Lista de de testes a realizar para verificar integridade do software

1. Cliques aleatórios. Entregar também a aplicação a outra pessoa, na parte do programa que se quer testar, é uma boa opção. O utilizador final não utiliza a aplicação da mesma forma e poderá descobrir erros escondidos.

2. Utilizar dispositivos diferentes (com diferentes sistemas operativos) para serem encontrados problemas de compatibilidade, tais como diferentes browsers.
3. Introduzir dados (como datas, nomes, etc) que possam parecer ridículos quando for necessário preencher algumas informações/ na barra de pesquisa.
4. Teste de compatibilidade com dispositivos móveis.

5. **Teste de funcionalidade básica** - Certificar-se de que todos os botões de todas as telas funcionam, garantindo que nenhum destes causa crash por si só. O objetivo aqui é o seguinte: não se pode permitir que outras pessoas toquem no software se ele falhar assim que inserir o próprio nome no campo de nome de usuário. É necessário executar testes para garantir que a funcionalidade básica da API funciona antes de enviá-la para testes mais intensivos.

6. **Revisão de código** - Outro par de olhos a olhar para o código-fonte pode revelar muitos problemas. Se a sua metodologia de codificação requer revisão por pares, execute este passo antes de fazer o código para testes. Lembre-se de fazer seu teste de funcionalidade básica antes da revisão de código.

7. **Análise de código estático** - Existem ferramentas que podem analisar o código-fonte ou bytecode sem executá-lo. Essas ferramentas de análise de código estático podem procurar muitas deficiências no código-fonte, como vulnerabilidades de segurança, por exemplo. Use ferramentas de análise de código estático para impor padrões de codificação e configure essas ferramentas para executar automaticamente parte da compilação.

8. **Teste de unidade** - Os developers deverão escrever testes de unidade para garantir que a unidade (seja um método, classe ou componente) esteja a funcionar conforme o esperado e seja testada em toda a faixa de entradas válidas e inválidas. Em um ambiente de integração contínua, os testes de

unidade devem ser executados toda vez que você se comprometer a alterar o repositório do código-fonte, e você também deve executá-los em sua máquina de desenvolvimento.

Os developers também trabalham com objetos simulados e serviços virtualizados para garantir que suas unidades possam ser testadas independentemente. Se seus testes de unidade falharem, corrija-os antes de permitir que outra pessoa use seu código.

9. **Teste de desempenho de usuário único** - Algumas equipes carregam testes de carga e desempenho em seu processo de integração contínua e executam testes de carga assim que o código é verificado. Isso é particularmente verdadeiro para o código de back-end. Mas os desenvolvedores também devem observar o desempenho do usuário único no front-end e garantir que o software seja responsivo somente quando eles estiverem usando o sistema. Se demorar mais do que alguns segundos para exibir uma página da web tirada de uma Web local ou emulada (e, portanto, responsiva), o software não está pronto.

10. **Usabilidade-** A capacidade do produto de software de ser compreendido, aprendido e utilizado pelo usuário, uma das formas de testar é usar várias pessoas de diferente faixas etárias e/ou diferentes conhecimentos tecnológicos para avaliar a sua usabilidade.

11. **Procura de erros de navegabilidade -** Procurar links para outras seções do produto ou para outras páginas que não estão a funcionar corretamente.

12. **Teste de Compatibilidade -** Verificar o produto em diversos browsers e sistemas operativos diferentes, de modo a perceber se está otimizado para todos eles.