E-BOOK

COMO ENSINAR UM ELEFANTE A DANÇAR

Evolução intencional de equipes, processos e aplicativos

Burr Sutter, *diretor da experiência do desenvolvedor*
Deon Ballard, *marketing de conteúdo*

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

br.redhat.com

Os aplicativos já não pertencem somente ao departamento de TI. Há uma expressão popular na área de TI que diz que todas as empresas atuais são também empresas de softwares. E a habilidade em fornecer novos serviços e funcionalidades aos clientes com rapidez é um dos principais diferenciais competitivos. A agilidade da TI é o segredo para que startups superem os enormes desafios e vençam as grandes batalhas.

Alguns anos atrás, em gerações tecnológicas mais antigas, os departamentos de TI eram setores internos que cuidavam da manutenção da infraestrutura e dos serviços corporativos. Algumas empresas tinham a necessidade de contratar serviços externos, principalmente serviços web. No entanto, essa era uma área ainda reduzida e restrita. A TI não era um departamento estratégico ou que gerava receita, mas um ambiente de suporte considerado como um centro de custo.

Um dos resultados desse ambiente dedicado à infraestrutura é que os desenvolvedores não tinham controle sobre a finalidade dos códigos criados. Os ciclos de lançamento eram longos, e as mudanças eram lentas. O desenvolvedor trabalhava em um projeto em que o código era enviado para fase de testes ou operações, e o lançamento era somente realizado alguns meses depois. Por conta desse longo tempo de provisionamento, os engenheiros se sentiam desmotivados a desenvolver novos códigos, pois nem sempre eles viam os códigos sendo utilizados na prática.

A transformação digital e as modificações culturais e tecnológicas, como o DevOps, estão proporcionando mudanças positivas e poderosas no processo de desenvolvimento, devolvendo o prazer e gratificação na criação de novos códigos. Os desenvolvedores podem criar um projeto e realmente vê-lo em execução. É uma mudança impactante, que transforma a criação de códigos em processos imediatos. Ver um aplicativo em execução fornece aos desenvolvedores um ciclo de feedback que os permitem aprimorar ou redesenhar os códigos sempre que necessário, aumentando as chances de sucesso desses projetos.

Nesse documento, usamos a metáfora do elefante para explicar o processo de transformação digital. Dessa forma, podemos dizer que o elefante representa onde a sua organização se encontra atualmente em relação à adoção de novas tecnologias e processos. A mudança de ambientes tradicionais para ambientes mais modernos que são baseados em microsserviços e DevOps, é dividida em etapas. Algumas organizações têm o privilégio de começar do zero. No entanto, para muitas empresas, o grande desafio é treinar esse pesado elefante e ensiná-lo a dançar como uma ágil bailarina.

O QUE QUEREMOS DIZER COM MUDANÇA?

A transformação digital é uma mudança estratégica para as empresas. Com ela, é possível adaptar os principais serviços de acordo com as demandas do setor ou conforme novas regulamentações surgem, bem como lançar atualizações à medida que surgem novas vulnerabilidades.

No entanto, não existe uma definição comum para a transformação digital, mas podemos dizer que ela envolve um processo de "mudança". Algumas vezes, esse termo é usado para designar novas arquiteturas (como microsserviços), novos processos (como DevOps) ou novas tecnologias (como os containers e interfaces de programação de aplicativos). Quando uma expressão pode significar diversas coisas, ela acaba perdendo o sentido e a eficácia. A transformação digital não é uma solução ou um serviço específico que você possa adquirir. Ela é algo que toda organização precisa definir para si mesma de forma exclusiva.

Não há um único padrão de arquitetura ou plataforma tecnológica que funcione perfeitamente em todos os ambientes. As organizações que alcançam o sucesso com a transformação digital são as que possuem um entendimento claro sobre os próprios objetivos e cultura. E eles são diferentes para cada uma delas. Por exemplo:

- O Walmart implantou código durante a Black Friday, quando 200 milhões de pessoas estavam on-line.[1]
- A Amazon implanta atualização de códigos a cada segundo (50 milhões por ano) em centenas de aplicativos e milhões de instâncias de cloud.[2]
- A Etsy realiza 60 implantações por dia com um aplicativo monolítico.[3]
- A Netflix realiza implantações centenas de vezes ao dia em uma arquitetura distribuída complexa. A empresa faz uma única alteração no código, indo do check-in à produção em 16 minutos.[4]

Cada uma dessas empresas trabalha com estrutura de equipes, tecnologias subjacentes, bases de códigos e arquiteturas totalmente diferentes umas às outras. O fato é que não há um único padrão ou uma tecnologia exclusiva usada entre elas que faça as operações funcionarem tão perfeitamente. Na verdade, todas essas empresas começaram com a avaliação de suas equipes, dívida técnica e estratégias corporativas. E então, elas seguiram em direção aos próprios objetivos, intencionalmente e com coerência. Dessa forma, essas empresas alcançaram os resultados almejados.

Esse é o processo ideal para ensinar um elefante a dançar. Seja qual for a situação atual da sua empresa, é possível modernizá-la de forma intencional e com uma estratégia clara, eliminando a dívida técnica, as falhas no projeto etc. Mas para que isso aconteça, é necessário ter clareza sobre os objetivos corporativos e o quão longe você está de alcançá-lo.

COMO IDENTIFICAR O ELEFANTE CORPORATIVO

Avaliar o cenário técnico atual e refinar a estratégia corporativa não são tarefas simples. É difícil ter esse tipo de perspectiva. Há uma parábola conhecida sobre seis homens cegos que encontram um elefante. Cada um deles toca em uma parte diferente do animal para tentar identificá-lo. Um dos homens toca na presa e acha que é uma lança. Outro toca um lado do corpo e acredita ser um muro. Outro toca as orelhas e pensa que é um leque. O interessante é que a parábola tem finais diferentes, dependendo da fonte. Em algumas versões, os homens não conseguem aceitar a opinião uns dos outros e acabam brigando. Em outras, um novo homem capaz de enxergar se aproxima e descreve a aparência do animal, unindo as características de cada percepção.

1 *O'Maidin, Cian. "Why Node.js Is Becoming The Go-To Technology In The Enterprise". NearForm, 10 de março de 2014, www.nearform.com/blog/node-js-becoming-go-technology-enterprise/. Acessado em 1º de setembro de 2017.*

2 *McKendrick, Joe. "How Amazon Handles a New Software Deployment Every Second". ZDNet, 24 de março de 2015, www.zdnet.com/article/how-amazon-handles-a-new-software-deployment-every-second/.*

3 *"The Great Microservices Vs Monolithic Apps Twitter Melee". High Scalability, 28 de julho de 2014, http://highscalability.com/blog/2014/7/28/the-great-microservices-vs-monolithic-apps-twitter-melee.html.*

4 *Bukoski, Ed, et al. "How We Build Code at Netflix". The Netflix Tech Blog, 9 de março de 2016, https://medium.com/netflix-techblog/how-we-build-code-at-netflix-c5d9bd727f15.*

Os finais diferentes talvez sejam a parte mais realista da parábola. A moral da história é que todos nós temos diferentes perspectivas e conjuntos limitados de informações que, frequentemente, tomamos como base para criar nossas hipóteses. O resultado dessas diferenças está associado à comunicação e aos relacionamentos inerentes ao trabalho em equipe: alguns indivíduos vão além de suas próprias perspectivas, outros ficam presos a elas. O que enxergamos depende de quem somos, de onde estamos, do que esperamos, do que sabemos e do que *não* sabemos.

