

**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

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

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL

Reitor: Prof. Osvaldo Casares Pinto

Pró-Reitora de Ensino: Profa. Clarice Monteiro Escott

Diretor do Campus Restinga: Prof. Gleison Samuel do Nascimento

Coordenador do Curso de Ciência da Computação: Prof. Rafael Pereira Esteves

Bibliotecária-Chefe do Campus Restinga: Paula Porto Pedone

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

## RESUMO

Em muitas empresas existe uma 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, porém a pratica dessas atividades só ganha força quando aparece alguma necessidade ou problema.

Para resolver este problema, propõe-se neste trabalho, um *framework* com o objetivo de trazer aos usuários uma ferramenta de fácil uso e com recursos úteis para o desenvolvimento dessas tarefas, contando com a facilidade e versatilidade da linguagem *Python* e a integração de navegadores com o *framework Selenium*.

O conjuntos de ferramentas que o *framework* proposto apresenta são: gerenciamento automático dos controladores de navegadores(*drivers*), módulo 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

Currently, many companies lack a solution for test automation or web process management integrated to the 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 emerge when a need or problem appears.

To solve this problem, we propose a framework aims to present to users an easy-to-use tool with useful resources for the development of these tasks, with the simplicity and versatility of the *Python* language and the integration with browsers with *selenium framework*.

The sets of tools that the proposed framework presents are: automatic management of the drivers of navigators (*drivers*); module containing reports and logs to control activities performed; standardization of screen elements development using the standard *PageObject* and identification of layout change.

**Palavras-chave:** Automation, Selenium, Tests.

## LISTA DE FIGURAS

Figura 1 – Diagrama de Componentes . . . . .	18
Figura 2 – Estrutura pybot.ini . . . . .	19
Figura 3 – Lista de Seletores . . . . .	20
Figura 4 – Uso padrão Selenium . . . . .	23
Figura 5 – Uso padrão Selenium com módulo . . . . .	24
Figura 6 – Uso padrão Pybot . . . . .	25
Figura 7 – Profissões . . . . .	26
Figura 8 – instalacao . . . . .	27
Figura 9 – uso . . . . .	27
Figura 10 – basico . . . . .	27
Figura 11 – avancado . . . . .	28



## **LISTA DE TABELAS**

Tabela 1 – Comparativo dos Frameworks . . . . .	16
---	----

## **LISTA DE ABREVIATURAS E SIGLAS**

API	Interface de Programação de Aplicação (do Inglês Application Programming Interface - API)
BDD	Desenvolvimento Orientado a Comportamento (do Inglês Desenvolvimento Guiado por Comportamento - BDD)
DDT	Desenvolvimento Orientado a Dados (do Inglês Data-driven testing - DDT)
DOM	Modelo de Documento Objeto (do inglês Document Object Model - DOM)
TDD	Desenvolvimento orientado a testes (do ingles Test Driven Development - TDD)

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>12</b>
<b>2</b>	<b>TRABALHOS RELACIONADOS . . . . .</b>	<b>14</b>
2.1	Selenium Webdriver . . . . .	14
2.2	Robotframework . . . . .	15
2.3	Comparativo . . . . .	15
<b>3</b>	<b>MÓDULOS . . . . .</b>	<b>18</b>
3.1	Core . . . . .	18
3.1.1	Manager . . . . .	18
3.1.2	Configuration . . . . .	18
3.2	Component . . . . .	19
3.2.1	WebElement . . . . .	19
3.2.2	PageElement e PageElements . . . . .	19
3.2.3	PageObject . . . . .	20
3.3	Report . . . . .	20
3.3.1	Logger . . . . .	20
3.3.2	CsvHandler . . . . .	20
3.4	Driloder . . . . .	20
<b>4</b>	<b>TECNOLOGIAS UTILIZADAS . . . . .</b>	<b>22</b>
4.1	Python . . . . .	22
4.2	Selenium WebDriver . . . . .	22
4.3	Git e GitHub . . . . .	22
<b>5</b>	<b>MÉTODO DE DESENVOLVIMENTO PROPOSTO . . . . .</b>	<b>23</b>
<b>6</b>	<b>RESULTADOS OBTIDOS . . . . .</b>	<b>26</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>29</b>
	<b>APÊNDICE A – PESQUISA DE SATISFAÇÃO DO PYBOT . . . . .</b>	<b>30</b>

## 1 INTRODUÇÃO

O ciclo de vida de software tem diversas etapas, das quais podem ser elencadas: Análise de requisitos, Concepção do Projeto, Desenvolvimento, Implantação e por fim Manutenção. Usualmente, nas etapas de Desenvolvimento e Manutenção ocorre a maior parte da criação ou a codificação do *software*, e na concepção de um projeto a necessidade da criação de um processo de testes que cresça junto do sistema não costuma ter a relevância necessária.

