Colocando produtos prontos em produção

Você teve "aquela" ideia de produto digital ou está trabalhando num time com um produto muito maneiro. O time como um todo se dedica ao máximo, e você faz tudo que pode para garantir a qualidade e o sucesso: agilismo, testes (tdd, testes automatizados, testes de comportamento), vários ciclos, etc. Finalmente todas as funcionalidades parecem estar prontas. Todos trabalharam muito para finalmente colocar no "ar" o tão sonhado projeto.

Tudo pronto!

Pronto mesmo?

Ter todas as funcionalidade implementadas realmente significa que o sistema está pronto para produção, pronto para os usuários reais? Será que agora você pode tirar as suas merecidas férias e ir para um local onde "não haja conexão" e ficar com celular totalmente desligado?

É mais provável que nesse momento você está reunido com seu time, estabelecendo um escala de plantão para caso algum problema aconteça.

Muitos de nós desenvolvedores, não importa a experiência, trabalhamos muito orientados a *entregas*, a requisitos e a "builds verdes" (referências a sistemas de integração contínua que a cada alteração executa todos os testes automatizados do projeto e garante a qualidade do foi feito).

Ter testes automatizados e testes de comportamento, equipe de QA garantindo que tudo que se pediu está ali, não garante que o sistema está pronto para o ambiente de produção.

Coisas como grande volume de acesso, problemas na conexão ou na rede, serviços de terceiros que mudam sem aviso, ataques de hackers, podem e vão acontecer. E não é o fato da sua funcionalidade feita que vai garantir que esse tipo de coisa não vai tirar o seu produto do “ar”. Somente uma mentalidade treinada que irá ajudar.

É preciso estar pronto. E para estar pronto é preciso conhecer as características fundamentais de um sistema pronto, quais são as possíveis "ameaças" e problemas que irão enfrentar. Essa é exatamente a ideia aqui: ensinar os desafios e como vencê-los.

Mas antes de mais nada, preciso entender que tudo irá falhar. Tendo isso em mente o que iremos fazer é nos preparar da melhor forma possível para quando isso acontecer de

forma que não nos afeta, nos afete pouco e que no pior dos casos nós possamos voltar ao estado “ótimo” o mais rápido possível.

Nossa responsabilidade e cuidados com um sistema não terminam após o deploy. Cada nova funcionalidade no ar representar uma alteração no equilíbrio e com isso precisamos monitorar, analisar e medir. E isso, ao me ver, não pode ser simplesmente delegado para uma outra equipe diferente da que desenvolveu aquela aplicação.

Para finalizar, por mais que a maioria dos desenvolvedores sejam apaixonadas pela tecnologias, pelas linguagens e pelas "belas" soluções de arquitetura que criam, a coisa mais importante nesse mundo dito real é o **dinheiro**.

Produtos são feitos para reduzir custo, para dar lucro e para ser uma fonte de renda.

Ida e vindas no desenvolvimento são custos extras. Indisponibilidades das aplicações são custos (operacionais ou de oportunidade). Esse custo pode ser grande o suficiente que leve a falência.

O que é estar pronto para produção

Design de software que ensinado e praticado hoje em dia é terrivelmente incompleto. Ele fala somente do que os sistemas devem fazer. Ele não toca no assunto do que não se deve fazer. Eles não devem quebrar, cair, perder dados, violar privacidade, gerar prejuízo, destruir companhias ou espantar os clientes / usuários. [Release It - Michael Nygard, Introduction]

A maioria dos textos, livros, cursos, artigos, etc, foca apenas no aspecto das funcionalidades e da qualidade dentro do "tempo de desenvolvimento". Coisas como testes automatizados, ciclos de feedback com o cliente, deploys automatizados, ambientes automatizados de validação, etc.

Isso me parece muito desconectado da realidade, onde temos usuários fazendo fluxos loucos, digitando coisas que nunca imaginamos, flutuações de redes, alteração em serviços de terceiros não previstas, volumes de acesso muito acima do imaginado, etc.

Vivemos uma variação da piada "funciona na minha máquina", onde essa variação seria: funciona dentro dos parâmetros que consideramos no desenvolvimento.

Precisamos entender que as decisões de arquitetura e design precisam levar em conta a vida após o deploy.

Precisamos pensar e nos proteger das condições que não queremos que os nossos sistemas estejam. Pouco pensam o que fazer quando um banco de dados sair do ar; ou em uma estrutura de distribuição se a api de um terceiro sair do ar; como fazer se o usuário clicar em um botão várias vezes ao invés de uma só.

Não se trata de antecipação de problemas que nem sabemos se existem. E sim de preparar para a vida real!!!! Como disse antes, o banco irá cair! Aquela API vai cair! O link entre o Brasil e o Estados Unidos vai cair!

Já que gostamos tanto de copiar a engenharia, vamos trazer as boas práticas de preparo que nela existe, como fator de tolerância, resiliência, retorno de falha, coeficiente de segurança, etc.