Isso se torna ainda mais complexo se levarmos em consideração que a maioria das organizações existentes tem mais de um elefante. A Netflix é um unicórnio, mas não somente por causa de sua arquitetura de microsserviços avançada e ambientes de implantação e teste baseados em containers. A empresa conseguiu desenvolver os processos da equipe e o stack de tecnologia de acordo com a estratégia corporativa, sem dívida técnica. Ela era uma startup.

Qualquer organização com aplicativos e equipes legadas tem mais do que um elefante corporativo. Isso inclui:

- Estruturas de equipe e padrões de comunicação atuais.
- Processos de desenvolvimento, teste, compilação e lançamento.
- Dívida técnica e aplicativos comuns existentes.
- Objetivos e visões estratégicas diferentes entre os departamentos.

É provável que cada stakeholder veja esses problemas com perspectivas diferentes.

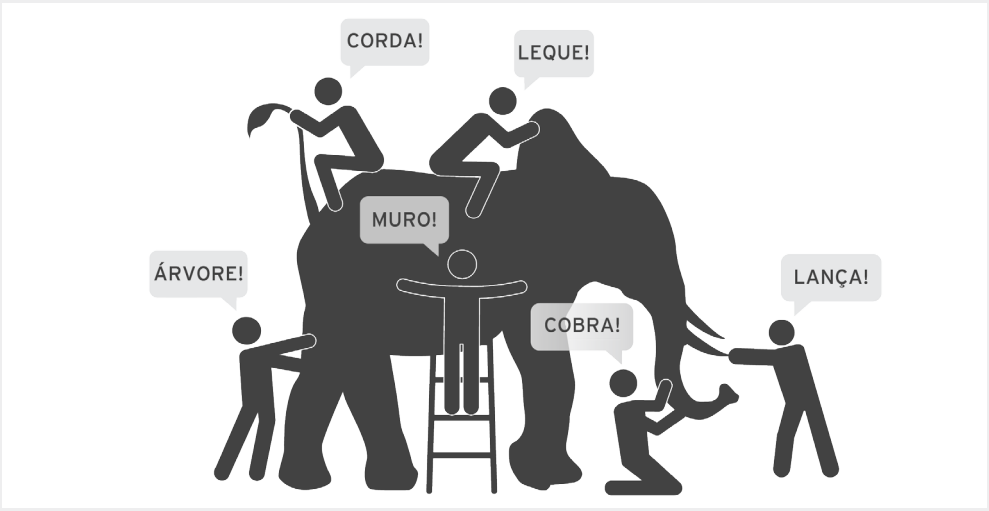

Figura 1. Alegoria de seis homens cegos e um elefante

UMA VISÃO DARWINISTA DA TRANSFORMAÇÃO DIGITAL

Existe uma tendência em pensar na TI ou nas iniciativas de transformação digital como tarefas binárias, ou seja, só é possível realizar uma tarefa de cada vez. Isso talvez não funcione por dois motivos. Primeiro, às vezes, é possível executar mais do que um processo ou escolher uma solução híbrida. Segundo, muitas vezes não se trata de mudar um único fator, já que cada mudança requer alterações fundamentais ou dependem de modificações na cultura e nos processos.

Nada acontece repentinamente.

Devemos considerar a transformação digital como uma sequência contínua, com etapas diferentes ao longo do processo que possibilitam chegar ao próximo estágio da evolução.

Figura 2. As etapas do Darwinismo digital

DevOps e mudança nos processos

A base da evolução digital é o DevOps. Assim como a estratégia corporativa, os aplicativos são um reflexo das equipes e da comunicação entre elas. DevOps, ou mudanças similares nos processos, faz com que mais stakeholders participem das discussões de desenvolvimento e, assim, oferecer insights mais amplos sobre como a equipe de operações mantém o software e a infraestrutura e como clientes e parceiros realmente usam esses aplicativos. Ele cria um ciclo de feedback mais fechado entre as equipes, com linhas abertas de comunicação. E essa comunicação aberta é a base para qualquer outra etapa da evolução.

Infraestrutura de autosserviço

Essa etapa consiste em uma mudança focada na tecnologia. Ela introduz eficiências que geralmente estão associadas a plataformas de tecnologias modernas. Com containers e catálogos de autosserviço, os grupos de desenvolvedores, testes e operações podem acionar ambientes consistentes com muita rapidez. Em algumas organizações, o tempo de provisionamento de novas instâncias é reduzido de dias para minutos. Os profissionais de TI não precisam esperar dias por um recurso de computação.

OS MICROSSERVIÇOS SÃO UMA ETAPA NECESSÁRIA?

A última etapa da evolução digital é a adoção de microsserviços devido à complexidade para mantê-los. Mas eles precisam ser o estágio final da evolução da sua organização?

Na verdade, não.

Se os outros aspectos da evolução estão funcionando, você ainda pode usar a arquitetura monolítica para fazer lançamentos semanais, além de utilizar técnicas avançadas de implantação, CI/CD e infraestrutura distribuída e escalável.

Escolha os microsserviços apenas quando as equipes forem grandes, e quando precisar fazer lançamentos mais de uma vez por semana ou em programações diferentes. Equipes ou bases de códigos pequenas não precisam ser divididas em bases de códigos de microsserviços.

Automação e orquestração

Adotar a automação exige uma mudança em dois ângulos: o tecnológico, com plataformas avançadas de implantação como o Red Hat® Ansible ou Puppet, mas há também a necessidade de realizar mudanças nos processos. Várias organizações adotam processos rigorosos de mudança e gerenciamento de riscos. Mas para aproveitar os benefícios das novas tecnologias, é preciso adaptar os processos com o uso de metodologias mais ágeis.

Pipelines de integração e entrega contínuas (CI/CD)

Com a entrega contínua, é possível fazer mudanças no software com rapidez e iteratividade. A ideia de um pipeline é de que haja processos e tecnologias em execução capazes de reduzir o risco de códigos com baixa qualidade (ou quebrados) chegarem até a fase de implantação. Essa etapa mostra a maturidade dos estágios anteriores: DevOps e comunicação aberta entre as equipes, processos de testes e compilações em execução, além de teste e implantação automatizados. Quando todos esses estágios são sólidos, é possível usar os códigos com rapidez. Esse é o pipeline.

Caminhos avançados de implantação

Uma vez que os processos e a infraestrutura estiverem voltados para implantações rápidas, será possível usar os sistemas para mitigar os riscos das atualizações, avaliar a eficiência da funcionalidade e realizar testes reais para gerar novas ideias. Isso inclui ter ambientes separados e balanceamento de carga entre eles durante as implantações (chamadas de "blue-green deployments"). Nesse processo, são usados dois ambientes diferentes para testar a interação do usuário (teste A/B) ou fazer atualizações para uma pequena porcentagem de usuários e, de modo seguro, aumentar esse número gradativamente (conhecido como "canary deployments").

Microsserviços ou sistemas distribuídos

Um microsserviço é uma aplicação pequena que realiza uma única função separada. A arquitetura geral de aplicativos pode precisar realizar dezenas ou centenas de funções diferentes, cada uma definida e orquestrada em um microsserviço. A arquitetura de microsserviços, como qualquer outra de computação distribuída, é complexa e simples ao mesmo tempo. Serviços individuais são muito mais simples e fáceis de manter, adicionar e descontinuar. Já a arquitetura geral é mais complexa. Quando criado corretamente, um "projeto baseado em microsserviços é o melhor resultado de todo o conhecimento adquirido sobre um bom design de aplicativos".[5] Com essa arquitetura altamente distribuída, você escala com mais facilidade, introduz novos serviços e atualizações com mais simplicidade e reduz o risco de falhas no sistema. Por conta dessa elasticidade na arquitetura, os microsserviços são muito associados a empresas inovadoras como a Netflix e a Google.

