Perguntas QA

Tags	Engineering

O que é um QA Engineer?

Um QA Engineer dá suporte ao desenvolvimento de software e equipas, no que toca a criação, testagem, implementação e resolução de problemas de uma aplicação/serviço/produto. Isto significa que estão envolvidos no desenvolvimento do principio até ao fim.

Processo de Testes

Dependendo da area de foco, a fase de testes pode ser grande ou pequena no scope do projeto. Mas o objetivo essencial é o mesmo: testar a funcionalidade, procurar bugs e prevenir que estes bugs afetem outras partes do projeto.

As atividades de testes variam consoante o tipo de testes de software, alguns exemplos incluem:

- Black box testing
- White box testings
- Integration testing
- User acceptance testing (UAT)
- Automation testing
- Negative testing
- Performance testing

Os test steps de cada bug, permitem ao QA Engineer encontrar problemas que sejam prioritários e outros que sejam menos relevantes. Os dados extraídos em cada teste vão permitir tirar alguns insights que podem ser usados para tratar bugs pela equipa de desenvolvimento.

Quais as diferenças entre Quality Assurance e Quality Control?

QA vs QC são diferentes relativamente ao seu intuito. O Quality Assurance centra-se em como manter a qualidade e prevenir issues, onde o Quality Control está focado em identificar problemas do produto ou serviço.

Como exemplo, a principal responsabilidade de um tester iria focar-se no controlo de qualidade e na testagem de software para descobrir e reportar defeitos da aplicação. Ao invés um quality assurance engineer faz parte de todo o processo de desenvolvimento e pode usar uma postura proativa na prevenção de erros ao longo do ciclo de desenvolvimento.

Quais são alguns Exemplos de Ferramentas de Quality Assurance?

Algumas ferramentas de QA são tão simples como uma Requirement Traceability Matrix. Estes documentos de requisitos garantem que os mesmos são associados e testados ao longo do processo de verificação. Outros exemplos incluem software de gestão de testes ou programas que facilitam a testagem de QA como os softwares de automação de testes.

A automação tem uma grande papel na testagem de software em conjunto com a testagem manual. Os scrips de testes entram em jogo principalmente nos testes de regressão.

Perguntas Básicas de QA

Qual é a diferença entre quality assurance, quality control ?

Enquanto que o QA se foca em manter a qualidade e prevenir erros ou bugs o QC foca-se em identificar problemas no produto ou serviço. Ou seja o QA atua na prevenção e o QC atua na verificação e validação de erros.

Como é definido o Ciclo de Desenvolvimento de Software e como é que a testagem se insere no mesmo?

O ciclo de desenvolvimento de software é definido por 6 fases sendo que a testagem se insere ao longo de todo o ciclo:

1. Análise de Requisitos - Durante esta fase são definidos os requisitos com os stakeholders e identificados os processos que são testáveis.
    a. Critério de Entrada - Requisitos documentados, critérios de aceitação e arquitetura de projeto pretendida.
    b. Critério de Saída - Requirement Traceability Matrix aprovada e report de testes automatizáveis.
2. Desenvolvimento de Test Cases - Nesta fase os testes são criados. Cada teste tem os seus inputs, procedimentos, execução, condições e resultados esperados. Os test cases devem ser transparentes, eficientes e adaptáveis. Depois de todos os test cases estarem criados a cobertura do teste deve ser de 100%. Qualquer script de automação deve também ser criado durante esta fase.
    a. Critério de Entrada - Test Plan aprovado com Timelines e Análise de Custo/Risco.
    b. Critério de Saída - Test Cases e Scripts de Automação aprovados.
3. Setup de Ambiente de Testes - Durante esta fase, o ambiente de testes é configurado e lançado. Esta fase pode incluir várias ferramentas de testes, como o TestComplete, Selenium, Appium ou Katalon Studio. Algumas vezes, esta fase também inclui o setup dos servidores de testes, smoke tests também são realizados para que seja garantido que tudo está a funcionar como pretendido.
    a. Critério de Entrada - Definição da Arquitetura do Projeto e Sistema.

    b. Critério de Saída - Um sistema de testes totalmente funcional e test cases aprovados.

