PagoNxt | GetNet

TI QUALIDADE

Responda as questões a seguir:

1) O cliente adquiriu um novo modelo de máquina de cartão de crédito e ela aceita as bandeiras Visa, Master, Elo, Amex e Hiper e executa duas operações para cada bandeira débito e crédito. Quantos testes serão necessários para validação dessa máquina e qual técnica a ser utilizada?

R: Número de bandeiras (Visa, Master, Elo, Amex e Hiper) = 5 x 2 operações para cada bandeira (Débito e Crédito) = 10 testes para validação da máquina de cartão.

Técnica = Combinação - calcula o número de agrupamentos (bandeiras) de elementos diferentes (operações).

2) A bandeira Visa disponibilizou um novo range de bins no mercado que possui seis dígitos 232425 a 232460. Quais os testes deverão ser feitos desse range de bins para confirmar que está funcionando?

a) 230000,232425,232460

b) 0,232425,232460,240000

c) 232425,232460

d) 232424,232425,232459,232460,232461

R: Letra a) 230000,232425,232460 é o correto. 230000 é o valor abaixo do intervalo de 232425 a 232460. Também temos o primeiro número dentro do intervalo (232425) e o último número dentro do intervalo (232460). É possível validar se o teste rejeita o número fora do intervalo (230000) e aceita como esperado os números dentro do intervalo (232425 a 232460).

3) Cite uma abordagem desenvolvimento ágil e explique-a, responda em suas palavras.

R: Uma abordagem de desenvolvimento ágil mais conhecida é o Scrum. O trabalho dentro do Scrum é dividido em Sprints, que geralmente dura de 2 a 3 semanas. Durante esse período, a equipe trabalha para que seja completado depois de definido suas features e estórias de usuários. Em cada time de Scrum, existem os papéis hierárquicos

bem definidos, como o Stakeholder, que é a pessoa mais próxima ao cliente, o Product Owner, que junto ao Stakeholder define os interesses do cliente e suas prioridades do backlog do produto no board, existe também o Scrum Master, que lidera a equipe e ajuda a aderir aos princípios e práticas da metodologia ágil, e também os times de desenvolvimento e o time de QA (Quality Assurance), que é responsável por desenvolver o produto e testar suas funcionalidades até que isso seja entregue, e sem erros. No Scrum também existem as cerimonias ou ritos, a serem seguidos, a reunião de Planejamento, que acontece no primeiro dia de cada Sprint, definindo quais são as features e funcionalidades a serem trabalhadas; as dailys, para acompanhar o progresso do time diariamente e ver quais as dificuldades encontradas e fazer o ajustamento de plano conforme necessário; a reunião de revisão, onde se obtém feedback do PO e Stakeholder mostrando o trabalho feito, e a reunião de Retrospectiva e Refinamento, onde a equipe analisa o processo e o backlog e ajuda a identificar maneiras de melhorar nas próximas sprints. O Scrum ajuda a colaborar, flexibilizar e a entregar melhor ao cliente final. Quando dividido o trabalho e recebendo feedbacks regularmente, é notável a mudança e melhoria nos requisitos do cliente, garantindo que o produto atenda a todas as necessidades do usuário final.

4) O que é CI/CD?

R: Continuos Delivery e Continuous Integration é uma prática de desenvolvimento de software que combina sua integração em ambas as partes.

CI - Prática que os desenvolvedores integram regularmente o código que estão escrevendo em um repositório compartilhado. Toda vez que ele é integrado, os testes automatizados são executados, verificando se as alterações deixaram passar algum erro. Ele detecta problemas de integração o quanto antes, permitindo que os erros sejam corrigidos rapidamente e evitando maiores problemas no futuro. Exemplo de ferramenta CI: Jenkins.

CD - Prática que visa automatizar o processo de implantação de software em ambientes de produção ou próximos, de forma confiável e consistente. Qualquer mudança de código que passe com sucesso por testes de integração é automaticamente disponibilizada para implantação. Com isso, as equipes de desenvolvimento são capazes de entregar as novas funcionalidades ou correções de bugs rapidamente e sem riscos. Exemplo de ferramenta CD: Git.

