PROGRAMA PRIME EXPERTS
PRIME CONTROL

Versão 1.0 - Junho/2021

TERMO DE CONFIDENCIALIDADE

Este documento contém informações confidenciais e destinam-se unicamente ao uso pelo (s) indivíduo (s) a quem são endereçados ou direcionados através do Programa Prime Experts da Prime Control.

É expressamente proibido qualquer reprodução total ou parcial, compartilhamento ou uso impróprio deste conteúdo sem autorização prévia por escrito da Prime Control.

Automação de Testes Robot Framework

NOSSA EQUIPE

HELDER FERNANDES

Atuante na área de tecnologia a mais de 8 anos em empresas nacionais e multinacionais. Boa parte desta atuação voltada para a área de Quality Assurance com diversas especializações em Automação de testes. Experiências com clientes de diversos nichos de mercado e atuação em diversas pontas do processo de QA como testes de aceitação, testes de performance, API, contrato, mainframe, automação móbile e Web.

NOSSA EQUIPE

JOSÉ CRISTIANO LUCENA

Experiência profissional superior há 8 anos em empresas nacionais e multinacionais, atuando nas áreas de tecnologia da informação, serviços e operações, em posições de Analista de Teste / Automação de teste, voltado para planejamento de testes, análise, modelagem, análise de risco, implementação, execução dos testes, verificação da qualidade do software e automação de testes. Desempenhando atividade de Analista de testes em projetos de alto impacto para grandes corporações como OI, Natura, Porto Seguro, Lopes, Riachuelo, Super Digital entre outros.

NOSSA EQUIPE

RODRIGO CANDIDO

Experiência profissional de 16 anos em TI, atuando como desenvolvedor, analista de sistemas/testes, coordenador e especialista. Conhecimentos em diversas tecnologias, linguagens de programação e frameworks de testes. Nos últimos 6 anos atuando com foco em disciplinas de testes, testes de performance, automações de páginas web, aplicativos desktop/mobile e webservices. Principais atuações: Itaú Unibanco, Via Varejo, Santander, Easynvest(Nu Investimentos), Guide Investimentos, Unimed Do Brasil, Unimed Seguros, Vivo, Nextel, Tenda Construtora, MC1, Sascar, Americanas e Prime Control.

Agenda

- **Instalações Iniciais**
  - O que é automação de testes?
  - Robot Framework
  - Editor de Código
  - Drivers
  - GitHub

Agenda

- **Introdução**
  - O que é
  - Sua arquitetura
  - Abordagem *Keyword-driven*
  - Estrutura
  - *Libraries*
  - Versionamento de código

Agenda

- **Básico Geral**
  - Criando sua primeira *Keyword*
  - Executando o teste
  - Variáveis
  - Argumentos
  - Retornos
  - FOR
  - IF
  - Setup e Teardown
  - Log e Report

Agenda

- ***Web Testing***
  - Conhecendo a *SeleniumLibrary*
  - Locators HTML
  - Interações:
    - *Text Field*
    - *Buttons*
    - *Checkbox*
    - *Radio Buttons*
    - *Lists*
  - Criando o primeiro Script de Teste Web

