GTI

SPRINT 2 – MISSÃO 6

PROJETO: "DEPLOYMENT QUALITY ASSURANCE"

ESTUDO DE CASO

Uma certa empresa decidiu estabelecer uma cultura QUALITY ASSURANCE em seu modelo de negócio, visando impactar positivamente processos de qualidade em suas as suas áreas de operação e tecnologia.

ESCOPO DO PROJETO

O projeto será composto por 3 Sprints que se complementam, onde os alunos deverão construir ações que validem a empresa a possuir uma cultura orientada a Q.A.

Em **duplas** os alunos desenvolverão projeto 3 em Sprints:

- SPRINT 1: Vale 0,5 ponto na AC-1 e presenças nas aulas
- **SPRINT 2: Vale 1 ponto na AC-2 e presenças nas aulas**
- SPRINT 3: Vale 1 ponto na AC-3 e presenças nas aulas

OBJETIVO

Aprender as nuances e aplicabilidade do *QUALITY ASSURANCE* em uma organização. Construir um projeto de implementação de Gerenciamento de Qualidade Total e realizar atividades que valem nota.

SPRINT 2 (1 ponto)

Início: **18/09** – Término: **09/10**. Vale 1,0 ponto na AC-2 e presenças nas aulas.
Composto por 4 missões que se complementam para a entrega total do projeto:

- Missão 5: Automação de Testes I– Vale 25% da AC-2
- **Missão 6: Automação de Testes II – Vale 25% da AC-2**
- Missão 7: Testes de API /QA em Mobile – Vale 25% da AC-2
- Missão 8: Validações e entrega final – Vale 25% da AC-2

MISSÃO 7

VALE 25% DA NOTA AC-2

ETAPA 1 – Processo de CI/CD e GitHub Actions

Definição:
CI/CD é uma prática de desenvolvimento de software que usa automação para integrar, testar e entregar código com frequência e confiabilidade. CI/CD é a sigla para Integração Contínua (Continuous Integration) e Entrega Contínua (Continuous Delivery).

Objetivo:
A CI/CD é uma abordagem ágil que combina a parte operacional com o desenvolvimento de um projeto. Ela tem como objetivo acelerar o processo de entrega de atualizações e novas funcionalidades aos usuários.

Pipeline:
A CI/CD pode ser representada como um pipeline, onde o novo código é enviado, testado e publicado como código pronto para produção.

Implementação:
Para implementar a CI/CD com sucesso, é recomendado começar com uma estratégia clara, definindo os recursos necessários e automatizando o máximo possível.

Frameworks populares para CI/CD:

- **Jenkins:** Uma das ferramentas mais conhecidas e amplamente utilizadas para CI/CD. É open source e altamente extensível com uma vasta gama de plugins.
- **Azure DevOps**: Oferece uma suite completa de ferramentas para CI/CD, incluindo Azure Pipelines.
- **GitLab CI/CD**: Integrado ao GitLab, oferece pipelines de CI/CD robustos e fáceis de configurar.
- **GitHub Actions**: Ferramenta de automação oferecida pelo GitHub, a popular plataforma de hospedagem de repositórios Git.

O que é o GitHub Actions?
É uma feature que permite automatizar tarefas no ciclo de vida de desenvolvimento diretamente no GitHub. Ele facilita a criação de pipelines de CI/CD, desde a execução de testes automáticos até o deploy de uma aplicação.

Como Funciona o GitHub Actions
- **Workflow**: Uma automação composta por ações. Um repositório pode ter vários workflows.
- **Job:** Um conjunto de etapas que são executadas em um ambiente virtual.
- **Action:** A menor unidade do workflow, uma tarefa que faz parte de um job, como rodar testes ou realizar deploy.
- **Trigger:** Evento que inicia o workflow, como um commit ou pull request.

ETAPA 2 – Projeto Python com CI/CD Usando GitHub Actions

Vamos criar um projeto básico em Python, com CI/CD configurado para executar testes e realizar o versionamento.

1. Inicializar um Repositório no GitHub: abra seu repositório QA no GitHub.

2. Abra o VsCode e abra um novo terminal:

3. Clone o repositório no seu computador:
   git clone https://github.com/seu-usuario/nome-do-repositorio.git

4. Abra o repositório: cd nome-do-repositorio

5. Criar um Projeto Python Simples. No diretório do projeto, crie um arquivo **main.py** com o seguinte código simples:

```
def soma(a, b):
    return a + b
```

6. Crie um arquivo de testes test_main.py:

```
import pytest
from main import soma
def test_soma():
    assert soma(2, 3) == 5
```

7. Criar Arquivo de Requisitos: no terminal, crie um arquivo requirements.txt:

```
pytest
```

8. Configurar o GitHub Actions: dentro da pasta do projeto, crie uma pasta: .github/workflows

9. Dentro dela, crie um arquivo python-ci.yml para configurar o GitHub Actions:

```
name: Python CI

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build:
```

```
    runs-on: ubuntu-latest

    steps:
    - name: Check out the code
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: '3.x'

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Run tests
      run: |
        pytest
```

Esse workflow é acionado a cada push na branch main. Ele faz checkout do código, instala as dependências e executa os testes unitários.

