CONHECIMENTO DE QUALIDADE DE SOFTWARE

E AUTOMAÇÃO/C#

1. Como você define qualidade de software?

Defino a qualidade de software como a medida em que um software atende aos requisitos e expectativas das partes interessadas, cumprido seu papel ao qual foi criado. Podemos usar indicadores para quantificar a qualidade de uma aplicação, como usabilidade, eficiência, confiabilidade, segurança, portabilidade, também podemos aplicar métricas para acompanhar o crescimento do desenvolvimento e encontrar possíveis impeditivos, como tempo de respostas gerais e taxa de usuários afetados por um bug. Além da definição acima, podemos definir com padrões e boas práticas de engenharia de software, como metodologias de desenvolvimento, testes e documentações adequadas ao projeto.

2. Qual o melhor momento para executar os testes UAT(User Acceptance Test) no projeto? Quem deveria executar esse teste? O que deve ser considerado como critério de aceite para essa fase do projeto?

O momento ideal para executar os testes UAT é após a conclusão dos testes de integração e antes do lançamento do software em produção. Os testes UAT devem ser realizados pelos usuários finais ou por representantes deles, como usuários-chave ou stakeholders. Os critérios de aceite para a fase de UAT devem ser definidos no início do projeto, em colaboração com as partes interessadas, e devem ser baseados nos requisitos de software e nas expectativas dos usuários em relação ao seu desempenho, funcionalidades, usabilidade, segurança, entre outros aspectos relevantes. Ao final dos testes, os resultados devem ser avaliados em relação aos critérios de aceite já definidos.

3. Que tipo de casos de teste poderia ser automatizados em um sistema que resultaria na redução do tempo do período de teste durante o desenvolvimento do projeto?

Casos de teste que envolvem a repetição de atividades ou que requerem a execução de um grande número de passos ou cenários são ideais para a automação de testes, pois podem reduzir significativamente o tempo de teste durante o desenvolvimento do projeto. Alguns exemplos de casos de teste que poderiam ser automatizados incluem: testes de regressão, testes de desempenho, testes de integração, testes de usabilidade e testes de segurança. Em resumo, utilizando-os separadamente ou em conjunto, esses testes podem envolver a interação de vários componentes do sistema, sendo que a criação da automação auxilia a garantir que todos os cenários possíveis sejam cobertos e executados corretamente sempre que houver uma atualização no ciclo de desenvolvimento.

4. Quando você abre um BUG, que tipo de informação deve ser incluída pelo QA para ajudar os desenvolvedores a entender o problema e ajudar na investigação do problema?

Quando se efetuado um report de bug, deve procurar especificar o máximo possível o problema encontrado, porém, alguns aspectos são essenciais para orientar os desenvolvedores acerca do bug, assim como: título, descrição, ambiente/versão, anexos/evidências, prioridade, etiqueta, passos para reproduzir o erro, contexto de uso e afins. Fornecendo estas informações auxilia o tempo de ajuste da aplicação diminuir drasticamente.

5. Liste algumas ferramentas utilizadas para desenvolvimento de automação REST que você conhece?

Tenho conhecimento sobre as ferramentas: Postman, Jmeter, SoapUI e REST-assured.

6. Se você teve alguma experiencia com a ferramenta acima mencionada, descreva:

Tenho experiência com JMeter para testes de carga/performance utilizando chamadas simultâneas em banco de dados, APIs REST e URLs. E com a ferramenta Postman, que oferece uma ampla gama de recursos que tem como objetivo de simplificar os testes de APIs RESTful. Os testes que já efetuei estão voltados para automação e análise de requisições HTTP com os métodos GET, POST, PUT e DELETE que garante que as solicitações funcionem corretamente em diferentes cenários e também ajuda a gerar documentação para as chamadas baseadas no body das mesmas. Ambas as ferramentas foram utilizadas em paralelo e com o Swagger, que facilita e organiza a documentação das chamadas da APIs internas e externas. As áreas que efetuei a implementação de ambas as ferramentas foram: Balanço financeiro, controle de dados internos de aplicações mobile e web sobre estoque de produtos, dados do cliente e sincronização de informação gerais.

