Por que e o que é possível testar?

Larissa Gabriela
26/12/2021

Confira neste artigo:

Assim que iniciamos nosso trabalho como desenvolvedor ou desenvolvedora, é comum pensarmos que quando criamos um software ele está finalizado assim que escrevemos a última linha de código. Neste momento não nos atentamos a uma etapa muito importante do processo de desenvolvimento, **o teste**. Em resumo, o teste é a etapa em que verificamos as falhas (os famosos bugs) que nossa aplicação pode ter.

Testes são tão importantes que temos um profissional especializado para essas situações. Aqui introduzimos um termo muito comum, o **Quality Assurance (QA)**. A tradução para ele seria algo como “garantia de qualidade” se referindo ao profissional ou área que irá garantir que um produto ou serviço está sendo oferecido da melhor forma o possível.

Definir testes em poucas linhas é complicado porque podemos abordar diversos tipos para diferentes aplicações; como mobile, frontend, backend, segurança dentre outras. Neste artigo você encontrará uma explicação ainda mais completa sobre as mais comuns.

Queremos convencê-lo do porquê de testar, além de apresentar o que é possível testar. Será que eu preciso testar mesmo o que eu não vejo?

Mas vamos lá, por que testar?

Vamos pensar em exemplos práticos.

- Supondo que você trabalhasse no setor financeiro de uma empresa, você acabaria lidando com pagamentos e possivelmente uma quantidade razoável de dinheiro. O ideal é que possíveis senhas e dados pessoais de clientes ou funcionários estivessem bastante seguros, certo?

nossa aplicação.

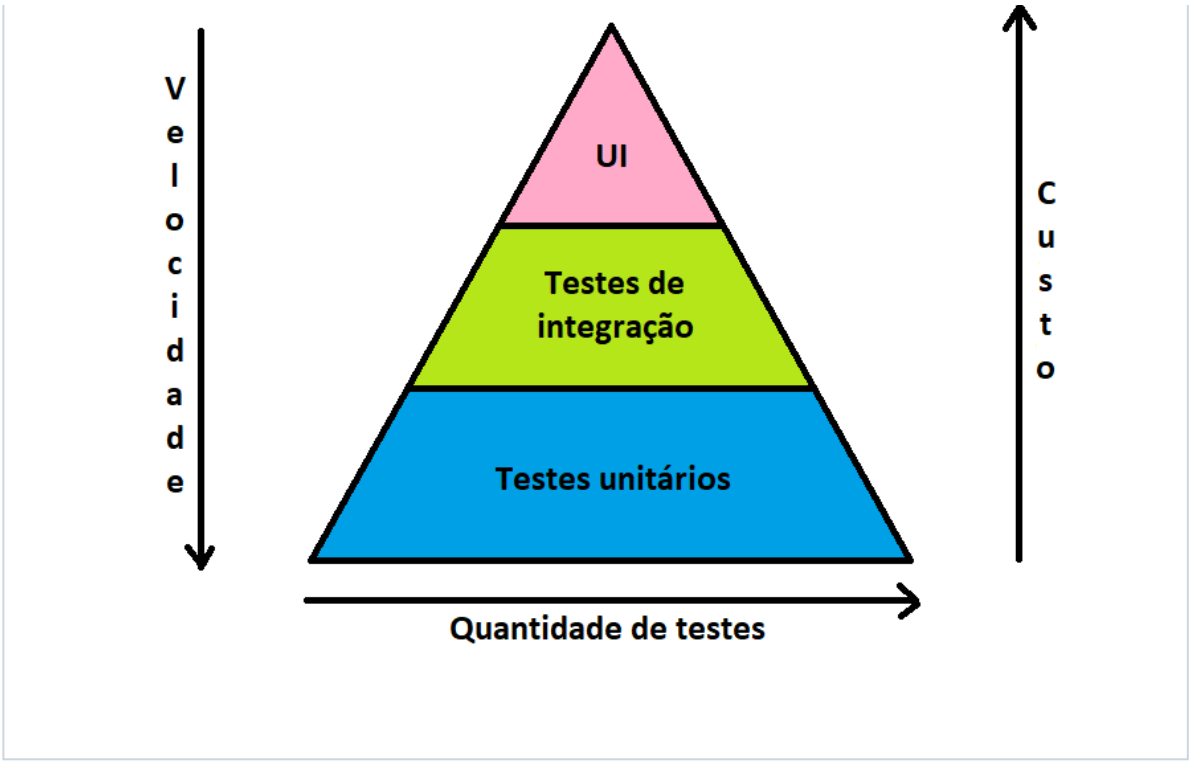
Geralmente testes não são gratuitos e demandam tempo, mas ter esses dados expostos poderia nos prejudicar ainda mais, seja financeiramente ou até diminuindo a credibilidade da nossa empresa. Até problemas que nos expusessem menos poderiam ser prejudiciais

- Imagine que a página de login de uma plataforma de streaming estivesse com um bug bem na hora que você quisesse assistir a sua série favorita. Você teria uma experiência ruim e talvez não recomendasse aquela plataforma. A credibilidade do produto diminuiria cada vez mais, caso o problema continuasse.