As 5 Características de um sistemas prontos para produção

Na seção anterior eu te dei uma visão mais geral do que seria, no meu entender, o que é pronto para produção. Resumindo, é ter o sistema capaz de lidar com os problemas da vida real que com certeza irão aparecer.

Embora isso seja importante, creio que assim como eu, você esteja sentindo falta de uma definição mais concreta e possível de ser medida. Pois se não é possível medir, fica difícil melhorar.

Sendo assim, segue abaixo as 5 principais características e aspectos que devemos ter e analisar quando queremos fazer sistemas prontos para o ambiente de produção:

1. **Resiliência / Estabilidade**: capacidade de voltar de um erro ou falha de algum componente do qual depende. Exemplo: como o sistema se comporta quando o banco sai do ar; como o sistema se comporta se uma api sai do ar, etc.
2. **Performance**: capacidade de atender a demanda esperada. Isso pode ser considerado mais uma métrica do que um aspecto. A performance é algo que implica em muita coisa como custo de operação, falhas, quedas de entrega de funcionalidade, lentidão, etc.
3. **Escalabilidade**: é a capacidade em caso de sobre demanda, do sistema escalar para atender. Exemplo: quantas instâncias eu consigo colocar no ar; é possível ter mais de uma instância da aplicação no ar; ao criar mais instâncias do serviço não irei tirar outra coisa do ar.
4. **Deploy e ida para produção**: é medida de facilidade de colocar alguma coisa em produção para o usuário sem gerar indisponibilidade. Aqui também considero a capacidade de voltar sem grandes efeitos a versão se tivermos um bug naquela subida.
5. **Monitoração e Logs:** como o produto está sendo monitorado, e quais são informações como saúde, tempo de resposta, temos dele. Como os logs são fáceis de serem lidos e como podemos rastrear uma requisição para identificar algum bug ou problema.

Design e arquitetura para produção

Ter todos as funcionalidades cobertas de testes automatizados, ter ferramentas de integração contínua executando operações a cada alteração, ter testes de aceitação e, em alguns casos, o sinal verde da equipe de qualidade (QA), são coisas importantes mas não determinantes para garantir que o sistema irá funcionar perfeitamente em produção.

É preciso também pensar na sustentabilidade.

As arquiteturas e as decisões de design precisam levar em consideração os 5 aspectos citados acima e além disso responder a perguntas como por exemplo:

- *Seu sistema em caso de falha de alguma vai conseguir continuar funcionando ou voltar rapidamente?*
- *Se tiver um pico de acesso, vamos conseguir rapidamente nos recuperar e dar conta? O usuário não vai ficar com uma má impressão do produto.*
- *Temos condição de descobrir a causa de um problema em produção facilmente?*
- *Temos como auditar os sistemas?*
- *Temos como saber qual é a "saúde" da aplicação ?*

A arquitetura tem um papel muito importante para que um produto esteja pronto para produção e as decisões tomadas vão implicar toda a vida de um produto… para bem ou para o mal.

No final tudo se resume a dinheiro …

Nós desenvolvedores não gostamos muito de falar sobre custos. Não gostamos muito de pensar que nossa principal função é de gerar valor, seja reduzindo custos, ou entregando funcionalidades que vão gerar receitas.

Sendo assim, de uma forma simplória, podemos dizer que tudo que fazemos afetam os custos da empresa: operacional e de oportunidade.

Ao fazer algo que não está pronto para a vida em produção pode elevar o custo operacional a ponto que torne o produto e a empresa inviaveis. No pior dos casos, um sistema que não responda por algum motivo, pode representar a perda da oportunidade de

uma venda e perda de reputação e com isso, como já escutei uma vez, podemos estar deixando dinheiro na mesa.

A Amazon fez um estudo indicando quanto a conversão de visitas em compras mudam com o tempo para carregar a página. Isso significa, que cada segundo podem te custar muito dinheiro.

Tem histórias interessantes de empresas aéreas que faliram por conta de sistemas inoperantes. Sites de venda de ingressos que caíram por conta de acesso acima do esperado e com isso tem má reputação até hoje. Portais de compras que tiveram prejuízo por conta de erros de publicação e não ter uma monitoração.

No final tudo vai se resumir a dinheiro. Fazer coisa prontas para produção não algo interessante pois envolve técnicas, arquiteturas maneiras, soluções complexas, e uso de vários recursos… Fazer coisas prontas significa dinheiro: custo menores, mais possibilidade de ganhar e etc.

Sempre que estiver na dúvida entre fazer algo ou não, pense o quanto vão custar cada situação. E no final escolha a mais barata!

Ao longo das outras seções, verá que sempre vou voltar na questão do dinheiro. Pois ela é determinante.

Desfazendo mitos

A um bom tempo atrás participei de um evento da Amazon Web Services (AWS), nesse evento ganhamos um brinde que era 100 dólares para gastar com serviços AWS. Logo que cheguei em casa, fiquei super empolgado e criei uma conta e comecei a migrar alguns serviços que tinha espalhado para dentro da AWS.

Migrei umas aplicações que estavam no Heroku, outras que estavam numa VPS quem não vem ao caso nome, fiz backup de fotos no S3 (serviço de storage na Amazon), enfim, fiz um monte de coisa.

