Selenium *Webdriver* criando uma camada de metodos própios fazendo com que caso alguma atualização da API do Selenium *Webdriver* altere os *scripts* criados não sejam impactados. Fazendo uso do Driloader mencionado na subseção 3.2.4 ele verifica a necessidade do download do driver para poder executar o *Selenium Webdriver*.

3.2.1.2 Configuration

Responsável por gerar as configurações básicas para o framework e disponibilizá-las no arquivo *pybot.ini*. Este arquivo é criado para cada script do usuário e nele arquivo é possível adicionar qualquer tipo de configurações ou parâmetros necessárias para o usuário,

apenas sendo preciso seguir os padrões descrito na imagem 2 e utilizando com o comando *configuration.getConfig('Seção', 'variável')*

Figura 2 – Estrutura pybot.ini

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

3.2.2 Component

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

3.2.2.1 WebElement

Serve para abstrair o uso da classe WebElement do próprio Selenium Webdriver. Contendo uma classe para cada tipo de campo dos *html*, ele dispõe de algumas funcionalidades básica, como a atribuição de uma valor para um elemento do tipo *input text* irá escrever 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)

3.2.2.2 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, prevenindo assim uma das exceções mais comum do Selenium Webdriver que é a *StaleElementReferenceException*, que é quando o elemento em questão não existe mais no DOM ou a referência que tinha não é mais a mesma. Conta com uma lista de seletores que facilitam para o usuário buscar os elementos e deixam o código mais legível. 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')