Atualmente, existem diversas ferramentas que possibilitam a criação de testes automatizados, desde testes mais simples, como a verificação de um campo de texto ou título em uma página web, até testes mais complicados, como 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. Porém, para fazer uso dessas ferramentas existentes é necessário avaliar diferentes questões previamente, como compatibilidade com sistema operacional, linguagem suportada e ferramentas oferecidas, e a curva de aprendizagem para começar a utilizá-las pode ser demorada e custosa.

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 desenvolvedores enfrentam ao utilizar outras ferramentas de testes. 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 usuários uma série de ferramentas para auxiliar a criação e implantação desses processos, contando com uma estrutura de criação dos *scripts* de testes. Para isso, a solução permitirá a criação de testes 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 Driload.

O restante do Trabalho de Conclusão de Curso é organizado da seguinte forma: O capítulo 2 serão apresentados os trabalhos relacionados, contando com 2 exemplos de ferramentas existentes e um comparativo dessas ferramentas e o framework proposto; O capítulo 3 irá apresentar os módulos presentes no framework com seu determinado propósito e funcionalidades; O capítulo 4 irá apresentar as tecnologias utilizadas para o desenvolvimento e controle do código fonte do framework; O capítulo 5 irá apresentar uma proposta de utilização do framework, fazendo uso de alguns conceitos apresentados. E por fim no capítulo 6 será apresentada uma análise

dos resultados obtidos de testes de usabilidade com usuários junto de futuras melhorias para o framework.

## 2 TRABALHOS RELACIONADOS

Conforme (SANTOS, 2016) a importancia da criação de processos automatizados cresce junto da importancia que os *softwares* vão tendo na sociedade, e as empresas tem uma certa dificuldade quando tentam de adotar ou implantar tais tipos de processos, devido a falta de profissionais com esse conhecimento ou pelas tecnicas presentes hoje.

A solução para automação de testes e processos em navegadores com maior adoção pela comunidade de desenvolvimento de software é o *Selenium* (SELENIUM, 2017). O *Selenium* trata-se de uma ferramenta composto por diversos projetos tais como *Selenium Grid*, *Selenium IDE*, *Selenium Remote Control* e o *Selenium WebDriver*, cada um com suas determinadas funcionalidades. Dentre os projetos citados, o *Selenium WebDriver* é o que mais se assemelha ao projeto proposto pois trata-se de um framework para integração com navegador, este por sua vez é utilizado como uma das dependências do Pybot(4.2).

Ainda, outra ferramenta também relacionada ao projeto proposto é o Robotframework, esta por sua vez é uma solução bem mais sofisticada, contendo uma quantidade abrangente de módulos e usos, porém junto traz uma complexidade maior para seu uso.

Similar a estes sistemas, a solução proposta neste Trabalho de Conclusão de Curso busca seguir direcionando a integração do *Selenium* com o *Python*. Porém, pretende-se com o Pybot o desenvolvimento ferramenta simples e de fácil uso, portanto.

Para entender melhor o problema, apresenta-se uma análise das funcionalidades e usos das ferramentas citadas 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 disponibiliza-se para usuário uma API para integração com o navegador escolhido. Com essa API é possível enviar diversos tipos de comandos para o navegador, tais como, verificação e iteração qualquer tipo de elemento presente no DOM (Modelo de Documento Objeto, do inglês Document Object Model), geração de *prints* de tela e manipulação do próprio navegador, como por exemplo maximizar a janela, redimensionar ou abrir uma nova janela. Todos os comando executados no navegador são comandos nativos de cada sistema operacional, sendo necessario possuir o driver específico para cada browser e sistema operacional para que funcione.

Ainda, possui suporte as seguintes linguagens de programação: *Java*, *Csharp*, *Python*,

*Ruby, Php, Perl e Javascript*. Na maioria dos casos, o suporte e funcionalidade são os mesmos para todas linguagens, porém o maior suporte é dado para a linguagem *Java*.