Agenda

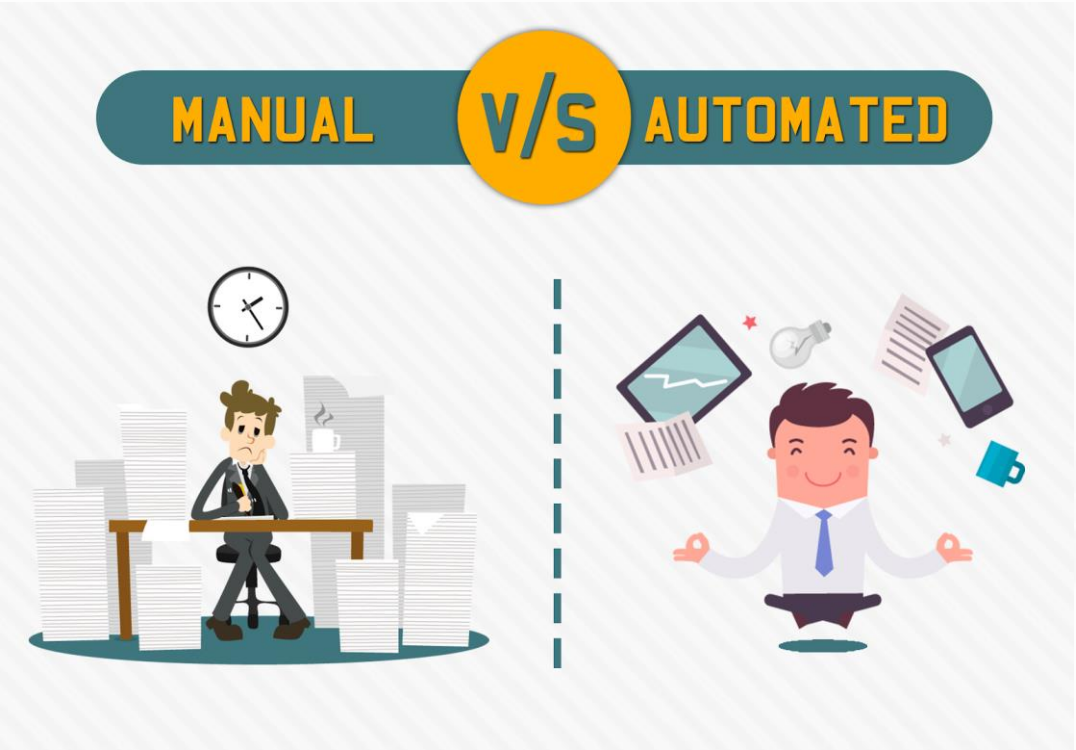
- ***Web Testing***
  - Organizando nosso código.
  - Usando Page Objects.
  - Criando um gerenciador de dependências.
  - Aprendendo a Debugar o código.
  - Automatizando a criação de massa.

Automação de Testes

Se aprofundando no conceito

Automação

Automação de teste é uma forma de aumentar a produtividade, diminuir custos e conseguir níveis de entrega em outra ordem de grandeza se comparado à processos manuais de testes.

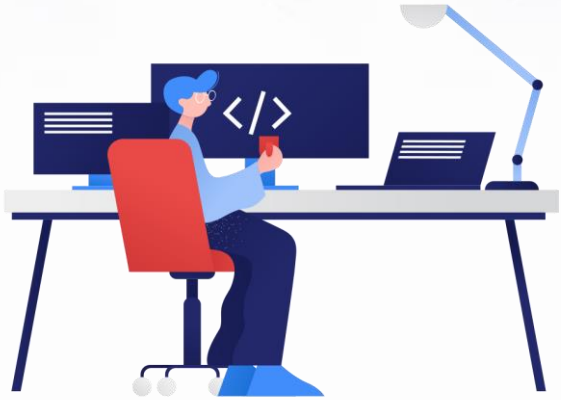


Automação e suas vantagens

- ✓Custo inferior
- ✓Feedback mais rápido
- ✓Maior eficiência
- ✓Maior motivação da equipe
- ✓Segurança de informação

Principais testes para automatizar:

- ✓Testes de regressão;
- ✓Tarefas repetitivas;
- ✓Funcionalidades críticas;
- ✓Testes com cálculos matemáticos.



Tudo é automatizável?

Não, na realidade uma grande quantidade de testes automatizados, não possui relação alguma com a qualidade do aplicativo ou site em questão.

A Automação pode contribuir e muito, porém somente se for utilizada de forma inteligente.

Princípios:

Um dos princípio que podem nos ajudar a entender isso melhor é o princípio do **Paradoxo do Pesticida** que diz:

“Pode ocorrer de um mesmo conjunto de testes que são repetidos várias vezes não encontrarem novos defeitos após um determinado momento.”

Princípios:

E não é por que temos automação que quer dizer que podemos automatizar tudo. **Teste exaustivo é impossível**:

“Testar tudo (todas as combinações de entradas e pré-condições) não é viável, exceto para casos triviais. Em vez do teste exaustivo, riscos e prioridades são levados em consideração para dar foco aos esforços de teste.”

Pensando nisso, quais as melhores práticas?

- A automação de testes não é um processo de testes
- Automatize os testes críticos primeiro
- Incorpore testabilidade ao aplicativo
- As ferramentas de automação de testes também têm defeitos
- Demo não é prova de conceito
- Dimensione a infra-estrutura adequadamente
- Encare a automação de testes como um projeto
- Alinhe as expectativas e garanta a colaboração de todos os envolvidos
- A aut. de testes é um investimento de longo prazo
- O teste manual é insubstituível

Robot Framework

Introdução

O que é o Robot:

Framework genérico que permite automação de qualquer tipo de sistema (web, API, mobile, desktop, etc), baseado em *keyword-driven* que abstrai a camada de programação em baixo nível.