"A ENTREGA CONTÍNUA NÃO É PARA TODOS": LEI DE CONWAY, DEVOPS E CULTURA

Durante a apresentação geral da DevNation 2016, Rachel Laycock afirmou que "operacionalizar o software é bastante difícil".[6] Ela compartilhou a história de uma enorme empresa de serviços financeiros que falhou ao implementar a entrega contínua. A organização demorava semanas para lançar atualizações, até mesmo as mais críticas. A empresa acreditava que a solução seria adotar a entrega contínua. Depois de seis meses tentando alterar os processos, Rachel voltou atrás e contou a um dos vice-presidentes que essa não seria a solução.

5 *Cotton, Ben. "From Monolith to Microservices". 3 de janeiro de 2017, https://www.nextplatform.com/2017/01/03/from-monolith-to-microservices/.*

6 *Laycock, Rachel. "Continuous Delivery". Apresentação da tarde. Red Hat Summit - DevNation 2016, 1º de julho de 2016, São Francisco, Califórnia. https://www.youtube.com/watch?v=y87SUSOfgTY*

"A CI/CD garante entrega rápida e contínua. Já o DevOps oferece suporte ao que você fornece."[9]

– BURR SUTTER

Parte do problema era a tecnologia e a arquitetura. A empresa de serviços financeiros tinha uma base de códigos com mais de 70 milhões de linhas, além de uma arquitetura bagunçada e mal-estruturada. No entanto, o software era a parte fácil, pois era um problema óbvio. O verdadeiro elefante dessa organização era a incapacidade de alterar o comportamento de seus vários departamentos. Isso impedia todos os esforços da empresa em adotar uma mudança.

Cultura em primeiro lugar

É importante notar que a evolução darwinista do software não é apenas uma mudança na tecnologia. Ela alterna entre mudanças em processos, equipes e infraestrutura, e as modificações culturais são infinitamente mais importantes. Rachel concluiu sua apresentação dizendo:[7]

"É possível criar o diagrama de arquitetura mais perfeito e desejado. Depois de envolver as equipes e os processos, é preciso criar um ambiente cultural que ofereça suporte à entrega contínua e à disciplina no uso da arquitetura. Porque uma mudança nos processos ou na estrutura não é duradoura. As pessoas não são como os computadores que seguem regras. Portanto, o verdadeiro elefante é a Lei de Conway. O que isso significa? As estruturas legadas da organização *sempre* destruirão a sua bela arquitetura."

Para que o software e a tecnologia evoluam, o segredo é pensar na evolução como um processo natural do ambiente. Em uma empresa, isso é a cultura. Para garantir isso, as mudanças necessárias podem ter o suporte do gerenciamento, mas não podem ser impostas por ele. As pessoas precisam querer mudar. É uma questão de livre-arbítrio, e não imposição.

O Gartner tem um dado que explica isso melhor: "90% das organizações que tentam usar o DevOps sem mudar especificamente a cultura falham."[8]

É fácil mudar a infraestrutura e a arquitetura de aplicativos. Mas para fazer isso efetivamente, primeiro é necessário realizar uma mudança cultural.

A Lei de Conway diz: "Qualquer organização que criar um sistema, inevitavelmente, produzirá um projeto com uma estrutura igual a da comunicação da organização." Isso pode ser interpretado de duas formas:

- Mudanças na arquitetura ou infraestrutura não terão qualquer efeito, a menos que você também modifique a estrutura de comunicação.
- Alterações na estrutura de comunicação resultarão em melhores processos e infraestrutura (independentemente da infraestrutura atual).

A Lei de Conway diz: "Qualquer organização que criar um sistema, inevitavelmente, produzirá um projeto com uma estrutura que será igual à da comunicação da organização."[10]

DevOps como primeira etapa

A metodologia ágil é uma abordagem de design de software que tenta unir todas as partes envolvidas em um grupo mais coeso. Isso inclui QA, gerenciamento de solução, desenvolvedores e até mesmo documentação. A ideia é esclarecer os objetivos com pequenas iterações que têm como foco tarefas específicas expressas como metas do usuário (chamadas de histórias). Isso derrubou o tradicional modelo cascata do desenvolvimento de software, em que o processo passava de uma equipe para outra.

7 *Laycock, Rachel. "Continuous Delivery". Apresentação da tarde. Red Hat Summit - DevNation 2016, 1º de julho de 2016, São Francisco, Califórnia. https://www.youtube.com/watch?v=y87SUSOfgTY.*

8 *"Gartner Highlights Five Key Steps to Delivering an Agile I&O Culture". 20 de abril de 2015, www.gartner.com/newsroom/id/3032517.*

9 *DevNation Federal, 8 de junho de 2017, Washington, DC, https://www.youtube.com/watch?v=tQOo2qaUc6w&t=1s*

10 *Conway, Melvin E. (abril de 1968), "How do Committees Invent?", Datamation*

"Os designs das equipes são o primeiro esboço da sua arquitetura."

– MICHAEL NYGARD, RELEASE IT!

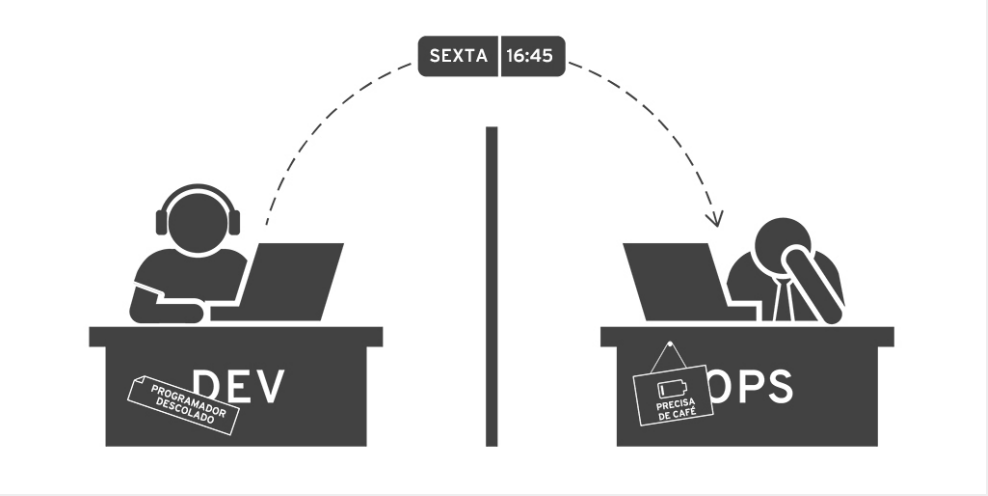

Figura 3. O muro de divisão antes do DevOps

No entanto, a metodologia ágil ainda lida somente com metade do verdadeiro ciclo de vida de um aplicativo. Depois de desenvolvido, o aplicativo é transferido para a equipe de operações. Ela fica responsável pela implantação e manutenção, que acontece geralmente em um período de manutenção no fim de semana.

O problema é que essa equipe nem sempre sabe o propósito do aplicativo, o que pode gerar uma implantação menos eficiente. Os desenvolvedores podem não ter conhecimento sobre o verdadeiro ambiente operacional e criar um aplicativo que não tenha um bom desempenho no ambiente de produção. Então, para mitigar o risco de alterações, muitas organizações instituem um processo oneroso de gerenciamento de mudanças para tentar explicar e justificar qualquer alteração.