## 2.2 ROBOTFRAMEWORK

Robotframework (ROBOTFRAMEWORK, 2017) é um dos *frameworks* mais abrangentes disponíveis para teste de software para linguagem *Python*. Esta solução consiste de uma vasta variedade de módulos distintos, contando com 11 módulos próprio do framework e mais 30 de projetos parceiros, possíveis de serem habilitados. Desta forma, possível de optar por incluir determinado módulo ou não no projeto, tendo, também, suporte a *Java* na maioria de seus módulos.

Sua arquitetura foi feita para executar testes utilizando-se de ATDD (Acceptance Test Driven Development ou Desenvolvimento Orientado a Testes de Aceitação) que trata-se de uma abordagem ou prática para a criação de requisitos colaborativamente entre o cliente e a equipe e fazendo uso da técnica de desenvolvimento Ágil BDD (Behavior Driven Development ou Desenvolvimento Guiado por Comportamento em português) onde são criados cenários para cada tipo de comportamento da aplicação é criado levando em conta 3 estados, Dado que, Quando e Então, como no exemplo a seguir.

Exemplo: Listas com alguma coisa dentro não podem estar vazias

**Dado** que uma nova lista é criada

**Quando** eu adiciono um objeto a ela

**Então** a lista não deve estar vazia.

O Robotframework conta também com os seguintes recursos:

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

## 2.3 COMPARATIVO

Foram levantados alguns dos pontos mais relevantes para a execução dos *scripts* de testes de cada um dos framework e feito uma tabela comparativa entre eles. Ambos *frameworks* utilizam

em sua base o próprio framework *Selenium Webdriver*, portando todas as funcionalidades de integração com os navegadores estarão presentes neles e não serão tratadas no comparativo feito a seguir na tabela abaixo 1.

Tabela 1 – Comparativo dos Frameworks

	Selenium	Pybot	Robotframework
Integração com navegador	Sim	Sim	Sim
Gerenciamento de drivers dos navegadores	Não	Sim	Não
Linguagens Suportadas	Java, C#, Python, Ruby, Php, Perl e Javascript	Python	Python e Java
Supote ao conceito de Page Object	Não	Sim	Sim
Suporte à Técnicas de Desenvolvimento	Nenhuma	Nenhuma	ATDD e BDD
Relatorios de Execução	Nenhum	Sucesso/Erro	Sucesso/Erro
Verificação dinâmica dos elementos	Não	Sim	Sim
Arquivo de Configuração	Não	Sim	Sim
Modular	Não	Não	Sim
Código Aberto	Sim	Sim	Sim

Fonte: Autor desta monografia, 2017.

Como podemos ver, o diferencial é nas funcionalidades específicas de cada um, onde podemos perceber uma separação de níveis. Onde temos uma ferramenta super completa com módulos independentes e específicos para cada necessidade dando uma versatilidade maior para usuário em questão de funcionalidades, que é o caso do Robotframework, e por outro lado temos outros 2 *frameworks* mais básicos, onde o Pybot traz melhoria em relação ao selenium padrão, dando uma maior facilidade para os usuários no início das tarefas utilizando algumas