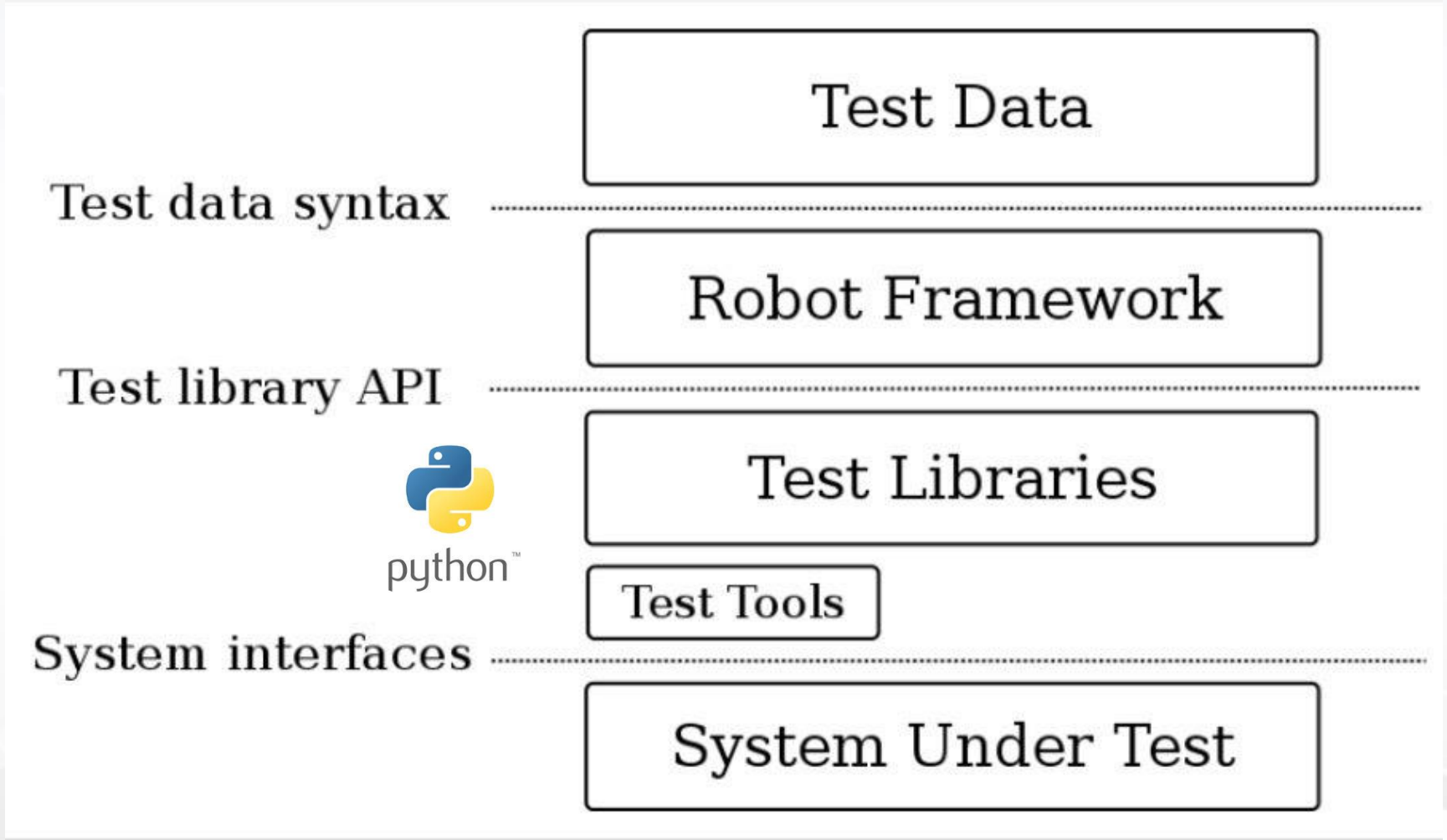
Sobre

- O **Robot Framework** foi desenvolvido em Python e inicialmente foi feito para a **Nokia**.
- É ***Open Source*** e **Multiplataforma**.
- Além de abranger diversos tipos de automações de testes, ele também automatiza processos (RPA).
- Possui sintaxe de dados tabular fácil de usar e permite que usuários utilizem e criem bibliotecas em Python (se precisar).
- **Não!** Você não precisa saber Python para usar, já existem diversas bibliotecas com *keywords* prontinhas para você escrever seus testes!