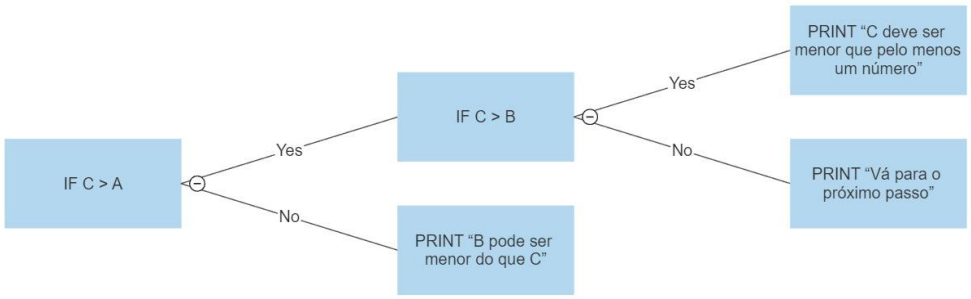
7. Você é o analista de teste e participa de uma sessão de estimativa de tempo para os novos requisitos que chegam para o projeto (Com toda a equipe do projeto). Além de considerar o tempo para o teste funcional, automação e de regressão, o que mais deve considerar em relação as necessidades do teste para dar sua estimativa?

Para a estimativa de tempo de desenvolvimento é importante levar em conta não só as etapas de teste funcional, automação e regressão, mas também, algumas etapas que podem reduzir o tempo desperdiçado e melhorar a eficiência nos testes, sendo elas: Análise dos requisitos: Serve para identificar os testes e casos de uso estão sendo executados corretamente, esses planejamentos devem ser executados em conjunto com os demais integrantes do projeto; Planejamento dos testes: É o momento de elaborar os planos de testes, levando em consideração os ambientes a serem preparados e testados, como por exemplo, sistemas operacionais diferentes ou diferentes tipos de navegadores e dependencias externas como linguagem de integração e serviços a serem usados; Dados de teste: É importante fazer um levantamento de dados/métricas realistas equivalentes a um ambiente de produção; Recurso de teste: O momento de efetuar uma análise dos recursos necessários para efetuar os testes a longo prazo, sendo eles, sistemas, servidores, serviços e tudo que for necessário para efetuar os testes e manter a qualidade e custos ao cliente. Essas etapas podem ser realizadas antes mesmo do desenvolvimento dos novos requisitos, de modo que se otimiza o tempo de preparação dos planos de teste e dos casos de teste. Implementação dos casos de teste: Etapa na qual o os testes são realizados na aplicação, sejam estes testes manuais ou automações; Fase de correção: Momento no qual o report dos bugs encontrados é feito e retornado para a equipe de desenvolvimento. É importante levar em conta nessa fase de testes adicionais podem ser necessários de acordo com a complexidade do requisito; Encerramento: Etapa na qual se discute os resultados com a equipe, a fim de otimizar o processo para as próximas iterações.

8. Para o seguinte código, quantos casos de teste são necessários para cobrir todas as condições (Verdadeiras e Falsas)?

```
READ A
READ B
READ C
        IF C > A THEN
        IF C > B THEN
                PRINT "C deve ser menor que pelo menos um número"
        ELSE
                PRINT "Vá para o próximo passo"
        ENDIF
        ELSE
                PRINT "B pode ser menor do que C"
END IF
```

Para cobrir todas as condições, são necessários 3 casos (C<=A; C>A e C>B; C>A e C<= B). Assim, temos as possibilidades ilustradas na árvore de decisão a seguir:



9. Baseado na questão acima descreva quais os testes que você pensou a respeito:

Para efetuar os testes no consigo acima, é possível aplicar inúmeros métodos de testes, como performance, segurança, validação entre outros. Mas os principais testes para verificar se a aplicação entrega o que foi proposto são os testes unitários e testes de funcionalidade.