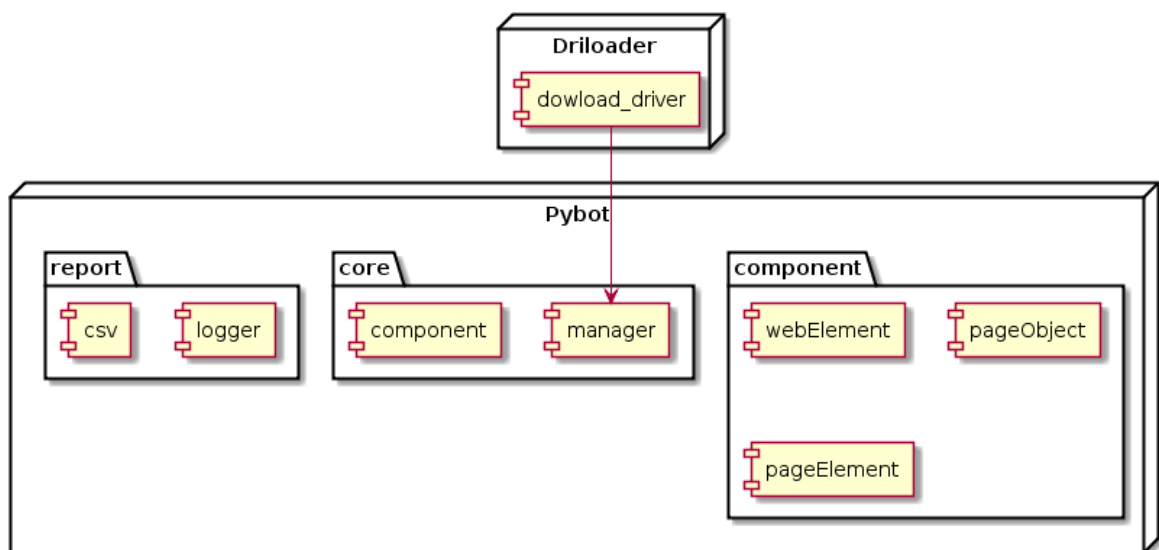
técnicas para agilizar e organizar o desenvolvimento dos *scripts*, relatorios de execução, arquivo de configurações para execução dos *scripts* e controle automatico dos drivers de cada navegador.

### 3 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 características e funcionalidades,

Figura 1 – Diagrama de Componentes

Packages - Component Diagram



Fonte: Autor desta monografia, 2017.

#### 3.1 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.

##### 3.1.1 Manager

Manager server para abstrair o uso do Selenium *Webdriver* criando uma camada de metodos próprios fazendo com que caso alguma atualização da API do Selenium *Webdriver* altere os *scripts* criados não sejam impactados. Fazendo uso do Driloder mencionado na subseção 3.4 ele verifica a necessidade do download do driver para poder executar o *Selenium Webdriver*.

##### 3.1.2 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

```

Fonte: Autor desta monografia, 2017.

## 3.2 COMPONENT

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

### 3.2.1 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 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