O DevOps é uma mudança cultural que tenta acabar com a separação entre os desenvolvedores, equipe de operações e stakeholders. A separação entre esses grupos é real, mas artificial. Uma equipe pode ser formada com profissionais de diversas funções. O DevOps tenta redefinir a equipe para incluir todos os grupos envolvidos no ciclo de vida de um aplicativo e criar uma abertura na comunicação entre esses grupos.

Os efeitos da mudança na cultura são muito mais profundos que DevOps, metodologias ágeis e outros modelos. É o compromisso de realmente colocar todos no mesmo time. Ao mudar os padrões de comunicação, você modifica seus resultados.

Com algumas etapas bem simples, grandes mudanças podem surgir. A cultura é a base para todas as modificações tecnológicas e de processos. Se estiver com dificuldades para criar uma cultura de DevOps, experimente o seguinte:

- Faça os desenvolvedores passarem um final de semana com a equipe de operações, assistindo a implantações de produção e aprendendo os processos.
- Verifique qual é a quantidade necessária de etapas ou tickets de serviço para que um desenvolvedor solicite um novo sistema virtual.

Ver como as outras equipes funcionam na prática pode ser um ótimo jeito de incentivá-las a mudar os processos e abrir a comunicação.

DESTAQUES DO RELATÓRIO "PUPPET'S STATE OF DEVOPS"

- Tempo de provisionamento 2.555 vezes mais rápido.
- 200 vezes mais implantações
- Recuperação de falhas 24 vezes mais rápida
- Taxa de falhas na mudança 3 vezes menor
- 22% menos tempo gasto em retrabalho.

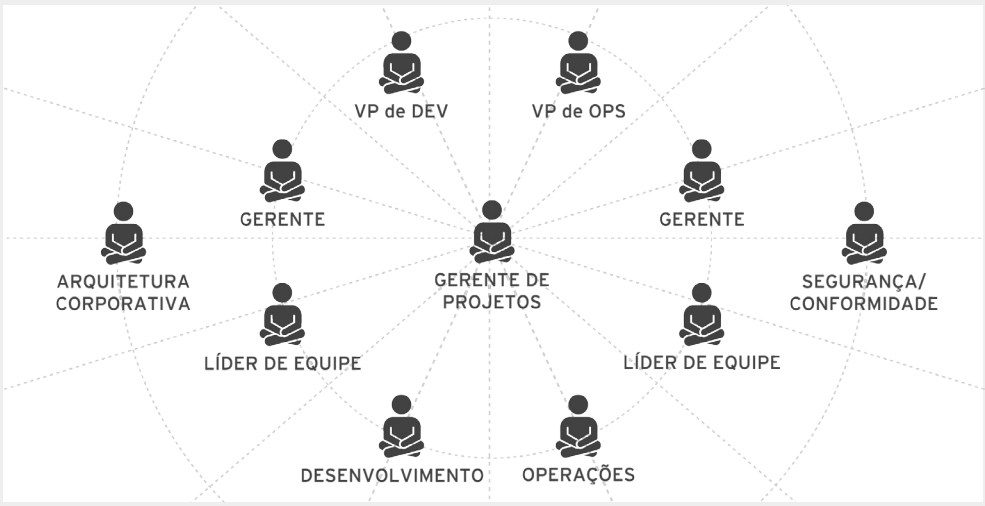

Figura 4. Visão completa dos caminhos de comunicação das equipes

O relatório "Puppet's State of DevOps" mostra a eficácia alcançada ao mudar a estrutura e a comunicação da equipe.[11] O estudo revelou que as equipes de DevOps alcançam:

- Tempo de provisionamento 2.555 vezes mais rápido.
- 200 vezes mais implantações.
- Recuperação de falhas 24 vezes mais rápida.
- Taxa de falhas na mudança 3 vezes menor.
- 22% menos tempo gasto em retrabalho.

A rapidez é uma das principais vantagens do DevOps. Ao analisarmos todo o processo de lançamento, fica claro a necessidade de ter todas essas diferentes equipes envolvidas para a disponibilização de um aplicativo. O nível de esforço para o lançamento desse aplicativo irá depender da transparência e clareza dos processos, infraestrutura e base de códigos.

11 Kim, Gene, et al. "State of DevOps Report". Puppet, 2016, https://puppet.com/resources/whitepaper/2016-state-of-devops-report.

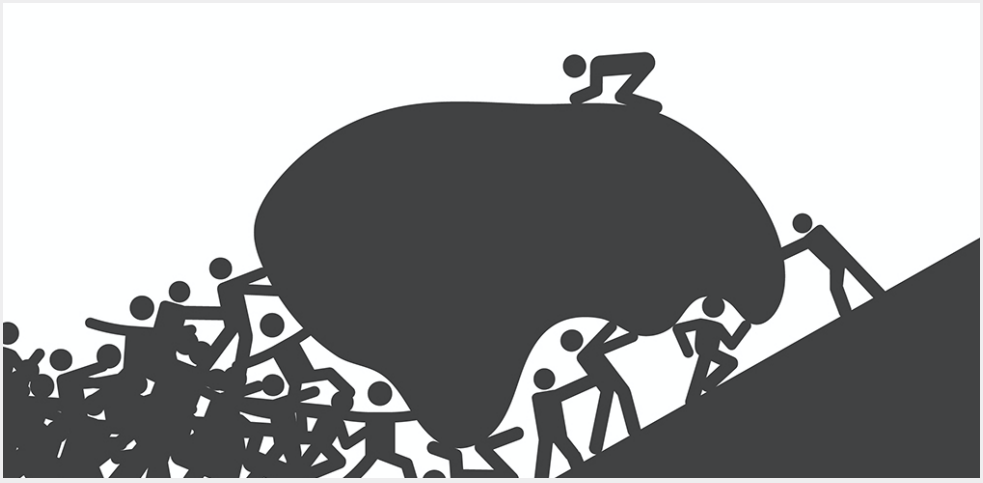

Figura 5. Esforço da equipe no lançamento de um único aplicativo

É fácil gerenciar esses problemas quando os lançamentos não são tão frequentes. Porém, quando o software impulsiona os negócios, todo o processo de lançamento precisa acontecer diversas vezes por ano. (Em empresas altamente inovadoras, como a Amazon, isso acontece centenas de vezes por dia.) Nesse caso, o enorme esforço de diversas equipes para colocar o aplicativo em produção precisa ser constantemente repetido.

Figura 6. Esforço em um único lançamento, com outros pela frente

O DevOps altera a abordagem: de um empenho coletivo e de urgência para uma iteração mais tranquila e sustentável entre o planejamento, desenvolvimento, testes e implantação. É por isso que as equipes de DevOps geram um grande aumento na produtividade, pois todo o ciclo de vida é reproduzível.

Seja sua arquitetura de aplicativos monolítica mais sofisticada ou microsserviços distribuídos, mudar a estrutura da equipe e a comunicação é crucial. A comunicação precisa fluir de forma natural, tanto antes do planejamento de um aplicativo quanto após a implantação dele. Uma equipe pode ser separada facilmente em um silo por serviço, a menos que você tenha criado e habilitado uma cultura que ofereça suporte para comunicação aberta e feedback.[12]

COMO PROJETAR UMA ARQUITETURA DE APLICATIVOS: FOCANDO EM MICROSSERVIÇOS

Uma nova arquitetura de aplicativos é a etapa final da evolução digital. No entanto, esse é um dos elefantes mais visíveis e um dos mais fáceis de identificar na organização. Vale a pena analisar a arquitetura, mesmo se a mudança nas plataformas e processos for prioridade.

