# **Atividade Prática de Testes Automatizados com Selenium**

## **📋 Contexto e Objetivo**

Nesta atividade, vocês irão exercitar habilidades fundamentais de automação de testes com Selenium em Python. O objetivo é criar scripts que naveguem em diferentes sites‑alvo, executem fluxos de teste definidos (login, carregamento dinâmico, compra e preenchimento de formulário) e gerem um relatório documentando cada passo, resultados esperados e obtidos, e eventuais erros ou exceções.

## **🔧 Pré‑requisitos**

Antes de começar, certifiquem‑se de:

* Ter Python 3.7+ instalado.
* Ter o pacote selenium instalado (pip install selenium).
* Baixar o WebDriver compatível com o navegador escolhido (ChromeDriver, GeckoDriver etc.) e adicioná‑lo ao PATH.
* Entender o básico de localização de elementos em HTML (ID, name, CSS selectors, XPath).

## **🗂 Estrutura Geral da Entrega**

1. **Código-fonte**: um script Python por site, nomeado de forma consistente (por exemplo, tc\_saucedemo.py, tc\_dynamic\_loading.py etc.).
2. **Documento de Relatório** (PDF ou Markdown): para cada teste, devem constar:  
   * **Caso de Teste**: título e ID (por exemplo, TC-001).
   * **Objetivo**: o que o script deve verificar.
   * **Passos Executados**: lista numerada das ações do Selenium.
   * **Dados de Entrada**: credenciais, valores de formulário etc.
   * **Resultado Esperado**: comportamento ou mensagem esperada.
   * **Resultado Obtido**: capturas de tela (opcional), prints de console ou logs.
   * **Status**: “Passou” ou “Falhou” e breve análise em caso de falha.

**Estrutura de Pastas**:  
  
 projeto-selenium/ # Nome do repositório

├─ drivers/ # WebDrivers / ChromeDriver / GeckoDriver

├─ scripts/ # Scripts usados para cada caso de teste

│ ├─ tc\_saucedemo.py

│ ├─ tc\_dynamic\_loading.py

│ ├─ tc\_demoblaze.py

│ └─ tc\_formy.py

└─ relatorios/ # Relatórios de cada caso de testes

├─ relatorio\_saucedemo.md

├─ relatorio\_dynamic\_loading.md

├─ relatorio\_demoblaze.md

└─ relatorio\_formy.md

## **🌐 Sites‑alvo e Casos de Teste**

### **1.** [**https://www.saucedemo.com/**](https://www.saucedemo.com/)

**Objetivos:**

* **TC-001**: Login bem‑sucedido com usuário padrão (standard\_user/secret\_sauce).
* **TC-002**: Login mal‑sucedido com usuário inválido (por exemplo, invalid\_user/wrong\_password).

**Pontos Importantes:**

* Verificar se, após login bem‑sucedido, a página de produtos é exibida.
* No caso de falha, capturar a mensagem de erro exibida no elemento com classe error-message-container.
* Registrar data/hora de início e fim do caso de teste.

### **2.** [**https://the-internet.herokuapp.com/dynamic\_loading**](https://the-internet.herokuapp.com/dynamic_loading)

**Objetivo:**

* **TC-003**: Ao clicar no botão **Start**, aguardar até o texto **“Hello World!”** estar visível e validar sua presença.

**Pontos Importantes:**

* Utilizar *explicit waits* (por exemplo, WebDriverWait + expected\_conditions) para aguardar o carregamento.
* Registrar o tempo total de espera.
* Capturar exceção caso o texto não apareça no timeout definido.

### **3.** [**https://www.demoblaze.com/**](https://www.demoblaze.com/)

**Objetivo:**

* **TC-004**: Simular uma compra de um produto qualquer.

**Passos Sugeridos:**

1. Escolher um produto clicando em um elemento da lista de filmes/softwares.
2. Clicar em **Add to cart** e aceitar alerta.
3. Ir para o carrinho e clicar em **Place Order**.
4. Preencher o formulário de cadastro (qualquer valor válido para nome, país, cidade, cartão, mês e ano).
5. Confirmar a compra e validar a mensagem de confirmação que aparece em um modal.

**Pontos Importantes:**

* Tratar alertas JavaScript.
* Capturar o texto completo do modal de confirmação (ID do pedido e valor).
* Salvar esses dados no relatório.

### **4.** [**https://formy-project.herokuapp.com/form**](https://formy-project.herokuapp.com/form)

**Objetivo:**

* **TC-005**: Preencher **todos** os campos do formulário e submeter.

**Campos a Preencher:**

* Nome, Sobrenome, Endereço de e‑mail, Telefone.
* Gênero (radio), Experiência (checkbox), Profissão, Ferramentas, Continente (dropdown), Comandos de Automação (multi-select).
* Clicar em **Submit** e validar o alerta de sucesso exibido.

**Pontos Importantes:**

* Explorar seleção por name, id, css selector ou xpath.
* Em dropdowns, usar Select do Selenium.
* Registrar valor final selecionado em cada campo no relatório.

## **📝 Instruções Gerais**

1. **Não copiem** o código uns dos outros; explorem diferentes estratégias de localização de elementos.
2. Configurem seu *WebDriver* apenas uma vez no início do script e finalizem com driver.quit().
3. Tratamento de erros: capturem exceções (*try/except*) e incluam no relatório qualquer falha inesperada.
4. Sigam a convenção de nomenclatura de arquivos e variáveis em Python (snake\_case).
5. Organizar commits no Git (mensagens claras).

## **✅ Critérios de Avaliação**

* **Correta identificação** dos elementos e uso de waits adequados.
* **Clareza e organização** do código, com comentários explicativos.
* **Exaustividade** dos relatórios: presença de todos os campos especificados.
* **Tratamento de exceções** e registro de falhas.
* **Estrutura de pastas** e convenção de nomes respeitadas.

**Entrega:** Enviar o link Git no classroom como entrega para essa atividade.

Boa prática de testes, bom uso de Selenium e boa sorte!