```

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

```

Fonte: Autor desta monografia, 2017.

### 3.2.3 PageObject

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

## 3.3 REPORT

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

### 3.3.1 Logger

Classe de geração dos Logs de execução do framework.

### 3.3.2 CsvHandler

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

## 3.4 DRILOADER

Driloader é o responsável pelo download dos *drivers* de cada *browser*, suportando *download* dos *drivers* do Internet Explorer, Firefox e Chrome, sendo possível para o usuário selecionar uma versão específica, a última 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

Driloder 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 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.

### 4.1 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.

### 4.2 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*.

### 4.3 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.

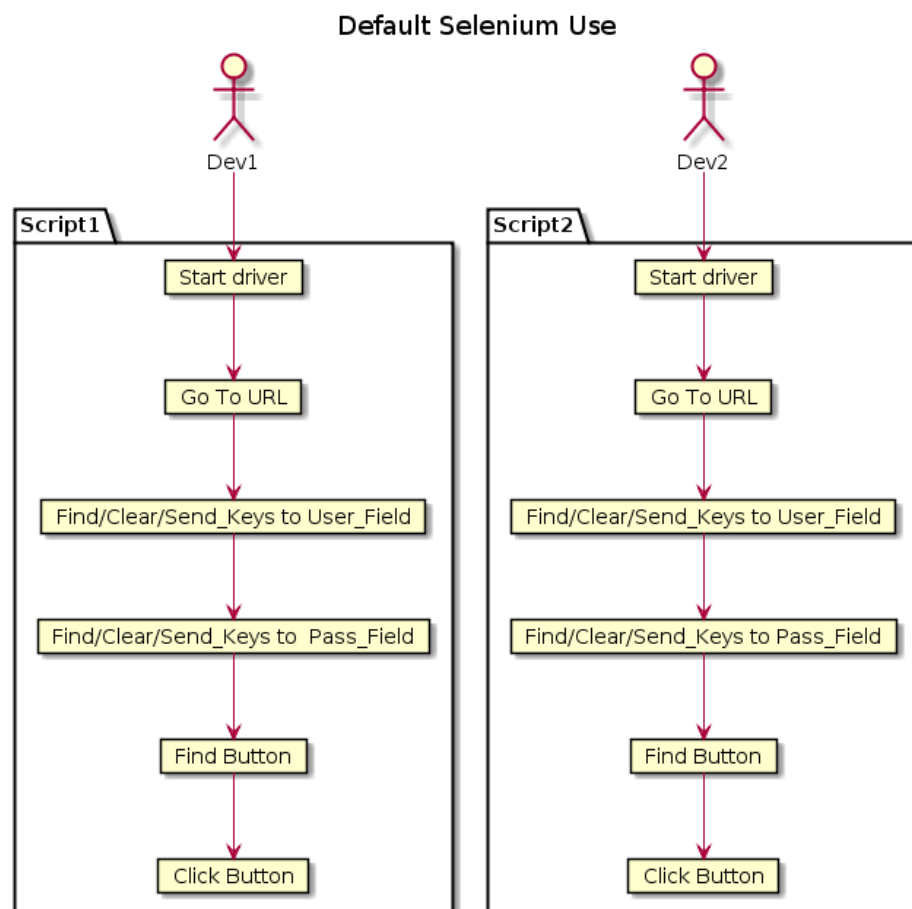