Vimos exemplos de diferentes áreas. Então é possível testar tudo? Vale a pena fazer testes no frontend? E no backend?

Pirâmide de testes

Antes de analisar área por área, vamos pensar juntos na teoria e na lógica do uso de testes. Para isso temos a **pirâmide de testes**. Ela foi idealizada por Mike Cohn em seu livro “Succeeding with agile”. A ideia é ser uma metáfora relacionando a complexidade, o tempo, entre outros fatores para auxiliar na jornada de testes. Veja a pirâmide abaixo:



Nela vemos então 3 divisões:

- Testes unitários
- Testes de integração
- Testes de experiência do usuário

De acordo com a pirâmide, a quantidade de testes que desenvolvemos está diretamente ligada à velocidade e ao custo deles. Isso porque testes unitários, por exemplo, têm a realização mais barata, logo, uma maior quantidade deles pode ser feita e eles compõem a nossa base. Já os testes de experiência de usuário com a interface (ou UI de User Interface) são mais custosos e lentos de realizar, o que os torna menos viáveis. Então, serão feitos em menor quantidade.

Levando essas informações em consideração, continuaremos analisando área por área.

Frontend

No frontend, um termo muito utilizado é **componentes**. Os componentes são partes de um software que possuem funcionalidades que independem de outras partes. Ou seja, é uma unidade que quando somada a outros componentes pode compor sistemas mais elaborados. Ligando a definição de componentes e testes unitários, faz sentido eles trabalharem juntos, certo? Mas e quando juntamos os componentes e formamos esse sistema?

Nesse caso, pode ser necessário o **teste de integração**. Como o próprio nome já diz, ele irá testar se pequenas partes do seu código estão funcionando bem em conjunto.

Vamos voltar àquela tela de login que usamos como exemplo. Poderíamos testar se o campo de preenchimento do login estaria funcionando e depois testar separadamente o campo para a senha. Para testar os campos individualmente, podemos utilizar os testes unitários. Mas os dados preenchidos devem ter uma relação entre si. Quando clicamos em um botão de “Entrar”, por exemplo, queremos receber uma mensagem avisando se tivemos sucesso ou alguma mensagem que remeta ao erro do usuário que tiver ocorrido. Nesse caso, caberia o teste de integração.

Mas podemos ir além, pois temos também o tipo de teste conhecido como **E2E (end-to-end)**. Esse teste irá verificar toda a jornada do usuário. Através de algumas ferramentas é possível simular possíveis caminhos que poderiam ser utilizados e verificar se a jornada está adequada e fluindo da forma correta.

Ao longo de todos os tópicos pelos quais passaremos no artigo, vamos tentar deixar dicas de ferramentas que podem ser utilizadas em cada área.

Uma ferramenta muito conhecida na área de testes em frontend é a **Cypress**. Com ela é possível fazer os testes citados utilizando diversos frameworks e bibliotecas — React, Angular e Vue, por exemplo — que utilizam o navegador para execução.

Vale lembrar que existem diversos outros tipos de testes e ferramentas. Entretanto citamos os mais comuns.

Outras ferramentas:

Backend

Pensar em testes em backend pode parecer um pouco menos palpável do que para front. O frontend é algo visível, já que temos uma interface gráfica.

apenas com testes de front, para analisar a parte de backend de uma aplicação, precisaríamos do front pronto. Ou seja, seria possível ter que aguardar mais tempo para fazer testes que poderiam ser antecipados. Além disso, conhecer os dois dados permite uma análise mais minuciosa.

Podemos nos perguntar: mas o que eu preciso saber para conseguir testar um backend? Entender bem sobre a linguagem utilizada, os objetivos daquela aplicação e as ferramentas de desenvolvimento são fatores importantes para isso. Dessa forma, a chance de algo passar despercebido diminui bastante.

Assim como no frontend, no backend também temos ferramentas voltadas especificamente para ele. Deixamos aqui algumas dessas ferramentas para que você possa explorar.

Além dos testes que já conhecemos, quando pensamos em aplicações mobile, outros testes serão necessários e evitarão falhas.

Nessas aplicações lidamos com algumas dificuldades que no desenvolvimento web não são tão críticas. Temos que lidar com configurações de hardware diferentes, e elas são só um exemplo, pois há diversas outras configurações. Além disso, temos ainda mais versões de um mesmo sistema operacional.

No quesito ambiente operacional, precisamos fazer o **teste de compatibilidade** para garantir que estamos interagindo da forma esperada com ele. Isso pode evitar que um aplicativo fique com aquela navegação difícil, com lentidão.