Tipos de Testes

- **Existem *libraries* que suportam testes para:**
  - Web
  - API
  - Mobile
  - GUI
- **Estilos de testes:**
  - *Keyword-driven* (formato procedural)
  - *Data-driven* (tabelas de dados)
  - *Gherkin* (BDD)
- **Executado por linha de comando.**
- **Possível executar em *Docker*, *Jenkins*, *Cross-Browser*.**

Arquitetura



A abordagem *keyword-driven*

- É uma abordagem de testes automatizados baseada em ação.
- As palavras-chaves (ou ***keywords***) são em alto nível, praticamente em linguagem nativa, que representa uma ação do usuário.
- Encapsula a implementação (baixo nível) do teste.
- Bom para testadores não técnicos.
- Boa Reutilização.
- Fácil aprendizado.
- Fácil escrita e leitura dos testes.

A abordagem *keyword-driven*

Exemplo: Abrir uma página web

Implementação (Python com Selenium WebDriver):

```
def setUp(self):
    self.driver = webdriver.Firefox()
    self.driver.implicitly_wait(20)
    self.base_url = "https://www.facebook.com/"
    self.verificationErrors = []
    self.accept_next_alert = True
```

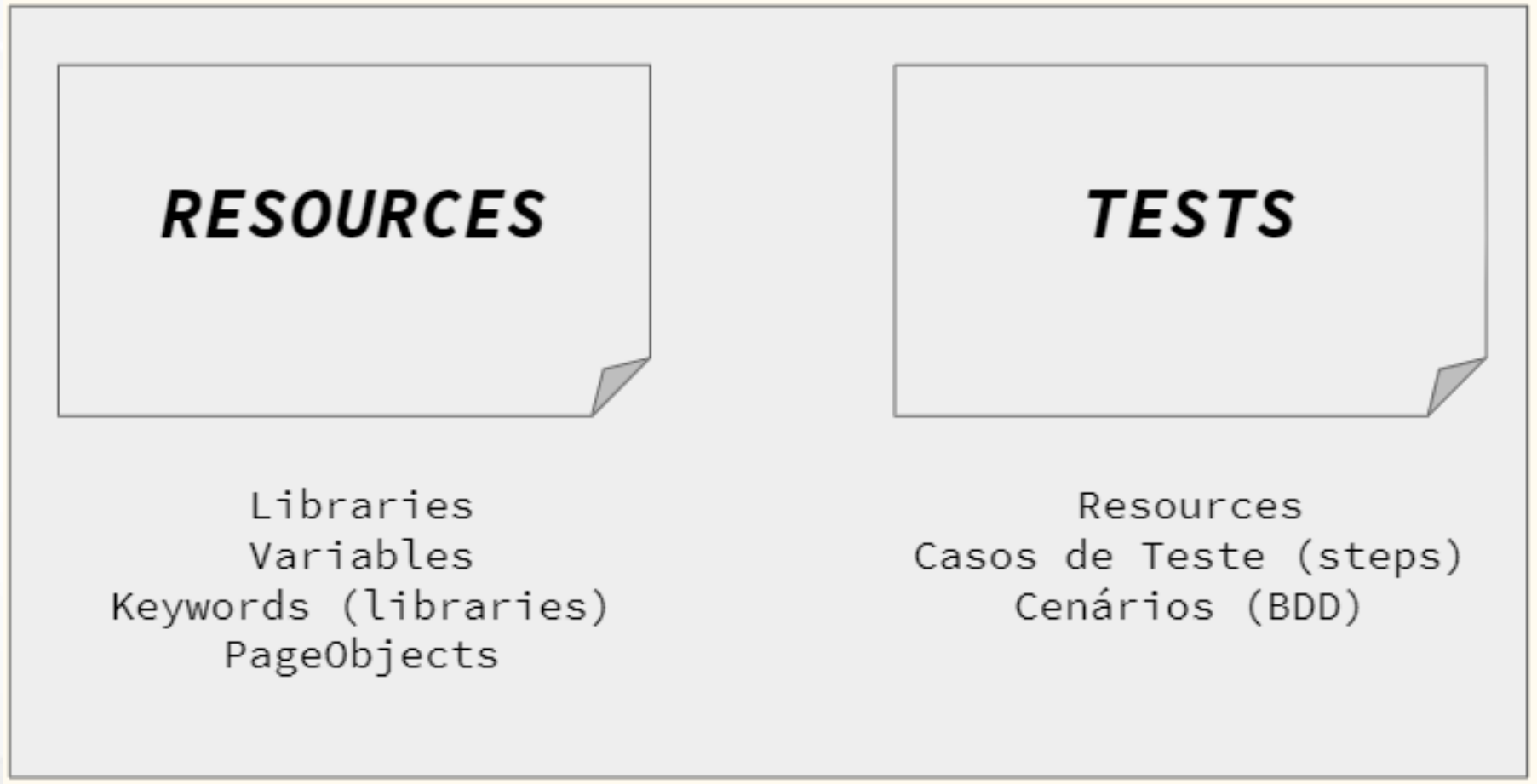
***Keyword* (RobotFramework com SeleniumLibrary):**

```
Open Browser    https://www.facebook.com/    firefox
```