4. Execução de Testes - Durante esta fase as features são testadas e lançadas no ambiente de testes usando os test cases que foram definidos. Os resultados esperados são comparados com os obtidos a são agregados num report a ser entregue à equipa de desenvolvimento.
    a. Critério de Entrada - Todos os Critérios de Saída das fases anteriores.
    b. Critério de Saída - Todos os testes são executamos e os resultados são documentados.
5. Fecho do Ciclo de Testes - Nesta fase o report de testes é preparado. Este report deve resumir todo o processo de testagem e comparar entre os resultados esperados e os obtidos. Estas comparações incluem os objetivos que foram alcançados, o tempo que demorou, o custo total, a cobertura dos testes e os problemas que foram encontrados.
    a. Critério de Entrada - Resultados de testes e logs de todas as fases anteriores.
    b. Critério de Saída - Report aprovado e entregue.

Quando é que as atividades de QA devem começar?

As atividades de QA devem começar logo desde o inicio do projeto. Quanto mais cedo as mesmas começarem mais beneficio trazem facilitando a obtenção dos parâmetros de qualidade desejados, quanto mais demorarem a ser implementadas as atividades de QA mais tempo, dinheiro e esforço para alcançar o standard de qualidade.

O que é um bom test case?

Os test cases devem ser transparentes, eficientes e adaptáveis. Cada teste tem de ter inputs, procedimentos, execução, condições e resultados esperados.

Como é que defines e escreves um bom test case?

1. Simples e transparente.

2. Reutilizável.
3. ID’s únicos.
4. Processo de revisão e aprovação.
5. Utilizador final ou os requerimentos em consideração.
6. Resultados esperados bem definidos.

Já usaste alguma ferramenta de automação?

Sim. Appium, Selenium, TestProject, Jenkins, STB-Tester.

Qual é a diferença entre severidade e prioridade?

- Severidade - É o grau de impactos que um bug tem no desenvolvimento ou operação de um componente ou sistemas.
- Prioridade - É o nível de importância ao nível do negócio atribuído a um elemento ou sistema.

Qual é a diferença entre Assert e Verify na automação de testes?

Enquanto que no assert o teste para se não for retornado o valor esperado no verify o test case correr até ao fim mesmo que as condições não sejam verificadas.

O que é um Use Case?

Use Case é uma metodologia usada para analisar sistemas e clarificar e organizar requisitos.

O que é um Test Plan?

Um Test Plan é um documento de abordagem sistemática que define os testes a ser executados num determinado sistema. Neste definem-se os detalhes dos testes e workflows do mesmo.

O que é que inclui um Test Plan?

Um Test Plan inclui:

1. Descrição do Produto.
2. Objetivos.
3. Estratégias de Teste.
4. Scope.
5. Cronograma.
6. Recursos de Teste.
7. Resultados a entregar.

O que é uma Test Strategy?

Uma Test Strategy é uma guia para os objetivos que têm de ser alcançados nos testes e como os executar em conforme com o definido no Test Plan. Ajuda com a análise de risco, planos de competência e objetivos dos testes.

Um Test Plan e uma Test Strategy são a mesma coisa?

Um Test Plan é um documento que abrange o scope e as diferentes atividade que envolvem os testes de um projeto. Uma Test Strategy é um documento mais high level que engloba as diretrizes e principio envolvido na realização do processo de testes.