```

3.2.2.3 PageObject

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

3.2.3 Report

Estes módulo está destinado para geração de logs de execuções internas do framework, criação e controle de logs definidos pelos usuário e a criação de planilhas analíticas de dados extraídos das páginas.

3.2.3.1 Logger

Classe de geração dos Logs de execução do framework.

3.2.3.2 CsvHandler

Utilizado para geração de planilhas com dados extraídos das páginas para análise posterior do usuário.

3.2.4 Driloader

Driloader é o responsavel pelo download dos driver de cada *browser*,suportando download dos driver do Internet Explorer, Firefox e Chrome, sendo possível para o usuário selecionar uma versão específica, a ultima versão ou detectar automaticamente qual a versão adequada para o *browser* instalado do usuário. Como para utilização do Selenium Webdriver é necessário um driver específico de cada *browser* foi tomada a decisão da criação desse projeto, inicialmente o Driloader era um módulo do framework mas pela autonomia e praticidade que ele proporciona aos usuários do Selenium Webdriver foi feita a separação dele do Pybot.

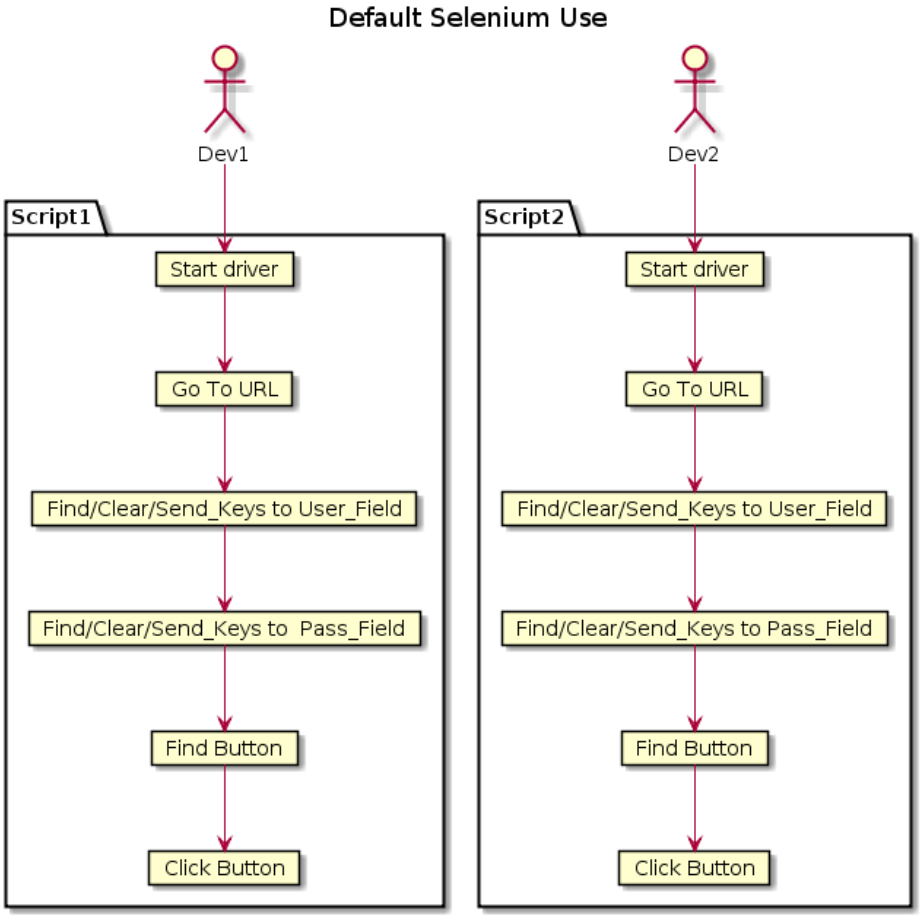
4 Método de Desenvolvimento Proposto

O framework é baseado no conceito de PageObject, onde todas as páginas web são tratadas como objetos e cada componente dela, que seja necessário 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 usuários, onde temos uma página que possui dois campos de texto, campo de usuário e outro de senha, e um botão para submeter o login.