Robot Framework

Estrutura

Arquivos Base



Seções

***** Settings *****	Na seção Settings podemos informar documentação (Documentation), as bibliotecas (Library), os scripts de baixo nível (Resources), setup/teardown da suíte e dos testes e timeout para os testes.
***** Variables *****	Com o nome já diz, é nesta seção que iremos declarar variáveis e definir os valores default. Exemplo: **${URL}** http://www.google.com.br
***** Test Case *****	Nessa seção escrevemos os casos de teste em **linguagem natural**, em *keywords*. Exemplo: **Validar login válido** Acessar homepage Informar usuário "HelderFernandes" Informar senha "12345" Submeter login
***** Keywords *****	A seção *Keywords* é onde implementamos os passos (*keywords*) escritas na seção Test Case. Exemplo: **Informar usuário "${NOME_USUARIO}"** Input Text ${CAMPO_USUARIO} ${NOME_USUARIO}

Libraries

- ***Standard Libraries (nativas)***
  - BuiltIn
  - OperatingSystem
  - Process
  - Collections
  - DateTime
  - String

- ***External Libraries (bibliotecas open source)***
  - AppiumLibrary (testes mobile com Appium)
  - SeleniumLibrary (testes web com Selenium)
  - RequestsLibrary (testes de API)
  - SikuliLibrary (testes de GUI com reconhecimento de imagens)
  - DatabaseLibrary (banco de dados)

- **Crie suas próprias bibliotecas (*Python*)**

Referências

- Site Oficial: http://robotframework.org/
- Blog Brasileiro: http://robotizandotestes.blogspot.com.br/
- Slides Inglês Introdução: https://www.slideshare.net/pekkaklarck/robot-framework-introduction
- Doc. da SeleniumLibrary: http://robotframework.org/SeleniumLibrary/SeleniumLibrary.html
- Slack Global: https://robotframework-slack-invite.herokuapp.com/
- Telegram Brasil: https://t.me/joinchat/Q5qH9xePmaaSC5hh
- Exemplos: https://github.com/mayribeirofernandes/testesrobotframework
- Práticas de Automação - https://www.devmedia.com.br/automacao-de-testes/10249
- Práticas de programação - https://www.inf.pucrs.br/

Robot Framework

Básico Geral

Keywords

Library SeleniumLibrary		
Keyword	**Parâmetros**	**O que ela faz**
Open Browser	url, browser=firefox [...]	Abre uma nova instância do navegador para a url informada. O argumento *browser* especifica qual navegador irá usar.
Click Element	localizador	Clica no elemento identificado pelo localizador.
Close Browser	-	Fecha o browser atual.
Element Text Should Be	localizador, texto esperado [...]	Verifica se esse localizador de elementos contém exatamente o texto esperado.
Wait Until Element Is Visible	localizador, timeout=None	Espera até que o elemento do localizador esteja visível. Falha se o *timeout* expirar antes que o elemento esteja visível.

