1. **Testes e cenários de testes**

* **BDD – Desenvolvimento guiado por comportamento**

A semântica para estruturar esse cenário de teste utiliza as palavras-chave **dado**, **quando** e **então** da seguinte maneira:

**Dado** - Pré-condições devem ser verdadeiras para que eu execute o teste?

**Quando** - Qual ação será executada no sistema que fornecerá o resultado validado?

**Então:** - De acordo com a ação disparada, qual será o resultado esperado?

* **Caso 1:**
  + **Dado:** Que eu tenho um usuário no sistema AluraPic
  + **Quando:** Eu digitar meu nome de usuário
  + **E:** Senha corretamente
  + **E:** Apertar o botão confirmar
  + **Então:** Devo ser redirecionado par página principal do sistema
* **Caso 2:**
  + **Dado:** Que eu tenho um usuário no sistema
  + **Quando:** Eu não digitar meu nome de usuário
  + **Então:** Devo ser informada da obrigatoriedade do campo
* **Caso 3:**
  + **Dado:** Que eu tenho um usuário no sistema
  + **Quando:** Eu digitar meu nome de usuário
  + **E:** Não digitar minha senha
  + **Então:** Devo ser informada da obrigatoriedade do campo
* **Caso 4:** 
  + **Dado:** Que eu tenho um usuário no sistema
  + **Quando:** Eu digitar meu nome de usuário
  + **E:** Senha incorreta
  + **E:** E apertar o botão de confirmar
  + **Então:** Devo ser informada que a senha está incorreta
* **Caso 5:** 
  + **Dado:** Que eu tenho um usuário no sistema
  + **Quando:** Eu digitar meu nome de usuário
  + **E:** Senha incorreta pela terceira consecutiva vez
  + **E: apertar o botão de confirmar**
  + **Então:** Devo ser informada que preciso esperar 15 minutos para tentar logar novamente.
* Testes funcionais, são os testes definidos de acordo com os requisitos funcionais do software;
* Podemos pensar nos nossos testes construindo fluxos de testes ou tabelas de decisão conforme as regras de negócio do nosso projeto;
* Conceitos de Cenários de testes definem “O que” deve ser testado, enquanto os Casos de Testes definem “Como”, ou seja, o passo a passo. O caso de teste é mais detalhado;
* Como escrever Casos de testes utilizando BDD, sigla para Behavior driven development (desenvolvimento guiado por comportamento ou desenvolvimento orientado a comportamento, numa tradução livre) com intuito de escrever testes guiados pelo comportamento do usuário e pelos negócios. Para tal, usamos a semântica Dado, Quando e Então;
* Um Plano de teste define e comunica a intenção e esforço do teste. Serve, por exemplo, para ganhar a aceitação e aprovação dos envolvidos, comunicar e justificar prazo de teste planejado.
* DISC

1. **Qualidade**

* Qualidade de software, pode ser entendida como um conjunto de características a serem satisfeitas, de modo que o produto atenda às necessidades de seus usuários. Falamos de fazer o software certo para o usuário certo;
* Num time ágil, o papel de QA exige ser generalista, olhar o software como um todo de maneira que minimize custos, tempo e efeitos indesejáveis. Isso deve ser feito o mais breve o possível, ajudando assim a construir um projeto melhor;
* No ciclo de desenvolvimento de software em cascata as etapas do projeto são feitas uma após a outra de forma síncrona. Portanto, neste esquema é dada muita ênfase às fases de análise e projeto antes de partirmos para a programação de fato. Uma desvantagem é que dessa forma fica difícil prevenir erros, dado que a etapa de testes fica apenas no final;
* Já nos modelos ágeis, a abordagem iterativa enfatiza a entrega rápida de uma aplicação em componentes funcionais completos, com entregas menores. A pessoa responsável de QA consegue participar de todo o processo de desenvolvimento;
* Critérios de aceite são os valores máximos ou mínimos aceitáveis para uma funcionalidade ser aceita;
* Definição de pronto (DoD - definition of done) são pontos que definem se uma atividade foi concluída ou não. Ela deve valer para o projeto como um todo.

