ADS /GTI

SPRINT 2 – MISSÃO 5

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 5

VALE 25% DA NOTA AC-2

TAREFA 1 – DEFINIÇÕES, OBJETIVOS e PREPARAÇÃO:

Automação de testes: é o uso de ferramentas para executar scripts de teste automaticamente, em vez de fazê-los manualmente.

Benefícios da Automação:

- Eficiência e velocidade na execução de testes repetitivos.
- Redução de erros humanos.
- Possibilidade de execução de testes em grande escala (regressão).
- Liberação de tempo para que QA foque em testes mais complexos (testes exploratórios).

Limitações da Automação:

- Não é viável para todos os tipos de teste (ex.: testes de usabilidade).
- Custo inicial elevado em termos de configuração e manutenção.
- Manutenção de scripts pode ser cara se a aplicação mudar muito.

A missão geral da AC-2 envolve a execução de testes automatizados.

1. Baixe o arquivo esse "**Missão5-Projeto QA ADS-5.pdf**" disponível no AVA;

2. Abra o GitHub oficial da dupla/trio e em seguida abra o repositório que estão usando para o projeto;

3. Suba no seu repositório o arquivo "**Missão5-Projeto QA ADS-5.pdf**" para constar no portfólio;

4. Agora abra o projeto deste repositório e visualize o quadro Kanban que está gerenciando o projeto da AC-1;

5. Jogue os cartões de 1 a 4 em FINALIZADOS;

6. Criar e colocar o cartão MISSÃO 5 para a lista EM ANDAMENTO;

TAREFA 2 - EXECUÇÃO:

Objetivo: Configurar o Selenium e integrar com GitHub para versionamento de código.

7. Instale o Selenium:
   - Abra o terminal no VsCode e digite:

     pip install selenium

8. Configure o WebDriver

Baixe o WebDriver para o seu navegador:

- Para **Chrome**, busque no Google a versão atual do ChromeDriver e baixe a versão correspondente ao seu navegador (consulte a versão atual do Google. Atualize se necessário).

Adicione o WebDriver ao PATH do Windows:

- No Windows, vá até **Variáveis de Ambiente** e adicione o caminho do WebDriver à variável PATH (não apague os caminhos já existentes, apenas adicione o abaixo);

Clique em Novo e adicione o caminho completo onde o WebDriver está localizado (por exemplo, C:\Users\SeuUsuario\Downloads\chromedriver_win64).

9. No VsCode, vá até a aba de extensões e busque por "Python";

10. Configure o repositório no GitHub: Acesse sua conta no GitHub;

11. Confirme se seu repositório está como PUBLIC;

12. Abra o terminal no VsCode e navegue até o diretório onde deseja armazenar o projeto:

    cd path/to/your/Project (apenas representação e você cria a pasta como quiser)

13. Clone o repositório GitHub no diretório local:

    git clone https://github.com/seu-usuario/seurepositorio

Isso criará uma pasta local conectada ao repositório GitHub.

> **Versionamento do Código**: Sempre que você fizer alterações no código, use os seguintes comandos para versionamento:
> git add .
> git commit -m "Sua mensagem de commit"
> git push origin main

Isso enviará suas alterações para o repositório GitHub.

TAREFA 3 - Criação de Script de Automação de Testes:

Objetivo: Desenvolver um script básico para automação de teste em uma aplicação web.

Passo a Passo Detalhado:

Passo 1: Criar o Script no VsCode

1. No **VsCode** e vá até o diretório onde você clonou o repositório GitHub.

2. Crie um arquivo chamado test_login.py para o seu primeiro teste.

Passo 2: Escrever o Script de Automação com Selenium

1. **Importação de Bibliotecas**:
   - No arquivo test_login.py, importe o Selenium e as bibliotecas necessárias:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.service import Service
import time
```

2. **Inicialização do WebDriver**:
   - Configure o Selenium para usar o WebDriver do navegador Chrome:

```
# Defina o caminho para o WebDriver (Substitua o caminho para o seu ChromeDriver)
service = service('C:/caminho/para/chromedriver.exe')

# Inicializar o WebDriver
driver = webdriver.Chrome(service=service)
```

3. **Navegar para a Página de Login**:
   - Utilize o WebDriver para abrir a página de login de exemplo:

```
driver.get("https://exemplo.com/login")
```

4. **Localizar Elementos e Realizar Ações**:
   - Use o Selenium para encontrar os campos de login e senha, preenchê-los e clicar no botão de login:

```
# Encontrar o campo de login e preencher
username = driver.find_element(By.NAME, "username")
username.send_keys("seu_usuario")

# Encontrar o campo de senha e preencher
password = driver.find_element(By.NAME, "password")
password.send_keys("sua_senha")

# Simular o clique no botão de login
login_button = driver.find_element(By.NAME, "login")
login_button.click()
```

5. **Esperar a Resposta da Página**:
   - Após o login, adicione um tempo de espera para verificar o resultado:

time.sleep(6) # Espera de 6 segundos para carregar a página

6. **Verificar o Resultado**:
   - Opcionalmente, você pode verificar se o login foi bem-sucedido com uma validação simples:

```
if "dashboard" in driver.current_url:
   print("Login realizado com sucesso!")
else:
   print("Falha no login.")
```

7. **Encerrar o WebDriver**:
   - Ao final, feche o navegador:

```
driver.quit()
```

Passo 3: Executar o Script

1. No terminal do VsCode, execute o script com o comando:

```
python test_login.py
```

2. Observe o comportamento no navegador onde o Selenium automatiza as interações.

Passo 4: Commit e Push do Código para o GitHub

1. Após criar o script, adicione e faça commit do código para o GitHub:

```
git add test_login.py
git commit -m "Adicionando script básico de login com Selenium"
git push origin main
```

<u>TAREFA 4 - FINALIZAÇÃO:</u>

14. Salve todos os códigos de hoje no MISSÃO 5;

15. Coloque no fim o nome e RA dos alunos presentes na atividade;

16. Coloque o cartão na lista EM VALIDAÇÃO.