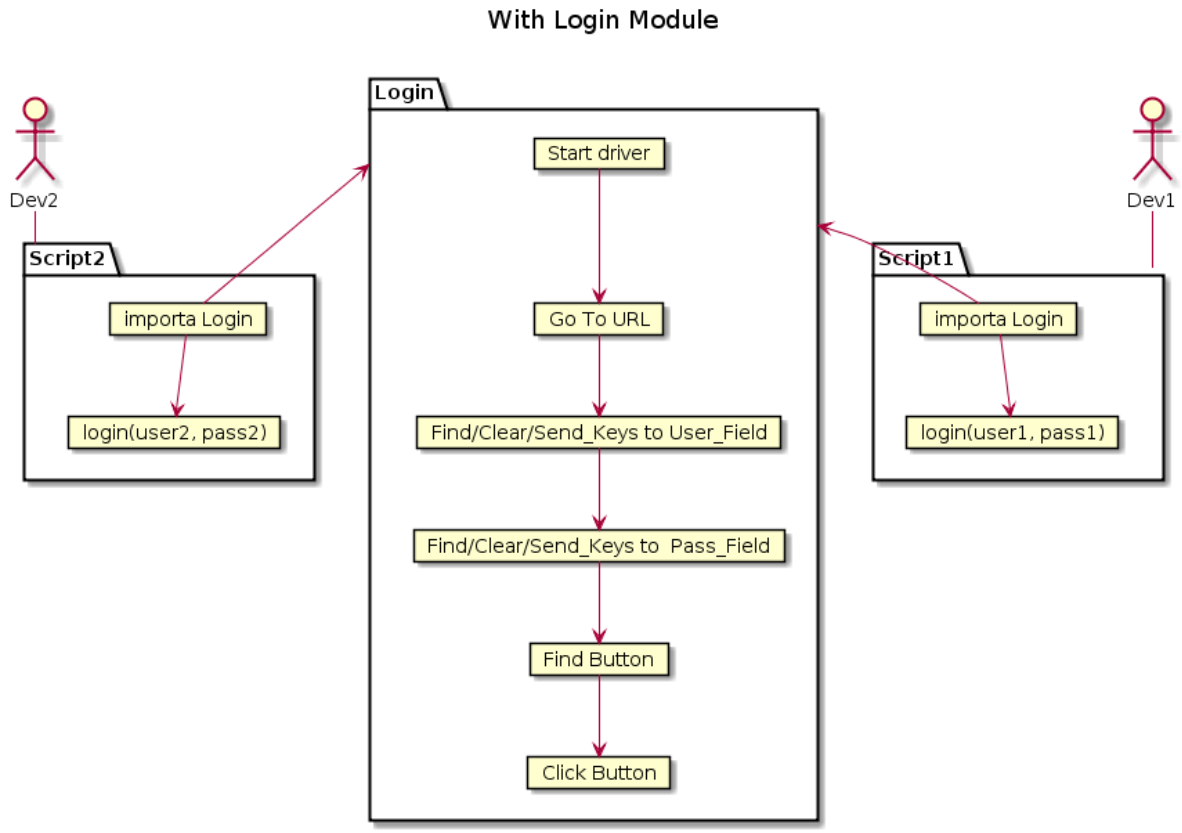
Primeiro temos um exemplo básico de como o Selenium propõe o desenvolvimento mostrado na figura 4. Primeiro é iniciado o *driver* do navegador, navegar para a URL, depois seguimos de 3 passos para cada um dos campos de texto, procurar ele, limpar o conteúdo (porque não se sabe se ele já possui algum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

Figura 4 – Uso padrão Selenium



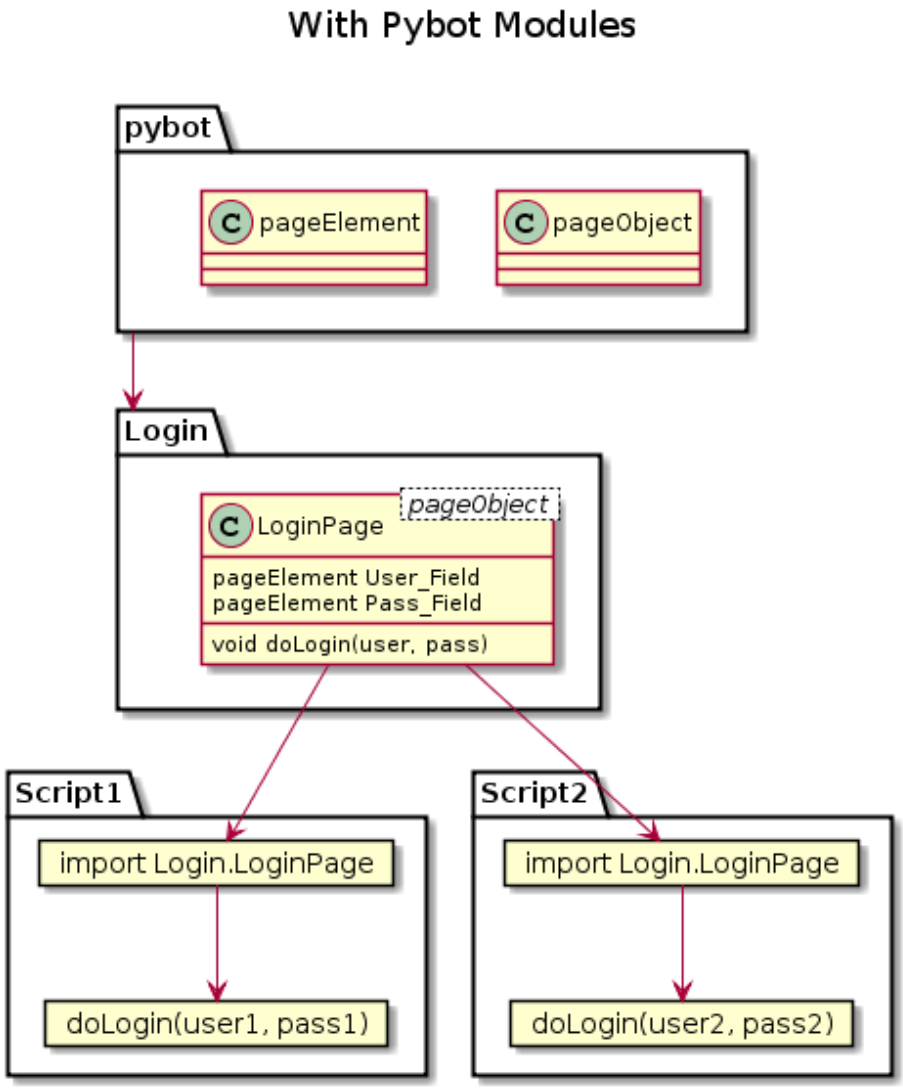
Num segundo exemplo poderíamos separar o *script* de login e criar um módulo separado para ele. Desse jeito os *scripts* podem fazer uso do mesmo codigo e caso uma terceira pessoa precise dele não seria um problema. Porém temos todo o mapeamento dessa página preso num módulo e caso seja necessário a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

Figura 5 – Uso padrão Selenium com módulo



Chegando num terceiro exemplo onde agora fazemos uso do framework *Pybot*, onde utilizando-se do módulo Component(3.2.2) podemos separar todos os componentes da tela em atributos da nossa página e criar um método onde precisando de dois parâmetros ele faz o processo de login, e ainda assim, caso necessário podemos ainda utilizar os campos mapeados para fazer algum outro método sem impactar o login. E caso alguma referencia dos campos mapeados mude, será necessário 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/>.

ROBOTFRAMEWORK. **robotframework**. 2017. Disponível em: <http://robotframework.org>.

SELENIUM. **Selenium**. 2017. Disponível em: <http://www.seleniumhq.org/>.

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