INSTITUTO FEDERAL DO PIAUÍ - IFPI CAMPUS CORRENTE
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ENGENHARIA DE SOFTWARE 3
PROFESSOR: IGOR BEZERRA REIS
TURMA: MÓDULO IV - TADS

INSTITUTO FEDERAL
Piauí
Campus Corrente

Aluno:_____ **Nota:**_____

Trabalho Final

Você deverá realizar testes de software automatizados utilizando o Selenium (com Python ou Java). Neste trabalho será avaliado a sua capacidade de:

- Identificar e documentar casos de teste;
- Criar scripts de automação de testes;
- Utilizar de boas práticas de codificação (código comentado, organização do projeto);
- Elaborar uma documentação clara e objetiva dos resultados.

FERRAMENTAS E TECNOLOGIAS

- **Linguagem de Programação:** Python ou Java.
- **Biblioteca de Teste:** Selenium WebDriver.
    - Para quem for fazer em Python, utilizar a biblioteca *pytest* (*https://docs.pytest.org/en/stable/*) ou *unittest* (*https://docs.python.org/3/library/unittest.html*)

    - Para quem for fazer em Java, utilizar a biblioteca *JUnit* (*https://junit.org/junit5/docs/current/user-guide/*)

- **Ambiente de Teste**:
    - Site The Internet - https://the-internet.herokuapp.com/
    - Site Ultimate QA - https://ultimateqa.com/automation/
    - Outras páginas que podem ser utilizada para teste:
        - https://www.globalsqa.com/samplepagetest/
        - https://react-shopping-cart-67954.firebaseapp.com/
        - https://katalon-demo-cura.herokuapp.com/profile.php#login
        - https://www.saucedemo.com/

.
.
.
.
.
.
.
.

ESCOPO DE TESTES

1. Cada aluno deverá criar **pelo menos 6 casos de teste diferentes**, dentre esses, os seguintes testes serão obrigatórios:
    - Testes de **autenticação** (login/logout).
        i. Simular o caso onde a autenticação ocorre com sucesso
        ii. Simular o caso onde a autenticação falha
        iii. Sugestão que pode ser utilizado: https://the-internet.herokuapp.com/login
    - Testes de **formulários** (preenchimento e validação de campos).
        i. Simular teste com envio corretos dos dados;
        ii. Simular caso com teste com envio de dados ausentes;
        iii. Sugestão que pode ser utilizado: https://ultimateqa.com/filling-out-forms/
    - Testes de **alertas** e pop-ups (confirmação, prompt).
        i. Simular caso com confirmação de alerta;
        ii. Simular caso com captura correta do prompt;
        iii. Sugestão que pode ser utilizado: https://the-internet.herokuapp.com/javascript_alerts

    - Utilize sua criatividade para desenvolver os demais testes, mantendo a mesma lógica dos anteriores e considerando o contexto de cada avaliação. Explore as diversas possibilidades de teste para cada elemento.

2. **Critérios de Aceite**:
    - Cada teste deve **fazer sentido funcional** (ex.: validar comportamento esperado, fluxos, mensagens de erro).
    - Todos os testes devem **rodar sem intervenção manual** (exceto em casos de pop-up que exijam ação do usuário, mas mesmo assim, tente automatizar ao máximo).
    - **Código bem comentado**, explicando a lógica e os passos mais relevantes.
    - **No ínicio do código**, em formato de comentário, deverá ter o nome do documento do caso de teste relacionado. (ex: DOCUMENTO DE CASO DE TESTE: `CT-001.pdf)`

3. **Estrutura Recomendada**:
    - **Setup**: Importação das bibliotecas, inicialização do driver e do ambiente de teste.
    - **Execução**: Passos do teste (navegar, clicar, preencher campos, submeter formulários, etc.).
    - **Validação**: Comparar resultado obtido com resultado esperado (mensagens, redirecionamento de página, etc.).
    - **Teardown**: Encerrar sessão, fechar navegador..

.
.
.
.

REQUISITOS PARA OS DOCUMENTOS DE CASO DE TESTE

Cada caso de teste deve ter um **documento** próprio, **inspirado no modelo** em anexo junto com este documento, devendo ter a seguinte estrutura base:

1. **Identificador do Caso de Teste**: Um código único, ex.: `CT-001`.
2. **Descrição**: Objetivo do teste em uma ou duas frases.
3. **Pré-condições**: Estado inicial ou requisitos necessários (ex.: usuário logado, página X aberta).
4. **Entradas**: Dados de entrada, se houver (ex.: valores de formulário).
5. **Procedimentos**: Passo a passo detalhado do que o teste faz.
6. **Saída Esperada**: Qual o resultado esperado após a execução (ex.: mensagem de sucesso, redirecionamento).
7. **Pós-condições**: Estado do sistema após o teste (ex.: usuário deslogado, item criado no banco).
8. **Observações**: Informações adicionais, potenciais riscos, links de referência, etc.
9. **Artefato:** Nome do arquivo de código gerado (ex: teste-ct-001.py)

ORGANIZAÇÃO DOS ARQUIVOS E ENTREGA

- **Código-Fonte**:
  - Organização em pastas:
    - `/src` ou `/tests` (onde ficam os arquivos de teste).
    - `/docs` (onde ficam os documentos dos casos de teste, em formato .PDF ou .DOCX).
  - Os códigos de teste devem ser **bem nomeados**, ex.: `CT-001-login.py` ou `CT-001-login.java`.
  - **Comentários**: Inserir comentários relevantes explicando o que está sendo validado em cada etapa.
- **Documentos de Caso de Teste**:
  - Cada caso de teste deve ter um documento conforme o modelo, nomeado, por exemplo, `CT-001-login.docx` ou `.pdf`.
- **Forma de Entrega**:
  - **Repositório Git** (ex.: GitHub, GitLab, Bitbucket).
    - Devem enviar o link do repositório onde todo o material está disponível na atividade criada no GSA
  - **Arquivo .ZIP ou .RAR:**
    - Anexar o arquivo compactado no GSA. O arquivo deve ser nomeado como: `Trabalho-Final-Eng3-[nome_do_aluno(a)].rar`
  - **DATA LIMITE DE ENTREGA**: **25/02/2025..**

.
.
.
.

OBSERVAÇÕES FINAIS

1. **Sejam criativos** nos testes, explorem os diversos cenários possíveis para cada teste;
2. **Erros ou dificuldades**: Registre no relatório ou no próprio código (em comentários) quais foram as principais barreiras e como foram solucionadas.
3. **Boas práticas**: Mantenha o código limpo e organizado, e documente qualquer biblioteca extra usada.
4. **Critérios de Avaliação**:
    - Cobertura e variedade dos testes. (No mínimo 6 testes)
    - Qualidade e clareza da documentação.
    - Organização e clareza do código (incluindo comentários).
    - Correção e completude da automação (scripts funcionais).

Boa sorte! Peguei leve no trabalho pois não quero reprovar ninguém nessa disciplina!

Dúvidas? Por favor, não tenham! Mas se tiver, envie uma mensagem!