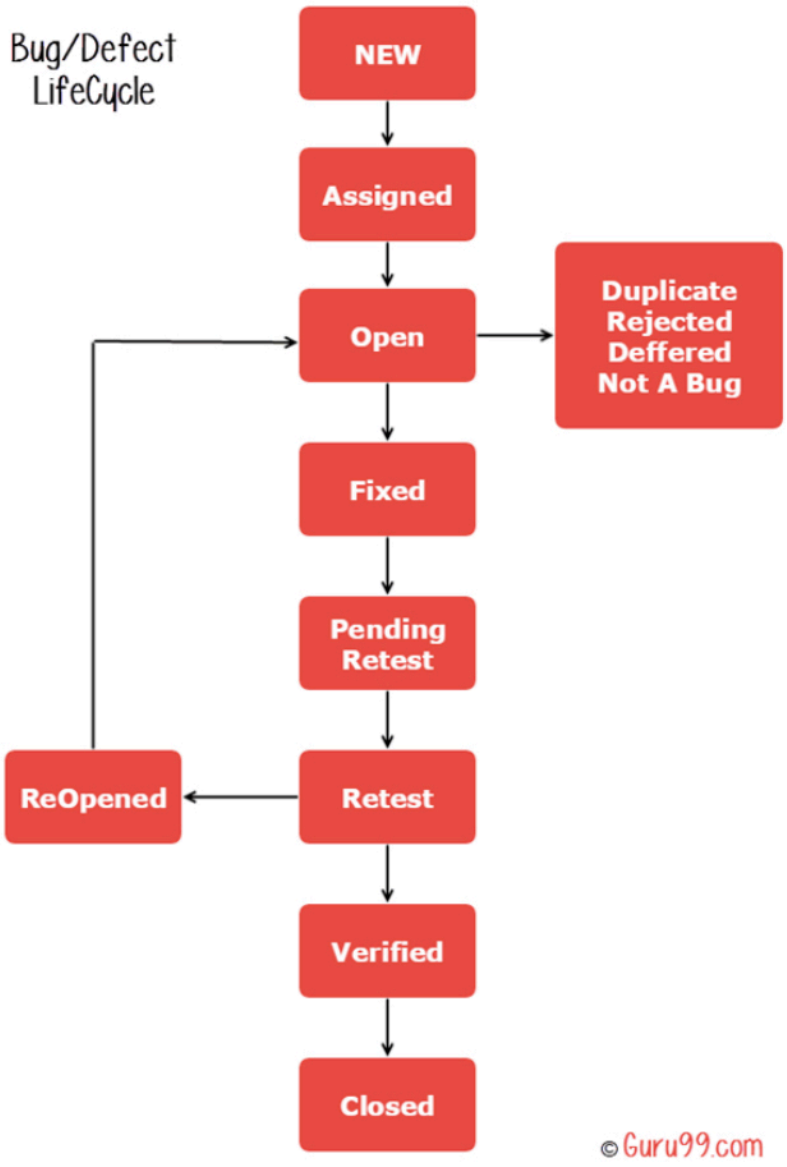
O que é um testware?

Testware é um termo utilizado para definir a categoria de softwares que ajudam a testar aplicações e projetos, sendo também visto como o ambiente em que são corridos os testes.

O que é um bug?

Um bug é um erro ou defeito na execução de um programa informático.

Quais são as fases do bug cycle?



- New - Quando um defect é reportado pela primeira vez.
- Assigned - Quando o bug é criado pelo tester é aprovado pelo lead testes e assigns o bug a developer team.
- Open - Quando os developers começam a analisar e a trabalhar no fix do bug.
- Fixed - Quando os developers fazem as alterações necessárias ao código e verificam que possivelmente reparou o bug reportam o bug como Fixed.
- Pending retest - Quando o defect é fixed pelo developer é requerido ao tester que abriu o bug que volte a testar para verificar se o bug foi corrigido, ficando

a espera que o tester o teste.

- Retest - Altura em que o tester volta a testar o bug após o fix.
- Verified - Após o retest é alterado o status para verified se não for encontrado novamente nenhum bug no teste que foi executado para o despoletar em primeiro lugar.
- Reopen - O status passa para Reopen se verificar que o fix que os developers deram não corrigiram o bug que foi reportado.
- Closed - Se o bug não existir então o tester pode passar o seu status para closed.
- Duplicate - Este status é atribuído se o defect se repetir variadas vezes ou se já estiver reportado o mesmo conceito de bug noutro ticket.
- Rejected - Se o developer entender que o defect não é genuíno ou relacionado com a utilização do produto atribui o status de Rejected.
- Deferred - Se o bug apresentado não for de extrema importância e já for esperado a sua resolução numa próxima release então é assinado o status de Deferred.
- Not a bug - Se não afeta a funcionalidade da aplicação testada então o bug pode ver alterado o seu status para Not a bug.