Força do design, dívida técnica e estratégia

Muitas organizações não conseguem trabalhar em torno dos próprios aplicativos existentes. Isso é um grande obstáculo para a mudança. Por isso, muitas iniciativas de transformação digital adotam uma abordagem de "remoção e substituição". Às vezes, parece muito mais fácil começar do zero.

No entanto, o problema é que o resultado do design é a dívida técnica. Remover um aplicativo sem saber exatamente o que o substituirá significa que, eventualmente, a mesma arquitetura impenetrável ressurgirá.

Rachel Laycock explicou: "Esperança não é um método de design. Tudo precisa ser intencional."[13]

Seja o que for que você estiver desenvolvendo, esta será sua dívida técnica. Frequentemente, as equipes começam a desenvolver aplicativos inovadores sem muita clareza sobre o design. Isso não é necessariamente ruim. A descrição de Martin Fowler sobre a força do design e a dívida técnica ressalta que começar sem uma ideia clara de design muitas vezes proporciona uma inovação inicial mais rápida.[14] No entanto, chega um momento em que o "bom design" segue uma trajetória fixa, e "nenhum design" se iguala.

---

12 Cotton, Ben. "From Monolith to Microservices". 3 de janeiro de 2017, https://www.nextplatform.com/2017/01/03/from-monolith-to-microservices/.

13 Laycock, Rachel. "Continuous Delivery". Apresentação da tarde. Red Hat Summit - DevNation 2016, 1º de julho de 2016, São Francisco, Califórnia. https://www.youtube.com/watch?v=y87SUSOfgTY

14 Fowler, Martin. "Design Stamina Hypothesis". 20 de julho de 2007, https://martinfowler.com/bliki/DesignStaminaHypothesis.html.

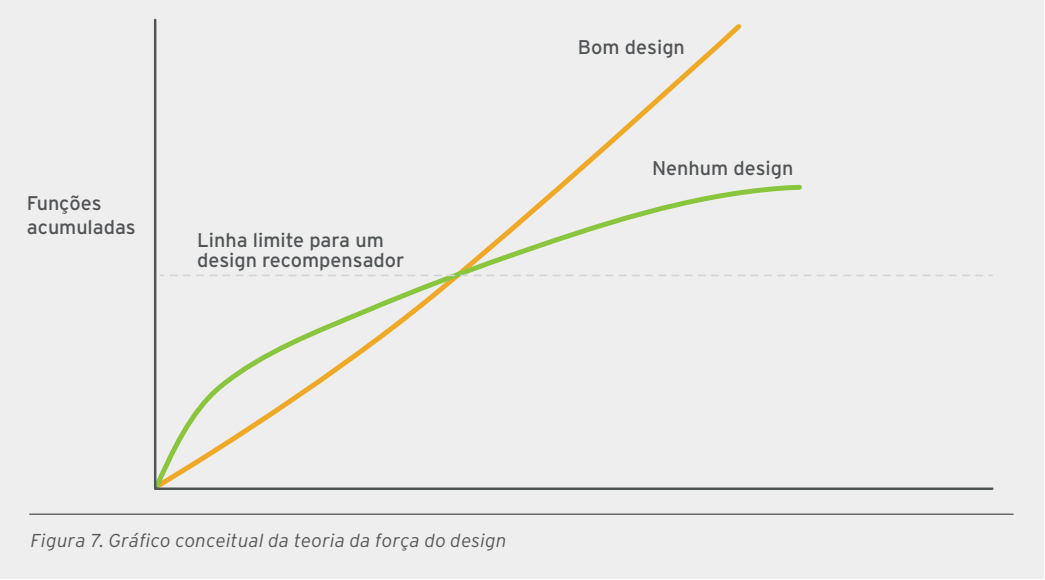

Figura 7. Gráfico conceitual da teoria da força do design

Antes de começar a planejar a arquitetura, é necessário entender as suas prioridades e objetivos estratégicos. Rob Zuber escreveu na Information Week:[15]

"Se você não tiver uma visão clara sobre os seus negócios, soluções ou seu lugar no stack, separar os serviços em grupos sem saber como eles serão usados não trará bons resultados. [...] Esse é um ótimo momento para pensar sobre a arquitetura. No entanto, é importante fazer isso de um jeito gradativo e que permita testar e validar as ideias, em vez de mudar de direção e, de repente, criar uma proposta totalmente diferente. Dessa forma, haverá um grande risco de terminar um longo projeto com diversas falhas. Além de ter que explicar ao vice-presidente de engenharia da empresa que, apesar de ter aprendido muito durante esse processo, a equipe decidiu não lançar o aplicativo. Com certeza, esse não é o objetivo. Você deseja fornecer valor e impacto positivo aos clientes e negócios por meio de processos planejados e bem pensados."

Em um artigo recente da IDC, Stephen Elliot escreveu que antes de definir uma arquitetura, é necessário identificar quem é seu cliente ou usuário final, o que ele valoriza, quais são os resultados desejados e como avaliar seu sucesso.[16]

Em outras palavras, ao iniciar um projeto, o primeiro passo é identificar as metas estratégicas e, em seguida, projetar uma arquitetura que ofereça suporte a elas, em vez de focar em como alcançar os objetivos ou qual arquitetura deverá ser usada. Dessa forma, o aplicativo é colocado como prioridade.

Definição de microsserviços e monolíticos

Atualmente, há três arquiteturas de aplicativo conhecidas que se baseiam na relação entre os serviços: monolíticos (fortemente acoplados), microsserviços (desacoplados) e arquiteturas orientadas a serviços (levemente acopladas), sendo esta última não muito utilizada atualmente.

15 *Zuber, Rob. "Transitioning to Microservices: The Story of Two Monoliths". InformationWeek, 25 de maio de 2017, https://www.informationweek.com/devops/transitioning-to-microservices-the-story-of-two-monoliths-/a/d-id/1328972.*

16 *Elliot, Stephen. "Enabling DevOps with Modern Application Architecture". Artigo da IDC, dezembro de 2016.*

PLANEJAMENTO INTENCIONAL

- Quem são seus clientes ou usuários?
- O que eles estão tentando fazer?
- Que infraestrutura você tem?
- Qual é o ciclo de vida dessa infraestrutura?
- Que serviços ou funcionalidades são necessários em um único fluxo de trabalho?
- Qual é o ciclo de vida desse fluxo de trabalho?
- Qual é o seu caminho de implantação e com que frequência você realiza esse processo?
- Quais recursos corporativos isso afeta?

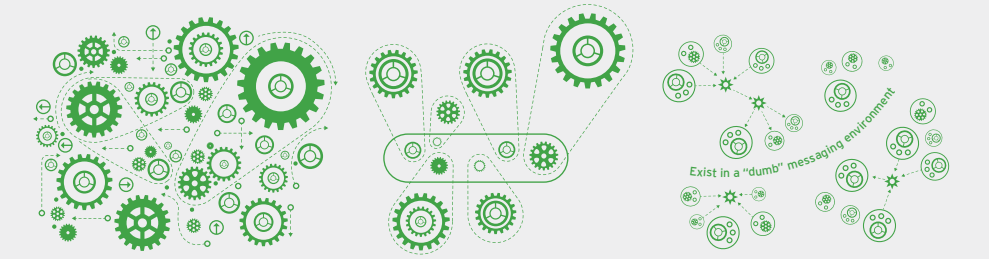

Figura 8. Acoplamento monolítico, arquitetura orientada a serviços e microsserviços

Em termos simples, um monolítico é um único stack de aplicativos que contém todas as funcionalidades dele. Ele é fortemente acoplado, tanto na interação entre os serviços como na forma que é desenvolvido e fornecido. Atualizar ou escalar um único aspecto do aplicativo monolítico o afetará inteiramente, além de impactar a infraestrutura subjacente.