1. **Tipos de testes**

* **Teste de regressão** é uma técnica de teste a ser aplicada quando surgem novas versões mais recentes do software e garante que não surjam novos defeitos em componentes já analisados;
* **Teste de sanidade** é o subconjunto do teste de regressão e também é realizado quando não temos tempo suficiente para fazer o teste mais elaborado. Ele tem um nível superficial e verifica se as funcionalidades mais críticas do sistema estão conforme o esperado;
* **Testes de limite** consistem em testar os valores mínimo e máximo (ou primeiro e último valores) de uma partição. Essa técnica é geralmente usada para testar requisitos que exigem um intervalo de números (incluindo datas e horas);
* **Teste de transição** de estados é utilizado para testar a capacidade que o software tem de entrar em e sair de estados definidos através de transições válidas e inválidas;
* **Testes não-funcionais** tem como objetivo testar aspectos do software que não são associados a funcionalidades. Ex: escalabilidade, desempenho, segurança;
* **Testes de performance** são um conjunto de testes que visam analisar o desempenho. Entre eles, temos:
  + **Teste de carga**, que tem como objetivo verificar o desempenho de um sistema quando ele é submetido a cargas variáveis de usuários ou transações.
  + **Teste de capacidade**, parecido com o de carga, tem como objetivo identificar os limites da aplicação, ou seja, quantos usuários simultâneos ou chamadas por segundo a aplicação é capaz de suportar dentro dos parâmetros de qualidade definidos.
  + **Teste de stress**, verifica a performance de um sistema quando é submetido a cargas que estão no limite ou acima do limite especificado inicialmente.
  + **Teste de usabilidade**, tem como objetivo observar usuários reais usando o software para descobrir problemas e pontos de melhorias.

1. **Pirâmide de Testes**

* O teste de caixa branca é usado para testar um sistema de software com base na sua arquitetura. Esse tipo de abordagem serve para testar cada módulo do sistema de acordo com seu código. Pode ser feito pelos próprios desenvolvedores ou QAs técnicos que entendam da estrutura do projeto. Um exemplo de testes de caixa branca são: teste unitário, de integração de módulos e de serviço (API);
* O teste de caixa preta é usado para testar a funcionalidade do sistema, independentemente de seu código. Seu objetivo principal é garantir que os requisitos sejam atendidos. Por exemplo: testes funcionais e regressivos;
* A pirâmide de testes é importante porque define níveis de testes e nos dá um norte quanto à quantidade de testes necessários em cada um dos níveis. No topo, temos os testes de ponta a ponta, no meio, teste de integração e na base, testes de unidade. Precisamos lembrar que a base da pirâmide compreende testes mais fáceis e rápidos para executar, enquanto o topo, mais difíceis e lentos;
* Evidência de testes são imagens e/ou vídeos que comprovam que um determinado teste foi executado e o resultado esperado foi obtido. Podem ser screenshots de uma determinada tela ou o vídeo de um software em funcionamento. São importantes para documentar os testes realizados, servindo de insumos e métricas.

1. **Gestão de erros**

* Que um bug é tudo que acontece no sistema diferentemente do que foi especificado como o ideal. Um defeito é um erro encontrado num código ou num documento e uma falha é o resultado ou manifestação de um ou mais defeitos;
* Que ao encontrar um bug é importante reportá-lo imediatamente, reproduzir o bug ao menos três vezes e testar a ocorrência dele em outras plataformas e módulos similares;
* Que o custo de corrigir um bug é muito maior do que criar mecanismos para evitá-los, então devemos prevenir bugs nos atentando ao sistema desde sua documentação;
* O reporte de bugs consiste em descrever bem o bug de forma precisa a fim que todo o time consiga reproduzi-lo;
* Classificamos bugs no sistema por severidade. Então, eles se dividem em: blockers (que bloqueiam nossos testes), críticos e graves, moderados e pequenos. Especificamos a severidade no nosso plano de teste, considerando o impacto deles no sistema.

1. **Estratégia de testes**

* A arquitetura de um sistema consiste na definição dos componentes, suas propriedades e seus relacionamentos com outros softwares. A documentação da arquitetura do software facilita a comunicação entre os membros dos times, pois registra as decisões iniciais acerca do projeto;
* Temos que definir um escopo de testes, já que nem todos os testes serão responsabilidade da pessoa de QA;
* Constrói-se um escopo respondendo a perguntas importantes como: o que ele deixa de fora? Quais níveis de teste vão ser executados? De quem será a responsabilidade de cada teste? Haverá testes automatizados? Quais serão as ferramentas usadas e como será feito o lançamento desse sistema?

1. **Refinamentos e conclusões**

* Para criar uma estimativa para seus testes é preciso considerar fatores como a complexidade, domínio do software, detalhamento da documentação e quantidade de retrabalho;
* É importante metrificar a qualidade do sistema para saber em quais pontos acertamos em quais precisamos implementar melhorias;
* Existem métricas de qualidade como: total de Defeitos, total de defeitos em produção, total de defeitos removidos, tempo médio de reparo, quantidade de testes automatizados, cobertura de código e satisfação do usuário.