Perguntas Relacionadas com Testes

Qual é a diferença entre testes funcionais e testes não funcionais?

Os testes funcionais verificam cada função de um software, enquanto um teste não funcional verifica aspetos não funcionais como o desempenho, usabilidade, confiabilidade, etc. Outra questão relevante é que os testes funcionais podem ser feitos manualmente, enquanto que que alguns testes não funcionais que avaliam questões como a performance são difíceis de executar de forma manual.

O que é significa verificação e validação?

A verificação é o processo de verificar se um software ou produto atinge o seu objetivo sem bugs. É o processo que garante se o produto desenvolvido está a funcionar de forma correta atendendo aos requisitos solicitados. Ou seja a verificação reponde a pergunta: Estamos a construir o produto bem?

A validação é o processo que valida se um software ou produto está a par com os requisitos solicitados ao mais alto nível. Este processo valida o produto e valida também se o que se está a construir é o produto correto e esperado pelos stakeholders. Ou seja a validação responde a pergunta: Estamos a construir o produto correto?

Verificação	Validação
Inclui a verificação de documentos, design, códigos e programas.	Inclui testar e validar o produto real.
A verificação é o teste estático.	A validação é o teste dinâmico.
Não inclui *a* execução do código.	Inclui a execução do código.
Os métodos usados na verificação são revisões, orientações, inspeções e verificação de mesa.	Os métodos utilizados na validação são Teste de Caixa Preta, Teste de Caixa Branca e teste não funcional.
Ele verifica se o software está em conformidade com as especificações ou não.	Ele verifica se o software atende aos requisitos e expectativas de um cliente ou não.
Ele pode encontrar os bugs no estágio inicial do desenvolvimento.	Ele só pode encontrar os bugs que não foram encontrados pelo processo de verificação.
O objetivo da verificação é a arquitetura e especificação do aplicativo e do software.	O objetivo da validação é um produto real.
A equipe de garantia de qualidade faz a verificação.	A validação é executada no código do software com a ajuda da equipe de teste.
Ele vem antes da validação.	Ele vem após a verificação.
Consiste na verificação de documentos/arquivos e é realizada por humanos.	Consiste na execução do programa e é realizada por computador.

Qual é a diferença entre load testing e stress testing?

O load testing é usado para descobri o limite superior de um sistema ou aplicação, tentando perceber quais são as suas limitações relativamente aos recursos internos que dispõe. Já um teste de stress é realizado para perceber quais são os comportamentos de um sistema ou aplicação quando colocado sobre pressão, seja pelo fator de tempo de uso como pelo fator de processamento necessário, os principais fatores que são testados neste tipo de teste são a estabilidade e robustez do sistema.

Quais são os diferentes métodos de testagem e quando é que os devemos usar?

1. Unit Testing - É usado para testar unidades individuais do código. O seu principal objetivo é determinar a integridade lógica ou seja que um determinado pedaço de código faz o que é suposto fazer. Exemplos: JUnit, PyUnit, PyTest, Mocha, Jest, Karma.
2. Integration Testing - É usado para testar o comportamento de vários componentes ao funcionarem em conjunto. O seu objetivo principal é assegurar que as relações criadas entre cada componente são integras e que existe a transmissão de dados entre os vários componentes.
3. End-to-End Testing - Também chamados de System Tests, focam-se em verificar o comportamento de um sistema de uma ponta a outra ou seja ter uma visão geral do produto na sua forma global. Pode se dizer assim que enquanto que os Unit e Integration Tests são “white box” tests porque se sabe o que integra cada componente do código os E2E Tests acabam por ser “black box” tests pois só sabemos os inputs que colocamos e os outputs que esperamos no entanto não temos noção do que possa integrar o componente num todo. Exemplos: Cucumber, Postman, SoapUI, Karate, Cypress, Katalon.
4. Acceptance Testing - É uma metodologia que é usado no ciclo de desenvolvimento e no qual o seu principal objetivo é verificar que um determinado produto ou feature foi desenvolvida de acordo com as especificações do produto. É neste tipo de testes que se usam as fazes como α-testing ou β-testing.