Escala dinâmica e failover são os possíveis problemas do uso de monolíticos. Esses problemas costumam ser resolvidos com projetos simples de escalabilidade, como a escala horizontal (que duplica a função em um cluster) e a escala vertical (que espelha as instâncias e amplia o hardware). Outro problema de escalabilidade, que é menos considerado, é em relação às equipes de desenvolvimento e operações. Vamos supor que seja necessária uma equipe de 50 pessoas para lançar um aplicativo monolítico a cada seis ou nove meses. Talvez seja possível aprimorar a escala com cinco aplicativos menores que conta com cinco equipes menores, lançando atualizações individuais em poucas semanas.

A arquitetura monolítica de aplicativos provavelmente é a mais antiga, devido à sua simplicidade inicial e relações mais claras entre serviços e interdependências. Ela também reflete uma infraestrutura de TI limitada e comum, com processos de desenvolvimento e lançamento mais rígidos.

Como os monolíticos são um tipo mais antigo de arquitetura, eles são frequentemente associados a aplicativos legados. Em comparação, as arquiteturas mais modernas tentam separar os serviços por funcionalidade ou recursos corporativos para fornecer mais flexibilidade. Isso é comum principalmente nas interfaces voltadas para o cliente, como APIs e aplicativos da web ou mobile, que costumam ser menores e exigir atualizações mais frequentes para atender às expectativas dele.

Uma das definições mais atuais de arquitetura distribuída são os microsserviços. Há algumas semelhanças com outros designs modulares, como arquitetura orientada a serviços (SOA). No entanto, em vez de oferecer um acoplamento leve entre os serviços, os microsserviços oferecem a independência entre eles. A definição de um único serviço costuma ser mais clara. Assim, é possível adicionar mais serviços, realizar upgrade ou removê-los da arquitetura maior com facilidade. Isso garante benefícios para a escalabilidade dinâmica e tolerância a falhas. Logo, os

serviços individuais podem executar failover sem afetar os outros e ser escalados conforme necessário, sem precisar de uma infraestrutura pesada. Ben Cotton afirmou: "Um projeto baseado em microsserviços é o melhor resultado de todo o conhecimento adquirido sobre um bom design de aplicativos."[17]

Por conta da sua fluidez, podemos dizer que a arquitetura de microsserviços está associada a tecnologias dinâmicas, como containers e clouds, que permitem iniciar e remover instâncias individuais com facilidade e de forma programática.

A computação distribuída, que é mais imediata, garante benefícios diretos para a organização e as equipes. Os impactos no trabalho incluem:

- Tolerância a falhas aprimorada e interrupções de serviço reduzidas.
- Protocolos simples como JSON/REST e HTTP/OAuth para integração facilitada.
- Serviços com suporte para múltiplas linguagens, que proporciona flexibilidade ao desenvolvedor.
- Rápido time to market para aplicativos e funcionalidades.
- Recuperação e compartilhamento de dados simplificados, sem precisar de barramentos de mensagens mais pesados ou conversão.[18]

O site Coding the Architecture afirma que uma arquitetura monolítica se compara à de microsserviços quando o tempo de execução do próprio aplicativo é monolítico e estático.[19] Vários aplicativos têm muitos pacotes ou módulos que podem ter sido criados ou implantados de maneira independente. No entanto, o próprio aplicativo interage como uma única entidade.

A questão é qual estilo de arquitetura é o mais adequado. É comum considerarmos que a tecnologia mais recente é a melhor. No entanto, é importante avaliar qual delas alcança os resultados corporativos desejados. Por exemplo, em uma conversa no Twitter, os engenheiros da Etsy e da Netflix discutiram sobre a necessidade dos microsserviços na entrega contínua e na autonomia do desenvolvedor. Representantes da Etsy mencionaram que a empresa conta com pequenas equipes de desenvolvimento ágil baseadas em funcionalidade, com implantações muito rápidas (cerca de 60 por dia) e sendo executadas em um aplicativo monolítico. Os engenheiros encontraram um sistema que funciona bem para a empresa e para sua cultura.

A arquitetura e as tecnologias mudam e evoluem. "As práticas recomendadas de ontem não serão os padrões no futuro", afirmou Rachel em sua apresentação na DevNation. "Os microsserviços são uma opção. São uma resposta. Mas não a solução. Na verdade, são uma possível solução."[20]

17 *Cotton, Ben. "From Monolith to Microservices". 3 de janeiro de 2017, https://www.nextplatform.com/2017/01/03/from-monolith-to-microservices/.*

18 ***Lambert, Natal**ie. "Micro Services: Breaking down Software Monoliths". NetworkWorld, 22 de novembro de 2017, https://www.networkworld.com/article/3143971/application-development/micro-services-breaking-down-software-monoliths.html.*

19 *Annett, Robert. "What Is a Monolith?" Coding the Architecture, 19 de novembro de 2014, http://www.codingthearchitecture.com/2014/11/19/what_is_a_monolith.html.*

20 *Laycock, Rachel. "Continuous Delivery". Apresentação da tarde. Red Hat Summit - DevNation 2016, 1º de julho de 2016, São Francisco, Califórnia. https://www.youtube.com/watch?v=y87SUSOfgTY*

Substituir, ou não, a arquitetura monolítica por microsserviços não deve ser a dúvida de uma organização. Na verdade, ela precisa se questionar *qual é o seu objetivo estratégico e o que é necessário fazer para chegar lá*. É importante entender a estrutura e cultura de comunicação, os recursos corporativos e os fluxos de trabalho necessários para saber qual a influência terão sobre como os serviços serão acoplados, qual o ciclo de vida deles e, por fim, a arquitetura de aplicativos.

TABELA 1. COMPARAÇÃO ENTRE A ARQUITETURA MONOLÍTICA E DE MICROSSERVIÇOS

	MONOLÍTICOS	MICROSSERVIÇOS
Desenvolvimento	• Desenvolvimento inicial mais rápido • Dificuldade em mudar ou adicionar recursos	• Design inicial é importante • Facilidade em adicionar ou alterar serviços
Fluxos de trabalho do aplicativo	• Facilidade de encaixar aplicativos nos fluxos de trabalho • Implementação de funcionalidade (por exemplo, autenticação ou monitoramento) em um único local	• Dificuldade em atribuir serviços aos fluxos de trabalho • Interdependências ou requisitos entre os serviços não são tão claros
Treinamento e manutenção	• Arquitetura simples • Requisitos rígidos de desenvolvimento para ambientes e linguagens	• Arquitetura flexível com mais complexidade no design • Serviços com suporte a múltiplas linguagens com APIs padronizadas ou sistema de mensageria para conexão
Escala	• Dificuldade de escala; dependente da infraestrutura do hardware • Escala total de aplicativos para um único pico de serviço	• Facilidade em escalar serviços individuais sem impactar a arquitetura geral • Uso de infraestrutura definida por software (containers, cloud) para obter responsividade dinâmica
Atualizações, failover e tempo de inatividade	• Todos os serviços são fortemente acoplados • Os serviços devem ser atualizados juntos, o controle de versão é acoplado • Risco de falha no sistema se houver falha em um serviço individual	• Serviços não são acoplados • Serviços podem ser adicionados ou atualizados de forma independente • Risco de falha limitado a um pequeno número de serviços
Automação	• Automação é amplamente desnecessária	• Requer automação e orquestração

Falácias da computação distribuída

Um engenheiro ironizou dizendo que os microsserviços transformam qualquer interrupção em um grande mistério.[21] Esse é um resumo curioso do maior problema da computação distribuída: a dispersão da complexidade.