## 5 Método de Desenvolvimento Proposto

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

Para melhor exemplificar o uso do conceito de PageObject será utilizado como exemplo um simples login para 2 usuários, onde será utilizada uma página que possui dois campos de texto, campo de usuário e outro de senha, e um botão para submeter o login.

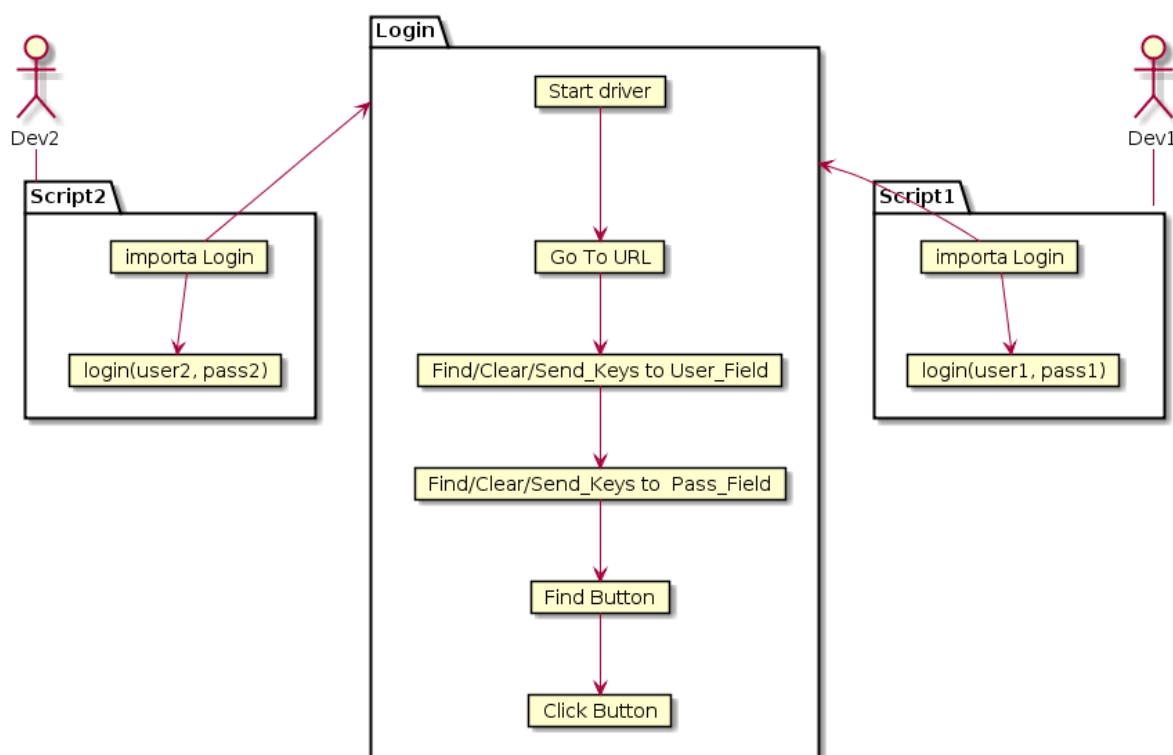
Primeiro, utilizou-se um exemplo básico de como o *Selenium* propõe o desenvolvimento mostrado na figura 4. Primeiro é iniciado o *driver* do navegador, navegar para a URL, depois são seguidos 3 passos para cada um dos campos de texto, procurar ele, limpar o conteúdo (porque não se sabe se ele possui algum texto pré cadastrado) 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 código 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 código.

Figura 5 – Uso padrão Selenium com módulo  
With Login Module

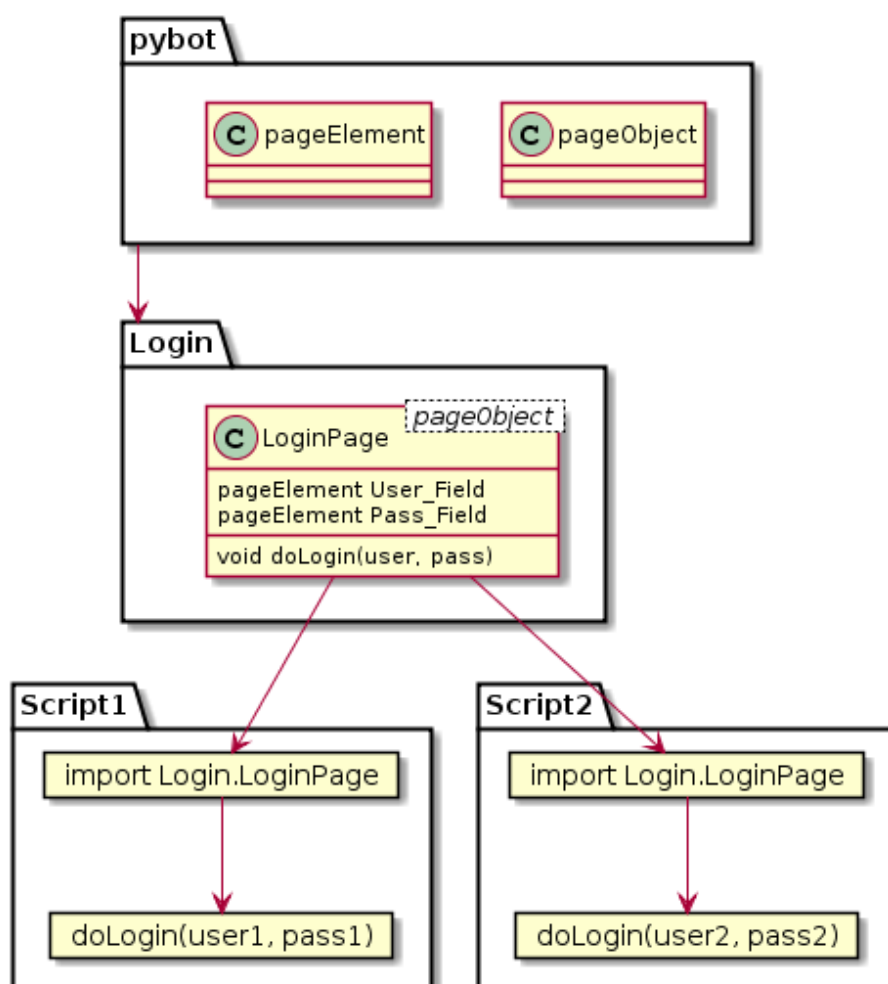