10. Volte à pasta do repositório e Commit e Push no Repositório: adicione os arquivos ao git e faça commit:

    git add .
    git commit -m "Configuração inicial do projeto Python com testes"
    git push origin main

11. Quando você fizer o push, o GitHub Actions será acionado e rodará o pipeline automaticamente.

ETAPA 3 – Simulação Completa de CI/CD com GitHub Actions

Vamos simular o processo completo de CI/CD:

12. Realizar um Push com Testes Unitários
Sempre que você fizer o push ou abrir um pull request no repositório, o GitHub Actions rodará o pipeline, incluindo a execução dos testes.

13. Verifique o status do pipeline na aba "Actions" do seu repositório no GitHub. Se os testes passarem, você verá a mensagem de sucesso.

14. Ajustar o Código e Atualizar a Versão: vamos simular uma mudança no código e um update de versão:
    def soma(a, b):

return a + b + 1 # Modificação para quebrar o teste

15. Atualize a versão do código no arquivo main.py e no README.md. Teste a falha do pipeline ao quebrar um teste:

Faça o commit e o push:

git add .
git commit -m "Alterar função soma e atualizar versão"
git push origin main

16. O pipeline falhará, mostrando que o teste não passou.

17. 3. Corrigir o Código e Fazer um Novo Push: Corrija o código para passar no teste:

def soma(a, b):
return a + b # Correção do código

18. Faça o commit e o push novamente:

git add .
git commit -m "Corrigir soma e passar testes"
git push origin main

Agora, o pipeline será bem-sucedido e os testes passarão.

TAREFA 3 – ROBÔ TESTE – SELENIUM IDE

1. Abra o ícone do **Selenium IDE** na barra de ferramentas do navegador. Clique no ícone para abrir a ferramenta.

2. Clique em "Create a new project";

3. Escolha um nome para o projeto (exemplo: "Teste Ecommerce") e clique em **OK**.

4. Clique no botão "Record" (Gravar um novo teste).

5. Escolha um nome para o seu teste (exemplo: "Teste loja");

6. Agora você irá usar o site de simulação de eCommerce SWAG LABS. Acesse pelo Chrome o endereço abaixo:

Swag Labs (saucedemo.com)

7. Conheça um pouco a DEMO antes de iniciar o próximo teste. Você precisará fazer um teste completo de usabilidade do sistema, realizando o login, escolhendo os produtos,

indo para o carrinho confirmar a compra, preencher dados e forma de pagamento e sair do sistema.

8. Realize um novo processo de gravação com o Selenium IDE. Inicie um novo teste com uma nova gravação

9. Insira o endereço citado acima;

10. Faça o Login e Senha: Login: **standard_user** – Pswd: **secret_sauce**.

11. Adicione todos os itens no carrinho;

12. Vá para o carrinho e remova dois 4 itens;

13. Faça checkout;

14. Insira nome e sobrenome e caixa portal (CEP) fictício e clique em Continue;

15. Confira os valores e carrinho e clique em finish;

16. Ao completar o Checkout, clique em Back Home;

17. Na home, clique na lista de item e faça logout;

18. Feche a janela do browser;

19. Quando terminar de gravar as interações, volte à janela do Selenium IDE e clique no botão **Stop recording** (Parar gravação).

20. Executar o Teste Gravado: Após parar a gravação, você verá uma lista de comandos gravados no Selenium IDE.

21. Para executar o teste gravado, clique no botão **Run current test** (Executar teste atual). O Selenium IDE abrirá uma nova janela e executará automaticamente o que você gravou.

TAREFA 5 – FINALIZAÇÃO ETAPA

22. Após gravar, clique no menu de três pontos no canto superior direito da interface do Selenium IDE.

23. Selecione **Save Project** (Salvar Projeto) e escolha uma pasta no seu computador para salvar o projeto e os testes.

24. Crie o cartão Missão 6 e salve os códigos de hoje;

25. Coloque no fim o nome e RA dos alunos presentes na atividade no cartão de hoje;

26. Coloque o cartão MISSÃO 6 na lista EM VALIDAÇÃO.

SUCESSO A TODOS!