Comandos de Execução

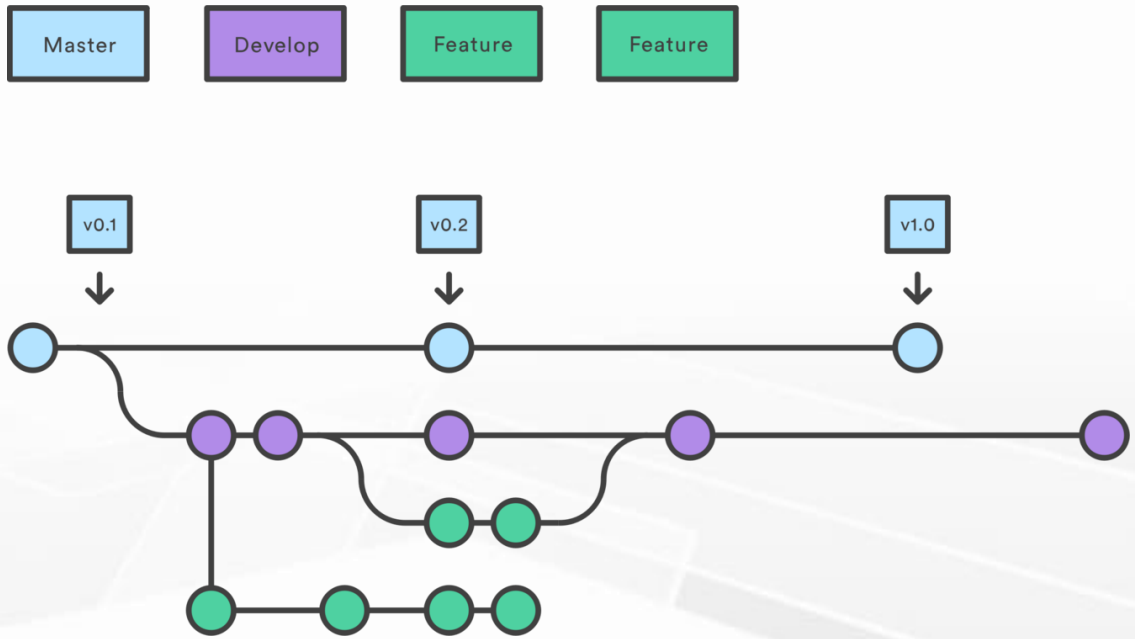
Comando	O que faz
robot meuDiretorioDeTestes	Roda todas as suítes contidas dentro do ***meuDiretorioDeTestes***
robot meusTestes.robot	Roda apenas a suíte ***meusTestes***
robot -t "Cenário 01: Pesquisar Produtos" meusTestes.robot	Roda somente o teste chamado ***"Cenário 01: Pesquisar Produtos"*** da suíte ***meusTestes***
robot -i "smoke tests" meusTestes.robot	Roda somente os testes que tiverem a **TAG *"smoke tests"*** da suíte ***meusTestes***
robot -v BROWSER:ie meusTestes.robot	Roda a suíte ***meusTestes*** porém trocando o valor da variável **BROWSER**
robot -L trace meusTestes.robot	Roda a suíte ***meusTestes*** em modo de **LOG LEVEL = TRACE (INFO, DEBUG)**
robot --help	Digite esse comando para ver todos os possíveis parâmetros de execução e a explicação de cada um.

Versionamento de Código

Um sistema de versionamento permite que várias pessoas trabalhem no mesmo conjunto de arquivos (repositório) ao mesmo tempo em que evita conflitos entre as alterações.

Cada membro do time de desenvolvimento tem sua "cópia" dos arquivos que ao final das alterações é colocada junto das versões alteradas dos demais.

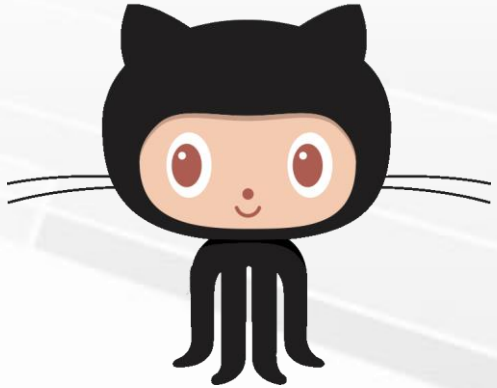
git



GitHub

O GitHub é um serviço na nuvem onde podemos armazenar nossos códigos criados e controlar a versão desses códigos. O código pode ficar visível publicamente e outras pessoas desenvolvedoras podem colaborar.