Fonte: Autor desta monografia, 2017.



Chegando num terceiro exemplo onde agora fazemos uso do framework *Pybot*, onde utilizando-se do módulo Component(3.2) 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 pode-se 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  
With Pybot Modules



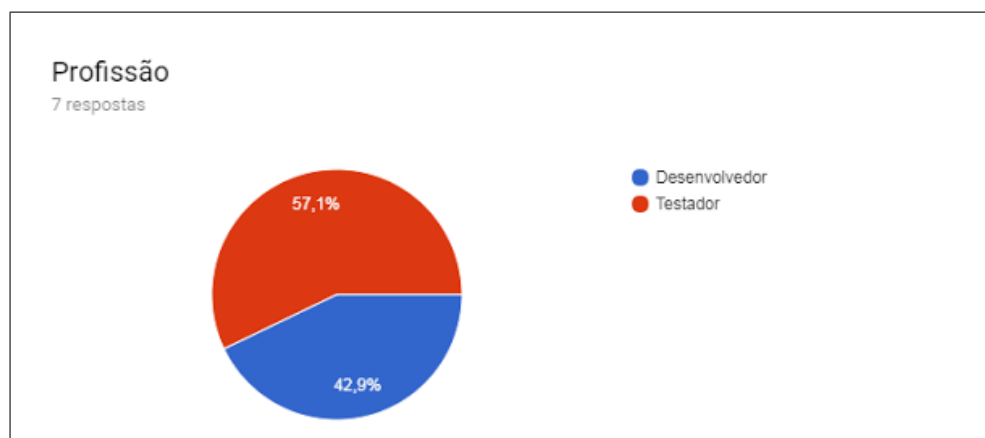
Fonte: Autor desta monografia, 2017.

## 6 Resultados Obtidos

Afim de validar se o framework proposto atende as necessidades dos usuários foi feito uma pesquisa de uso do framework com algumas pessoas da área da Programação. Através da plataforma *Google Forms* foram aplicados 5 perguntas sobre a experiencia dos usuários com o framework, foram levantados os seguintes pontos:

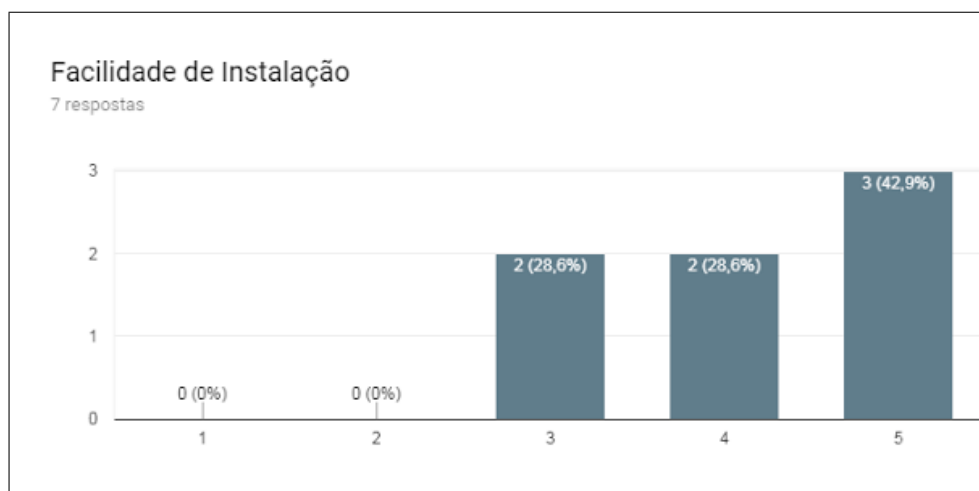
- a) Profissão do Usuario
- b) Facilidade de instalação
- c) Facilidade de utilização
- d) Atende as necessidades básicas
- e) Atende as necessidades avançadas

Figura 7 – Profissões



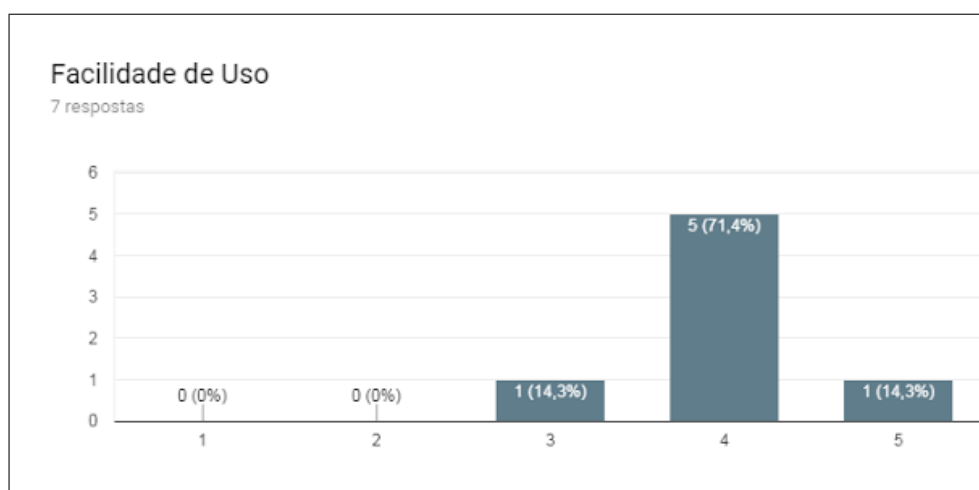