5. White Box Testing - Também chamado de testes estruturais descreve o tipo de testes onde os componentes do código são conhecidos. Como já dito antes exemplos de White Box Testing são os Unit Test e os Integration Tests.
6. Black Box Testing - Também chamado de testes funcionais ou de comportamento, é descrito como o grupo de testes onde não temos noção do código que está ser executado para fazer a aplicação funcionar. Como já referido neste tipo de testes só sabemos o tipo de inputs que colocamos e o tipo de output que podemos esperar. Exemplos deste tipo de testes são os End-to-End Tests.
7. Gray Box Testing - Este tipo de testes são uma combinação híbrida entre os black box tests e os white box tests. Assim sendo o Gray Box Testing utiliza a facilidade e simplicidade do Black Box Testing em que escolhemos um input e tentamos obter um output e o direcionamos a uma funcionalidade especifica do código na qual sabemos como é escrito e executado. A razão pelo qual o Gray Box Testing existe é porque tanto o Black Testing como o White Testing podem falhar a verificação de algumas funcionalidades importantes. Pois o Black Testing só testa a obtenção de outputs consoante os input que lhe damos e não testa a integridade dos componentes internos, podendo estar a obter o output esperado por mero acaso. E o White Testing só testa a integridade de unidades individuais e em como elas funcionam juntas, sendo insuficiente para encontrar defects que possam estar espalhados pelo sistema ou em vários componentes.
8. Manual Testing - Os testes manuais são executados por um utilizador que especifica o input e obtém um output. Esta metodologia de testes é lenta e propensa a erros. Hoje em dia é mais utilizada no beta testing de produtos para fazer o Acceptance Testing e procurar por defects relacionados com casos de extremo ou relacionados a algum nicho.
9. Static Testing - É descrito como qualquer teste em que não existe nenhum tipo de código a ser executado. Isto inclui atividades como a revisão de código em que se verifica a lógica e a integridade das funções e classes. Geralmente é também uma metodologia muito lenta e susceptível a erros como nos testes manuais.
10. Dynamic Testing - É descrito como os testes em que código está a ser executado. De forma geral todos os tipos de testes referenciados

anteriormente a excepção dos testes manuais são testes dinâmicos.

11. UI/Visual Testing - É definido com o tipo de testes que avalia a integridade e o comportamento dos elementos da UI. Exemplos: Selenium, Cypress, TestCafe, SauceLabs, Katalon Studio, Robot, Appium.
12. Smoke Testing - Refere-se a um grupo especifico de testes que serve para verificar se um sistema está a funcionar de forma razoável. Normalmente é escolhido um conjunto de testes não exaustivo que tentam validar as funcionalidades core de uma determinada aplicação.
13. Regression Testing - É uma metodologia de testes que verifica se alguma das features dos testes funcionais anteriores deixou de funcionar. Isto normalmente inclui correr o conjunto total de todos os testes unitários, de integração e de sistema para garantir que nada mudou. Este tipo de teste consome bastante tempo o que leva a normalmente ser feito apenas Smoke Tests. Quando é estabelecido uma pipeline de CI/CD são executados smoke tests a cada commit feito para a branch principal ou secundária, sendo que se reserva os testes de regressão para intervalos de tempo maiores ou quando é lançada uma feature nova para que se faça a Continuous Integration sem problemas.
14. Load Testing - Refere-se ao tipo de testes que testa a resposta de uma aplicação a um aumento da sua procura ou da sua utilização. Este tipo de testes procura simular um aumento de requests numa determinada aplicação ou até um ataque DDOS.
15. Penetration Testing - É uma forma de teste de segurança que procura verificar a robustez da segurança de uma aplicação.

Quais são as diferenças entre negative testing e positive testing?

O Positive Testing determina que a aplicação funciona como esperado ou seja se houver algum erro o teste falha. Já o Negative Testing garante que a aplicação consegue reagir bem a inputs inválidos ou a comportamentos inesperados do utilizador, por exemplo se um utilizador tentar por um número onde só se pode por letras o resultado esperado seria ser mostrado uma mensagem de erro que avisa o utilizador para inserir apenas letras. O propósito do Negative Testing é

detetar e prevenir situações em que a aplicação poderia deixar de funcionar corretamente.

