Automatize com Precisão: Guia Prático de Automação de Testes com Robot Framework

Autor: Vinícius Alves Jerônimo • Ano: 2025

Introdução

A automação de testes é essencial para entregar software com qualidade e previsibilidade.

Com o Robot Framework você escreve testes legíveis, reaproveitáveis e com relatórios nativos, acelerando o ciclo de feedback e reduzindo custos de manutenção.

O que é o Robot Framework?

O Robot Framework é um framework open■source para testes de aceitação e automação de processos (RPA).

Ele é baseado em palavras■chave (keywords), extensível por libraries (Python/Java/.NET) e possui relatórios de execução em HTML por padrão.

Principais vantagens:

- Sintaxe simples (tabelas/Markdown/.robot)
- Grande ecossistema: SeleniumLibrary, RequestsLibrary, DatabaseLibrary, Browser, SSHLibrary etc.
- Integração com CI/CD: Jenkins, GitHub Actions, GitLab, Azure DevOps
- Cross■platform (Windows, macOS, Linux)

Instalação e Configuração

Pré■requisitos: Python 3.8+ e pip.

Instalação do core:

pip install robotframework

Libraries úteis:

```
# Web (Selenium)
```

pip install robotframework-seleniumlibrary webdriver-manager

```
# API
```

pip install robotframework-requests

```
# Playwright (Browser library)
```

pip install robotframework-browser

rfbrowser init

Dica: crie um virtualenv para isolar dependências.

Estrutura de Projeto (sugerida)

```
project/
■■ tests/                # cenários de teste
■  ■■ web/
■  ■  ■■ test_login.robot
■  ■■ api/
■  ■  ■■ test_users.robot
■  ■■ resources/
■     ■■ keywords.robot   # keywords reutilizáveis
■     ■■ variables.robot  # variáveis globais (URLs, credenciais fake etc.)
■■ results/              # relatórios/artefatos de execução
■■ requirements.txt
■■ README.md
```

Primeiro teste Web (SeleniumLibrary)

```
*** Settings ***
Library    SeleniumLibrary

*** Variables ***
${URL}         https://example.com/login
${USERNAME}    usuario_demo
${PASSWORD}    senha_demo

*** Test Cases ***
Login Com Sucesso
    Open Browser    ${URL}    chrome
    Input Text      id:username    ${USERNAME}
    Input Text      id:password    ${PASSWORD}
    Click Button    id:login-button
    Page Should Contain    Bem-vindo
    [Teardown]    Close All Browsers
```

Teste de API (RequestsLibrary)

```
*** Settings ***
Library    RequestsLibrary

*** Variables ***
${BASE_URL}    https://api.example.com

*** Test Cases ***
Validar Lista de Usuários (200)
    Create Session    api    ${BASE_URL}
    ${resp}=    Get Request    api    /users
    Should Be Equal As Numbers    ${resp.status_code}    200
    Should Contain    ${resp.text}    "results"
```

Executando e Gerando Relatórios

Para executar todos os testes dentro de tests/ e salvar artefatos em results/:

robot -d results tests/

Saídas geradas:

```
- report.html  (resumo executivo)
- log.html     (detalhamento passo a passo)
- output.xml   (máquina/integrações)
```

Tags, Seleção e Organização

```
Use tags para agrupar e filtrar:
    *** Test Cases ***
    Login Com Sucesso
        [Tags]    smoke    web

Executar apenas smoke:
    robot -i smoke -d results tests/

Ignorar um conjunto:
    robot -e lento -d results tests/
```

Boas Práticas

```
- Separe **keywords** em resources/ para reuso e clareza.
- Centralize variáveis (URLs, dados fake) em variables.robot.
- Nomeie casos com linguagem de negócio.
- Use **Setup/Teardown** para abrir/fechar navegador/sessões.
- Padronize logs/screenshots (Selenium: Capture Page Screenshot em falhas).
- Valide mensagens e contratos (status code, schema, campos obrigatórios).
- Mantenha testes determinísticos (dados previsíveis) e independentes.
```

Robot + CI/CD (GitHub Actions)

Exemplo de workflow (.github/workflows/robot.yml):

name: Robot Tests

on: [push, pull_request]

jobs:

tests:

runs-on: ubuntu-latest

steps:

```
          - uses: actions/checkout@v4
          - uses: actions/setup-python@v5
```

with:

python-version: '3.11'

```
          - run: pip install -r requirements.txt
          - run: robot -d results tests
          - uses: actions/upload-artifact@v4
```

with:

name: robot-results

path: results

Troubleshooting Rápido

- Navegador não abre? Verifique driver/versão (ou use webdriver-manager).
- Elemento não encontrado: adicione esperas explícitas (Wait Until Element Is Visible).
- Timeout frequente: aumente timeout default (Set Selenium Timeout) e estabilize seletores.
- API flakey: reintente com lógica de retry e valide apenas o necessário.
- Pipeline falhando: faça upload dos artefatos (report.html/log.html) para diagnóstico.

Recursos

- Site oficial: https://robotframework.org
- User Guide: https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html
- SeleniumLibrary: https://robotframework.org/SeleniumLibrary/
- RequestsLibrary: https://marketsquare.github.io/robotframework-requests/
- Browser (Playwright): https://robotframework-browser.org/

Conclusão

Com o Robot Framework você entrega qualidade em ritmo ágil, com testes legíveis, relatórios nativos e uma comunidade madura.

Use este guia como base para estruturar seu projeto, criar seus primeiros testes e integrar ao pipeline.

Bons testes!