21 @HonestStatusPage. Twitter, 7 de outubro de 2015, https://twitter.com/honest_update/status/651897353889259520?lang=en.

Aumento nos gastos e custos de transação

Para obter uma verdadeira computação distribuída, são necessárias mudanças na infraestrutura que podem gerar custos muito altos. Há também gastos indiretos com treinamentos para aprimorar ou obter novas habilidades, alteração na estrutura de equipes e migração de sistemas. Esses gastos podem gerar economias futuras em indicadores, como time to market e tempo de inatividade reduzido (se a nova arquitetura for implementada corretamente). No entanto, esses benefícios não são imediatos.

Aumento na complexidade

Em vez de ter apenas um ponto de falha, como acontece com os monolíticos, a arquitetura de microsserviços conta com centenas de possibilidades diferentes. Em ambas as arquiteturas, acompanhar essa falha pode ser difícil porque talvez a causa raiz não seja tão óbvia. No entanto, os microsserviços aumentam a complexidade, porque as interdependências entre os serviços são ainda menos visíveis.

Até mesmo os ciclos de lançamento mais rápidos e as estruturas de equipe mais ágeis acrescentam complexidade. Para os microsserviços, a comunicação efetiva é crítica. Todas as arquiteturas de software precisam equilibrar a complexidade inerente. Ela pode estar oculta no próprio aplicativo (monolíticos) ou pode ser enviada às estruturas de comunicação da equipe (microsserviços).

Foco em sistemas e design

Para projetar uma arquitetura de microsserviços efetiva, você precisa fazer uma análise profunda no sistema. É necessário entender as interações entre os serviços, recursos corporativos e experiência do usuário. Essa análise aprofundada dos sistemas também inclui as estruturas de comunicação e processos na cultura organizacional. O Gartner observou que, sem um claro entendimento sobre o sistema geral, as arquiteturas de microsserviços terão o mesmo desempenho que os estereotipados monolíticos: "A adoção de uma abordagem holística que contemple a arquitetura de software, a infraestrutura e processos de desenvolvimento, é o que gerará resultados otimizados. Sem considerar essa abordagem, os problemas serão iguais ao de um sistema de software monolítico."[22]

Restrições de recursos

É esperado que haja restrições de recursos com os monolíticos devido aos problemas com a escalabilidade. O sistema precisa ter capacidade suficiente de hardware para lidar com os picos de carga em serviços maiores, o que pode resultar em parte dessa capacidade ser inutilizada. Caso contrário, há o risco de o sistema não ter a capacidade necessária para lidar com os picos de uso.

Com os microsserviços, a arquitetura é flexível. Como cada serviço individual é muito pequeno, é fácil escalar novos serviços em recursos muito leves e temporários, como containers e instâncias de cloud.

Já que os requisitos de recursos individuais de um determinado serviço são relativamente pequenos, é possível ignorar ou minimizar as demandas únicas da arquitetura geral. Assim, podemos considerar que:

- A rede é sempre confiável.
- A latência é nula.
- A largura de banda é infinita.
- A rede é segura.

22 Knoernschild, Kirk. "Refactor Monolithic Software to Maximize Architectural and Delivery Agility". Insights do Gartner, 18 de maio de 2017

- A topologia da infraestrutura não muda.
- Há um único administrador.
- O custo de transporte é zero.
- A rede é homogênea.

Há outra restrição de recursos, mas ela afeta um caso de uso de nicho de computação. Os aplicativos de alto desempenho podem ter requisitos que exigem muito da arquitetura de microsserviços, como modelagem climática ou mapeamento de DNA.

É importante observar as falácias em torno dos microsserviços porque nenhum sistema é perfeito. A intenção não é encontrar uma tecnologia ou arquitetura perfeita que solucione seus problemas. Na verdade, concentrar-se na cultura e na comunicação e refinar os processos é o que criará uma organização madura e eficaz. Dessa forma, será possível projetar uma arquitetura funcional que atenda aos objetivos específicos da sua organização.

Repensar um aplicativo essencial

Boa parte da discussão sobre a transformação digital enfatiza a infraestrutura e a cultura. O que faz muito sentido, já que elas são considerações fundamentais. No entanto, elas são os meios para se alcançar o objetivo final, que é criar um aplicativo que seja útil para os usuários e relevante para a organização.

Os aplicativos essenciais possuem determinadas características:

- São responsivos aos usuários
- Refletem a principal função ou propósito corporativo
- São adaptáveis ou reativos às mudanças dinâmicas no ambiente
- Estão conectados em vários ambientes
- São leves e flexíveis, possibilitando adição e manutenção de funcionalidades

Quando um aplicativo tem essas características, ele é considerado essencial.

Tanto as arquiteturas de microsserviços quanto as monolíticas podem refletir essas características. A arquitetura é uma escolha de design. Mesmo com uma arquitetura monolítica, mudar sua perspectiva sobre os diferentes elementos é fundamental para criar um aplicativo moderno e ágil. A Etsy tem a capacidade de adotar a entrega contínua, fazer dezenas de atualizações por dia, executar altas cargas de usuário e até mesmo aplicativos mobile, porque conta com uma incrível arquitetura monolítica, criada para esse propósito.

Essa intenção deve ser refletida nas diversas e principais áreas das arquiteturas monolíticas e de microsserviços, incluindo:

- Integração.
- Gerenciamento de dados e consistência.
- Sistema de mensageria e comunicação de serviços.
- Processos coerentes ou padrões do fluxo de trabalho.

Em geral, nos ambientes monolíticos tradicionais, essas questões eram consideradas problemas que precisavam ser corrigidos ou aspectos indefinidos do aplicativo. Por exemplo, a integração era, muitas vezes, uma alternativa para combinar diferentes aplicativos ou fontes de dados. Ela unia diferentes partes do ambiente. Até os fluxos de processo eram vistos como uma forma de aumentar a produtividade em relação à intervenção manual e automação de fluxos de trabalho. No entanto, nem sempre isso era uma decisão de design básica.

Com um aplicativo moderno, os aspectos da arquitetura, como integração e fluxos de processos, não são secundários. Eles são fundamentais para a execução do aplicativo. Isso é muito claro nas arquiteturas de microsserviços e também válido nos melhores monolíticos.

Integração e sistema de mensageria

Com os microsserviços, a questão é como acoplar esses serviços de forma a manter a autonomia deles e permitir a comunicação livre ao mesmo tempo. Isso é um problema de integração e mensageria. A integração define como esses serviços separados comunicam entre si. É uma escolha importante de design. Do ponto de vista da infraestrutura, os microsserviços costumam ser divididos como "containers mais APIs", sendo necessário acoplar esses serviços. Nesse caso, os containers são os serviços, e as APIs é a conexão entre eles. (Uma abordagem semelhante seria "containers mais sistema de mensageria" ou qualquer tecnologia que forneça uma maneira de transmitir dados entre os serviços.)

Isso é diferente até mesmo das arquiteturas orientadas a serviços em que o sistema de mensageria e a integração não estão centralizados em barramentos, e os dados não são convertidos entre os serviços.

Fluxos de processo

Um aplicativo moderno é responsivo aos usuários. Isso é mais evidente nos aplicativos mobile, que são simples, imediatos e voltados para o consumidor. Quando um cliente começa uma transação (por exemplo, ver o saldo bancário ou procurar uma passagem aérea), é necessário lançar um fluxo de trabalho de imediato. Esse fluxo precisa ser adaptável à próxima escolha do usuário e ainda atender aos protocolos da organização. O gerenciamento de processos de negócios (BPM) tradicional era a principal forma de automatizar tarefas para obter eficiência. No entanto, ao lançá-lo em uma arquitetura de aplicativos moderna, o BPM se torna um elemento importante no fornecimento de funcionalidades, além de contribuir com a experiência do usuário.