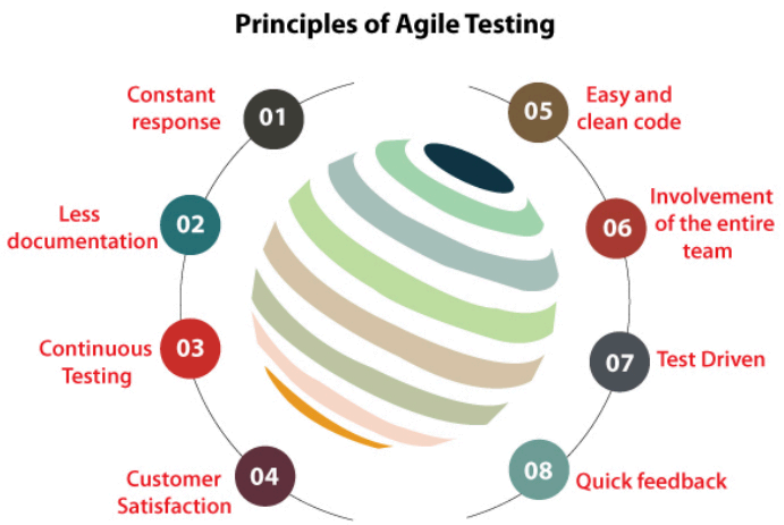
Perguntas de Workflow

O que é Agile testing?

É uma prática de testagem que segue as regras do desenvolvimento Agile. Ao contrário da metodologia Waterfall, o Agile Testing começa no inicio do projeto com Continuous Integration entre a equipa de desenvolvimento e a equipa de testes.

Os princípios Agile envolvem:

1. A comunicação continua entre as equipas.
2. Menos documentação.
3. Testagem continua.
4. Satisfação do cliente.
5. Código fácil e limpo.
6. Envolvimento da equipa toda.
7. Desenvolvimento baseado em testes.
8. Feedback rápido.



Consegues descrever a diferença entre Scrum e Agile?

A principal diferença entre Agile e Scrum é que, embora Agile seja uma filosofia de gestão de projetos que utiliza um conjunto central de valores ou princípios, o Scrum é uma metodologia Agile específica que é usada para facilitar um projeto.

Quão relevante é a testagem manual num workflow de automação?

Qualquer sistema ou aplicação deve ser testados manualmente antes de automatizar o teste. Principalmente, ajuda a garantir a qualidade do aplicativo, garantindo que cumpre com os requisitos do sistema.

Perguntas Direcionadas

Alguma vez escreveste test cases sem documentação?

Sim por motivos de não acesso a documentação ou por falta de organização e experiência que tinha no início.

Que métodos de QA usas e porque?

Sempre desenvolvi testes de automação com base no titulo ou test steps especificado.

Se tivesses de executar uma grande quantidade de testes numa janela de tempo pequena como é que o farias?

Tentaria perceber quais são as features principais da aplicação e correr os testes que conseguissem abranger não só o maior número de features como também as mais importantes.

Quais são os desafios que as Equipas enfrentam na automação durante a testagem?

Os maiores desafios das equipas acabam por ser:

- Comunicação Efetiva e Colaboração.
- Escolher as Ferramentas Certas.
- Necessidade de Recursos Qualificados.
- Escolher a Abordagem de Testes Adequada.
- Necessidade de Grande Investimento.

Perguntas de Resposta Aberta

Quais são as diferentes técnicas de verificação que conheces?

1. Inspeção - É uma vista não invasiva do sistema e dos elementos que podem incluir medições ou manipulação física.
2. Demonstração - Este método involve o uso de um elemento ou sistema de forma especifica de modo a produzir um resultado esperado.
3. Testar - A verificação por testes foca-se na obtenção de resultados com base nos inputs e dados que lhe fornecemos.

4. Análise - Usa as ferramentas de análise para detetar prontos fracos que podem causar problemas no futuro.