5) O que é TDD, BDD e ATDD e quando são aplicadas?

R: TDD – Test Driven Development - é mais focado em testes unitários, que são escritos antes do código de produção. Ele tem 3 fases: escrever um teste falho, escrever o código

mínimo necessário para passar no teste e fazer o refatoramento de código para melhorar sem alterar o comportamento de saída dele. O TDD garante que código seja testado desde o início e que cada funcionalidade tenha testes automatizados integrados.

BDD – Behavior Driven Development - é mais focado no comportamento do sistema e na comunicação entre as equipes. Ele é baseado em GIVEN, WHEN e THEN e descreve o comportamento esperado do software em termos de cenários dos usuários. O Gherkin, por exemplo, uma linguagem natural, é usado para descrever o comportamento do sistema em termos de entradas e saídas. É aplicado para garantir que o software atenda aos requisitos do cliente e que o comportamento esperado seja definido e testado.

ATDD – Acceptance Test-Driven Development – amplica o TDD envolvendo todas as partes interessadas na definição de testes, incluindo desenvolvedores, testadores e clientes. Juntos, são criados uma série de testes de aceitação que representam os critérios de aceitação do cliente. Eles são escritos antes do código de produção e são especificados executando o comportamento esperado do sistema. Ele é aplicado para garantir que o software atenda aos requisitos de cliente desde o início do desenvolvimento.

6) Cite uma heurística de testes utilizada para testes de front e de back e explique-as.

R: Heurística de particionamento de equivalência - divide o conjunto de possíveis entradas em classes de equivalência, onde cada classe representa um conjunto de entradas que devem ser tratadas de forma semelhante pelo sistema. É aplicado a alguns tipos de testes como Unidade, Integração e Sistema. Existem alguns passos a serem seguidos:

Identificação de Classes de Equivalência - identifica as diferentes categorias ou classes de entradas que o sistema pode receber;

Seleção de Casos de Testes – seleciona um ou mais casos de testes para cada classe de equivalência. Também garante que os casos de testes escolhidos representam a classe de equivalência;

Execução dos Testes – executa os casos de testes selecionados e verifica se o sistema se comporta como o esperado para cada classe de equivalência. Ele verifica se as entradas são aceitas corretamente e se as entradas inválidas são rejeitadas ou tratadas adequadamente;

Refinamento de Testes – refina e ajusta os casos de testes como esperado para garantir uma cobertura adequada das classes de equivalência.

7) Cite ferramentas de testes automatizados para Desktop, Web, API ´s e mobile.

R: Desktop – Selenium Webdriver – ele testa aplicativos desktop usando frameworks como Appium (multiplataforma) ou WinAppDriver (aplicativos Windows).

Web – Selenium Webdriver – a mais comum utilizada para automação de testes em aplicativos web; Cypress – tem como usabilidade uma experiência de desenvolvedor ágil e amigável.

APIs – Postman – cria, envia e automatiza solicitaçõs HTTP; REST Assured – biblioteca Java para testar chamadas, facilidando a automação de testes de integração e aceitação; SoapUI – cria, executa e automatiza testes de API.

Mobile – Appium – ferramenta de automação de testes multiplataforma para aplicativos móveis nativos, híbridos e baseados em web para iOS/Android; Calabash – escreve e automatiza testes de aceitação de aplicativos móveis nativos e híbridos em iOS/Android.

Prática de API

Desenvolva o script da automação seguindo as informações a seguir:

Documentação = https://reqres.in/

URI = https://reqres.in/api/

1) Validar o script de "CREATE" método "POST" cobertura de testes em RestAssured da API

2) Validar cobertura de Status Code, Campos obrigatórios e Contrato

3) Desenvolver com POJOs.

OBS1: Enviar o link do código no GitHub

https://github.com/renatoalfonsettisilva/getnet_challenge_Renato