Durante as aulas vamos aprender a usar e subir nossos códigos para o GitHub!



Principais comandos do git

Clonar um repositório:

- *git clone <link do seu projeto>*

Criar nova branch:

- *git branch <nome da branch>*

Criar nova branch a partir da branch atual:

- *git checkout –b <nome da nova branch>.*

Adicionamos todas as alterações ao pacote de envio:

- *git add .*

Criar nova branch:

- *git commit -m "<Mensagem descrevendo o que foi feito>"*

Baixar todas as atualizações da branch atual:

- *git pull*

Enviando as atualizações para o repositório:

- *git push*
- *Branch local – git push origin <nome da branch>*

Verifica o status da branch recorrente:

- *git status*

Lista as branchs existentes:

- *git branch*

Variáveis

- ***Simples***

  ${NOME} Maria

- ***Dicionários***

  &{PESSOA} nome=João sobrenome=Silva idade=15

- ***Listas***

  @{FRUTAS} abacaxi laranja morango banana

- ***Configurando o Escopo***

Exercícios - Variáveis

1. Crie uma variável do **Tipo Dicionário** que conterá dados de uma pessoa, com no mínimo 6 informações e imprima no console, uma informação por vez.

2. Crie uma variável do **Tipo Lista** de 05 frutas e imprima no console, uma por vez.

Argumentos e Retornos

```
[...]
*** Keywords ***
Somar dois números
        [Arguments]    ${NUM_A}    ${NUM_B}
        ${SOMA}              Evaluate      ${NUM_A}+${NUM_B}
        [Return]  ${SOMA}


[...]
*** Test Cases ***
Meu teste de soma
        ${RESULTADO}    Somar dois números    1    2
        Log    ${RESULTADO}
```

Argumentos Embutidos

```
[...]
*** Keywords ***
Somar os números "${NUM_A}" e "${NUM_B}"
        ${SOMA}         Evaluate    ${NUM_A}+${NUM_B}
        [Return]  ${SOMA}

[...]
*** Test Cases ***
Meu teste de soma
        ${RESULTADO}  Somar os números "1" e "2"
        Log    ${RESULTADO}
```

Exercícios - Argumentos e Retornos

1. Crie uma *keyword* que cria um e-mail formado por **nome_sobrenome_idade@robot.com**, onde o **nome**, o **sobrenome** e a **idade** são recebidos via passagem de argumentos e, ao final, a *keyword* deve retornar esse e-mail formatado. Imprima o e-mail retornado no console.

FOR

```
[...]
*** Keywords ***
Contar de 0 a 9
        FOR    ${count}    IN RANGE    0    9
                Log    ${count}
        END
[...]
*** Keywords ***
Percorrer itens de uma lista
        @{FRUTAS}    Create List    morango    banana    maça
        FOR    ${fruta}    IN    @{FRUTAS}
                Log    ${fruta}
        END
```

Exercícios - FOR

1. Crie uma *keyword* que imprima no console a frase "**Estou no número: ${numero}**" de **0** a **10**.

2. Crie uma *keyword* que imprima no console **5** nomes de países.

IF

[...]

****** Keywords ******

Tomar decisões

```
IF    '${NOME}'=='Maria'
        Log    Vou fazer isso aqui só quando for a Maria!
ELSE IF    '${NOME}'=='João'
        Log    Vou fazer isso aqui só quando for o João!
ELSE
        Log    Vou fazer isso aqui só quando for qualquer outra pessoa!
END
```

Exercícios - IF

1. Crie uma *keyword* que imprima no console a frase "**Estou no número: ${numero}**" de **0** a **10**, porém somente deve imprimir se for o número **5** ou **8**.

Setup e Teardown

Suite Setup: Uma *keyword* específica será executada **ANTES** de começar a execução da suíte.

Test Setup: Uma *keyword* específica será executada **ANTES** de começar a execução de cada teste.

Suite Teardown: Uma *keyword* específica será executada **DEPOIS** de encerrar a execução da suíte.

Test Teardown: Uma *keyword* específica será executada **DEPOIS** de encerrar a execução de cada teste.

Log e Report

LOG: Use para saber com detalhes como cada passo foi executado no seu teste/suíte.

REPORT: Use para saber o status e resultados da execução realizada.