Não sei se ainda é assim, mas naquele momento, o primeiro ano de AWS era gratuito para alguns serviços (EC2 por exemplo que são instâncias virtuais). Mas mesmo assim, ao criar uma conta na Amazon, você precisa definir um cartão de crédito.

Bem, passados, 4 meses, começou aparecer umas cobranças no meu cartão. Como eram bem pequenas nem liguei. No sexto mês, foi quando levei um susto, apareceu para eu pagar um conta de 130 dólares.

O que aconteceu é que acabou meus créditos que tinha ganho, e a Amazon passou a cobrar pelos serviços que eu estava usando. Que, para surpresa de muitos, era apenas uma instância de EC2, um banco de dados e arquivos no S3.

Agora imagina que você é o CTO ou VP de engenharia ou apenas o dono de um empresa ou startup e resolveu contratar a AWS para suas necessidades de infra. Com certeza será algo mais barato que montar um datacenter… Mas não é tão barato a ponto de não se ligar!

Com essa história que gostaria de começar a desmitificar os mitos que envolvem colocar coisas prontas em produção. E o primeiro e maior de todos é que com provedores de nuvens (Cloud: AWS, Google, Microsoft, etc), o custo de máquinas é baixo, infra é barata.

Custo da infra, o mito das máquinas infinitas e cloud computing

Faça a experiência de deixar uma instância EC2 t2-small (no caso da Amazon) ligada um mês e veja quando será debitado em seu cartão de crédito. Agora imagina, por exemplo, que você precisa ter um backend com vários serviços para um aplicativo: não teremos só duas máquinas, banco de dados simples, etc.

A solução para resolver problemas de escalabilidade, por exemplo, nem sempre é colocar mais máquinas. A solução para ter alta disponibilidade, não é somente ter várias instâncias espalhadas em várias regiões do mundo: isso faz parte mas não é tudo.

Mais máquinas podem ao contrário de resolver criar mais problemas.

Trabalhei num projeto no qual começamos a ter problemas de lentidão e travamentos por conta de volume de acessos. A primeira coisa que todos no time fizeram foi …. Vamos subir mais servidores. A questão que a cada servidor novo, mais uma vai no banco de dados.

Como o banco de dados tem um número limitado de conexões, no final, ao invés de só ter um ambiente lento, acabamos ocupando todas as conexões com o banco e com isso, derrubando todo os serviços, pois todos iam nessa instância.

Aqui ficam duas lições: entender o que realmente está acontecendo e nem aumentar as máquinas não é solução.

Além disso a conta desse mês na AWS ficou bem cara e foi motivo de bastante discussões entre os diretores da empresa.

As máquinas dão conta do recado. Não precisamos de otimizações de código prematuras.

Todo estudante da ciência de computação sabe que algoritmos são mais poderosos que processadores. Gosto de contar que se continuássemos com os mesmo algoritmos de ordenação que tínhamos no início, para processar uma grande lista no mesmo tempo de algoritmos mais modernos, seria necessário um supercomputador.

Melhores de práticas de código irão superar o esforço da hardware.

Um amigo que trabalhou numa empresa grande de comunicação, fez uma excelente palestra no qual ele conta como foi a jornada que ele e seu time trilhou para melhorar um serviço deles. Resumindo bem a história, eles tinham um serviço que inicialmente tinha pouco acesso mais a medida do sucesso do produto, esses acessos foram aumentando e aumentando muito.

Claro que a primeira solução foi colocar mais máquinas.

Rapidamente eles saíram de coisa como 3 máquinas, para um cluster virtualizado de 30 máquinas, E mesmo assim não estavam dando conta.

Esse meu amigo resolveu então reescrever a aplicação usando uma nova linguagem e plataforma, e seguir as melhores práticas para resolver o problema que tinha.

O resultado foi que o serviço atende muito melhor, com um tempo de resposta muito menor, com uma estabilidade muito superior, e com apenas 3 máquinas.

Lembro da frase de Fernando Carolo, no momento que escrevo é diretor de infra e operações da Tradeshif: *"Máquinas nunca vão resolver problemas de códigos mal escritos".*

Os grandes provedores e empresas da internet nunca vão cair ou falhar…. Google, Amazon, Globo nunca caem e nunca ficam fora do ar.

Quanto antes entender melhor: ***Todos os serviços irão falhar . Sempre!!!***

Sempre que ouço algo parecido conta a história que um serviço no qual trabalho ficou fora junto com um monte de outros pois o datacenter da Amazon no Brasil “saiu do ar”.

Quando alguém, se a experiência, poderia imaginar que a amazon estaria fora; uma outra vez, nossos sistemas pararam de responder porque a Google desativou um DNS… E posso ficar escrevendo milhares de páginas com histórias minhas e de outros que conheço para dar exemplos.

É preciso entender as dependências e como lidar com elas. Novamente, não é porque você está usando uma solução mágica de infra que não terá que se preocupar. Quanto menos a gente sabe menos preparado estaremos.