Fonte: Autor desta monografia, 2017.

Figura 8 – instalacao



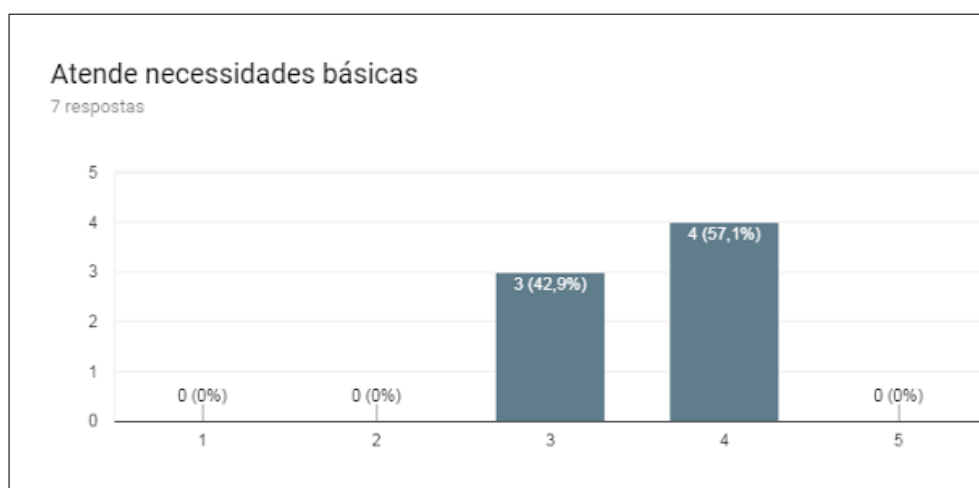
Fonte: Autor desta monografia, 2017.

Figura 9 – uso



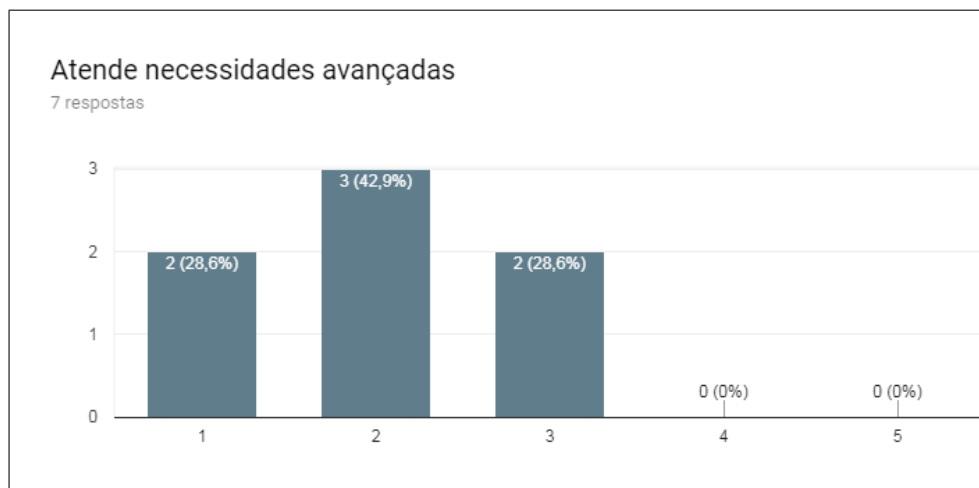
Fonte: Autor desta monografia, 2017.

Figura 10 – basico



Fonte: Autor desta monografia, 2017.

Figura 11 – avançado



Fonte: Autor desta monografia, 2017.

## 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>>.

SANTOS, M. d. O. dos. Um estudo sobre a influência das técnicas de testes automatizados no desenvolvimento de software. Universidade Federal do Amazonas, 2016.

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

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

## APÊNDICE A – Pesquisa de satisfação do Pybot

Profissão: ☐ Desenvolvedor ☐ Testador

Facilidade de Instalação: 1 ○ 2 ○ 3 ○ 4 ○ 5 ○

Facilidade de Uso: 1 ○ 2 ○ 3 ○ 4 ○ 5 ○

Atende necessidades básicas: 1 ○ 2 ○ 3 ○ 4 ○ 5 ○

Atende necessidades avançadas 1 ○ 2 ○ 3 ○ 4 ○ 5 ○

Comentarios: \_\_\_\_\_