Temos também o **teste de conectividade**. Nele percebemos como nosso aplicativo funciona quando está conectado à internet e quando não está. Podemos ver se algum aviso de erro na conexão com a internet aparece ou se nada é avisado pelo usuário, por exemplo.

O **teste de performance** permite observar o uso do processador, da memória, da bateria e de outros componentes quando nosso aplicativo está em uso. Dessa forma podemos melhorá-lo para demandar menos uso do hardware, uma vez que entenderemos a relação entre eles.

O mar de testes possíveis em mobile é gigante. Deixamos listados outros testes que podem ser explorados. São eles: **teste de interface, teste de low-level resource, teste operacional, teste de instalação, teste de segurança** e assim por diante.

Assim como para outras aplicações, temos abaixo exemplos de ferramentas que podem ser utilizadas para testes em aplicações mobile.

Segurança

No início desse artigo falamos sobre o setor financeiro de uma empresa e o quanto um vazamento de informações poderia prejudicar esse negócio. Para evitar um caso como esses, devemos usar o **teste de segurança**. Por meio dele verificamos o comportamento do nosso software mediante a tentativas incorretas de uso. Geralmente, a pessoa que testará essa aplicação agirá como um hacker e buscará por vulnerabilidades no sistema que colocam os dados em risco.

Os testes podem incluir, por exemplo, um **teste de vulnerabilidade ou teste de intrusão**, entre outros. No **teste de vulnerabilidades** é feita uma varredura de falhas, como erros no armazenamento das informações ou ainda na configuração do sistema de nuvem utilizado, geralmente algo que ocorreu por meio do uso de um software.

Já o **teste de intrusão**, em geral, é um teste manual onde um especialista busca determinar até onde um hacker ou outro tipo de ameaça pode chegar na sua aplicação.

Bem importante, certo? Assim conseguimos manter e evitar grandes problemas!

Quando não testar?

Falamos bastante sobre o que testar e diferentes aplicações para diversos tipos de testes. Mas será que realmente tudo precisa ser testado?

Para responder essa pergunta precisamos voltar à pirâmide de testes. Quando estudamos essa pirâmide, vimos que **testes custam dinheiro e tempo**. Portanto, sempre precisaremos pesar esses fatores.

Em casos de prazos apertados para a entrega de um projeto, nos deparamos com outro problema. Como sabemos, testes demandam um tempo razoável, e a data limite para a entrega do trabalho pode comprometer esse processo. Algumas alternativas podem ser entregar menos, porém com qualidade (se isso for viável), ou tentar pelo menos realizar os testes mais simples.

Esses são apenas exemplos de situações que podem comprometer essa etapa do desenvolvimento de um software. Mas lembre-se que o teste deve ser feito sempre que possível, levando em consideração o tempo e o custo e como podemos adaptá-los à nossa situação.

Conclusão

Fico feliz que tenha acompanhado essa jornada de aprendizado até aqui. Como vimos, o teste vai muito além do que é visto. Todas as etapas de produção de um software devem ser testadas. Não é um trabalho simples, demanda tempo e é custoso. Entretanto, não fazê-lo pode acarretar problemas muito maiores.

Independentemente da área em que você trabalhe ou queira trabalhar, seja backend, frontend, mobile ou segurança da informação, **certifique-se que está fazendo o possível para prezar a qualidade da sua aplicação**. Consegui te convencer?

Caso tenha se interessado pelo assunto, recomendo os seguintes conteúdos:

- Podcat Hipsters - Testes automatizados
- Fronteiras do Front-end | EP 06: Testes para Front End, faz sentido?
- Entenda a Pirâmide de Teste

Larissa Gabriela

Atuo como Instrutora da escola de programação na Alura, com foco em C#/.NET. Busco aprender cada vez mais e através dos meus conhecimentos auxiliar as pessoas a mergulhar no mundo da tecnologia. Técnica em telecomunicações pelo CEFET/RJ e graduanda em Física Médica pela UFRJ, estou no mundo da programação desde 2015. Nas horas vagas adoro jogar FPS e ver streams.

VER PLANOS

Veja outros artigos sobre
Programação

Quer mergulhar em tecnologia e aprendizagem?

Receba conteúdos, dicas, notícias, inovações e tendências sobre o mercado tech diretamente na sua caixa de entrada.

Email*

ENVIAR

Nossas redes e apps

VER PLANOS

Carreiras Alura

Para Empresas

Para Sua Escola

Política de Privacidade

Compromisso de Integridade

Termos de Uso

Documentos Institucionais

Status

Formações

Plataforma

Depoimentos

Instrutores(as)

Dev em <T>

Luri, a inteligência artificial da Alura

IA Conference 2025

Cursos imersivos

Certificações

Conteúdos

Alura Cases

Imersões

Artigos

Podcasts

Artigos de educação corporativa

Imersão Cloud Devops

Fale Conosco

Email e telefone

Perguntas frequentes

Novidades e Lançamentos

Email* ENVIAR