Assim, alguns casos de teste que poderiam ser executados são:

- Validar se há tratamento de erro para entrada com Caracteres;
- Validar se o comportamento é mantido para números muito grandes (tanto negativos quanto positivos);

- Validar se o comportamento é mantido para números muito pequenos (tanto negativos quanto positivos);
- Validar o comportamento para entradas iguais.

É possível notar ainda, analisando o código que há inconsistências matemáticas nas saídas, isso porque, ao entrar com o maior número na variável C, a saída é “C deve ser menor que pelo menos um número”. O mesmo ocorre ao entrar com A>B>C, cuja saída é “B pode ser menor do que C”.

10. Você vê algum problema no código abaixo? Se sim descreve qual.

```
try
{
    var quantityBuyers = 0;
    var totalPrice = 1475.89;

    var priceForBuyer = totalPrice / quantityBuyers;

    return priceForBuyer;

}
catch (Exception ex)
{
    throw ex;
}
```

O Código acima possui um erro estrutural comum na utilização do throw.
Quando uma exceção é lançada com a variável Exception ela substitui a localização atual pela localização onde foi capturada no bloco Exception.
A maneira correta de lançar seria com

catch (Exception ex)
{
throw;
}

Além disso, o código possui uma divisão por 0. Embora isso não represente propriamente um erro em C#, é bastante provável que no contexto em que está sendo aplicado (que envolve preços e compradores) essa divisão não faça sentido. Assim, o mais recomendado seria ter um tratamento para que essa divisão não ocorra.

11. Como faço para obter o nome de todas as pessoas nascidas a partir de 1980 da coleção abaixo? (escreva o código c#)

```
var people = new[] { new { Name = "José", DateBirth = new DateTime(1982, 03, 27), Active = true },
                     new { Name = "Leandro", DateBirth = new DateTime(1978,04,03), Active = false},
                     new { Name = "Pedro", DateBirth = new DateTime(1980,05,24), Active = true}};
```

Podemos fazer esta busca de diversas formas, porém, a mais eficiente seria utilizando a estrutura de consulta LINQ, aplicação abaixo:
Para isso, no início do código deve-se chamar a biblioteca nativa Linq com
using System.Linq

E então, para a consulta utilizar o código abaixo:

```
var nascidas_apartir1980 = (
   from person in people
   where person.DateBirth.Year >= 1980
   select person.Name
).ToList();
```

E então, exibir no console com

```
nascidas_apartir1980.ForEach(Console.WriteLine);
```

CONHECIMENTO DE QUALIDADE DE SOFTWARE

E AUTOMAÇÃO/C#

TESTE DE AUTOMAÇÃO:

Avaliação – Busca CEP

Utilizando uma linguagem de automação, preferência, C#(Specflow e NUnit), crie um script que realize os seguintes passos:

1. Entre no site dos correios;
2. Procure pelo CEP 80700000;
3. Confirmar que o CEP não existe;
4. Voltar a tela inicial;
5. Procure pelo CEP 01013-001
6. Confirmar que o resultado seja em “Rua Quinze de Novembro, São Paulo/SP”
7. Voltar a tela inicial;
8. Procurar no rastreamento de código o número “SS987654321BR”
9. Confirmar que o código não está correto;
10. Fechar o browser;

Detalhes do script:

- Um check por id;
- Um check por xpath;
- Um check por css;

Desejável:

Otimizar o código para rodar no menor tempo possível

Enviar um screenshot do resultado do teste rodado no Visual Studio.

O código implementado deverá ser disponibilizado no GITHUB ou outro repositório público para avaliação.
Tendo em vista que o código c# é executado diariamente por um servidor de automação, será necessário efetuar o versionamento do código criado.

a) Quais os comandos GIT para submeter o novo código para o repositório de versionamento?

Primeiro utilizamos o comando "git add" que serve para adicionar os arquivos no Stage area, em seguida utilizamos o comando "git commit" que salva as alterações do Stage no repositório local, e por último utilizamos o comando "git push" para enviar os commits do repositório local ao repositório remoto.
observação: sempre podemos utilizar o “git status” para saber o que foi adicionado no repositório local.
Exemplo de aplicação abaixo:
$git add .
$git status
$git commit -m "Adicionando funcionalidade X"
$git push