Dados

Em algum momento, todos os dados passam a ter monitoração de estado. Os dados precisam ser armazenados e acessados por serviços na arquitetura (sejam eles distribuídos ou monolíticos). Portanto, é necessário entender como eles são transferidos entre os serviços e quais tipos de dados estão sendo gerados.[23] O desenvolvedor Christian Posta afirmou que a parte mais difícil dos microsserviços é o gerenciamento de dados e a tentativa de definir limites naturais e transacionais entre os serviços.[24] Há uma certa tensão entre a consistência, a acessibilidade e a autonomia dos dados. É importante definir os domínios naturais e, então, informar os modelos de dados e as estruturas de armazenamento.

---

23 Brown, Kyle. "Refactoring to Microservices, Part 2: What to Consider When Moving Your Data". IBM developerWorks, 4 de maio de 2016, https://www.ibm.com/developerworks/cloud/library/cl-refactor-microservices-bluemix-trs-2/index.html.

24 Posta, Christian. "The Hardest Part About Microservices: Your Data". 14 de julho de 2016, http://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/.

"As pessoas tentam copiar a Netflix, mas só é possível copiar o que está visível. Elas copiam os resultados, mas não os processos."[27]

– ADRIAN COCKCROFT, ANTIGO ARQUITETO-CHEFE DE CLOUD DA NETFLIX

AGILIDADE COM RESPONSABILIDADE

Um dos principais objetivos da transformação digital é acelerar o processo de lançamento de aplicativos. No entanto, rapidez é apenas uma melhoria na eficiência. Para a transformação, a rapidez tem uma finalidade: ela possibilita inovações rápidas, novos recursos e a habilidade de testar novas ideias.

Ron Kohavi, engenheiro renomado e gerente geral da equipe de experimentos com inteligência artificial da Microsoft, afirmou em sua apresentação na Computer Science and Engineering Technology Open House 2013 da Universidade de Minnesota que menos de um terço das ideias melhora as métricas que deveriam ser aprimoradas.[25]

A maneira de avaliar uma boa ideia é por meio de testes completos e eficazes, e não apenas na qualidade do código, mas sim na experiência e preferência do usuário. Isso só é conhecido por meio da experiência. Como Ron mencionou, "os dados superam a intuição".

Essa é a finalidade da entrega contínua e das técnicas avançadas de implantação. A CI/CD é a plataforma para implantação rápida. As técnicas de implantação são ferramentas usadas para experimentação e refinamento.

Atrás desses dois estágios vem a mudança na cultura, que incentiva a inovação e dá suporte a falhas e riscos. A inovação não é um único ponto ou um destino, mas um processo que é alimentado por experimentações. Estar disposto a cometer erros durante o processo de inovação requer uma cultura de humildade.

Autosserviço, automação e CI/CD

Uma das reestruturações iniciais para a transformação digital é adotar uma cultura de DevOps, com pequenas equipes dinâmicas e comunicação aberta entre elas. A próxima etapa é a tecnologia, ao fornecer uma infraestrutura com suporte para ciclos de desenvolvimento rápidos.

Há duas etapas de tecnologia muito relacionadas com:

- Infraestruturas de autosserviço e elásticas: habilidade de solicitar e receber uma instância de acordo com as especificações exatas. Esse processo acontece quase que instantaneamente.
- Automação ou orquestração: capacidade de criar e gerenciar automaticamente várias instâncias em um ambiente.

Essas são tecnologias complementares: não é possível realizar a automação sem um ambiente elástico. Além disso, é difícil gerenciar centenas de possíveis instâncias sem as ferramentas que tornam o processo consistente e reproduzível.

Aprimorar a tecnologia nessa etapa da transformação pode aumentar a produtividade de maneira tangível. Um cliente da Red Hat conseguiu reduzir o tempo de disponibilização de cinco dias para quase 15 minutos, além de transformar o processo manual em automático ao incluir um catálogo de autosserviço para que os desenvolvedores pudessem solicitar rapidamente sistemas virtuais.[26] Essa mudança liberou recursos nas operações e aumentou a produtividade (e a moral) dos desenvolvedores.

---

25 *"Online Controlled Experiments: Introduction, Insights, Scaling, and Humbling Statistics". SOBACO University of Minnesota, 18 de outubro 2013, https://sobaco.umn.edu/content/online-controlled-experiments-introduction-insights-scaling-and-humbling-statistics.*

26 *"Red Hat Virtualization aumenta a eficiência e o custo-benefício". Estudo da Forrester "Total Economic Impact", 26 de janeiro de 2017, www.redhat.com/pt-br/resources/virtualization-tei-forrester-analyst-paper.*

27 *https://twitter.com/kelseyhightower/status/641886057391345664*

"O Docker é melhor descrito como containers aprimorados. Isso porque seu conjunto de ferramentas é baseado nos princípios da infraestrutura imutável, em que a configuração de aplicativos é determinada pelo desenvolvedor e colocada em produção com o mínimo de alterações adicionais."[28]

– IDC

As tecnologias são complementares, mas não são prescritivas. As infraestruturas elásticas podem representar a cloud (pública ou privada), máquinas virtuais ou containers. A automação pode ser um componente no sistema da infraestrutura (como o projeto "Heat" de orquestração com o OpenStack®) ou uma ferramenta externa, como o Red Hat CloudForms ou Kubernetes com containers baseados em Docker. Há diferentes caminhos tecnológicos dependendo das habilidades da sua organização e da infraestrutura existente.

A principal razão dos containers (como o Docker ou Red Hat OpenShift) serem tão associados à CI/CD é porque eles fornecem ambientes de sistema rígidos e reproduzíveis. Isso significa que menos problemas são transferidos entre ambientes organizacionais completamente diferentes. Essa é a famosa defesa do "funcionou no meu laptop" que ocorre quando uma mudança no código é transferida do desenvolvimento para a produção. As máquinas virtuais ou as instâncias de cloud podem variar entre o sistema operacional subjacente e as versões do pacote. Os containers precisam ter configurações idênticas no sistema operacional para que a imagem selecionada seja a mesma entre as implementações.

Essa consistência proporciona uma ótima base para a primeira parte da CI/CD, a integração contínua. Com ela, as mudanças no desenvolvimento são constantemente compiladas e integradas a cada check-in. Dessa forma, os problemas são detectados mais rapidamente. Isso costuma ser combinado com um conjunto de testes automatizado para verificar a estabilidade e funcionalidade. Esse processo contínuo de check-in, compilação e teste garante a mais alta qualidade do código.

Após adotar a integração contínua, sua organização pode dar sequência com a implantação contínua, levando as mudanças para produção mais rapidamente. Essa agilidade é mutualmente benéfica para os desenvolvedores e as equipes de operação. Os desenvolvedores e líderes corporativos têm a satisfação de ver novas soluções disponíveis no mercado mais rapidamente. As equipes de operações podem aplicar correções e até mesmo vulnerabilidades e exposições comuns críticas (CVE) em um tempo muito menor, garantindo um sistema mais seguro e com melhor desempenho.

O termo "contínuo" pode ter diferentes significados, dependendo do seu ritmo de desenvolvimento e necessidades corporativas. Com uma ótima arquitetura monolítica (processos e tecnologia sólidos com uma arquitetura de aplicativos mais tradicional), é possível fazer lançamentos toda semana com uma única atualização. A única restrição são os requisitos para processos ágeis e urgentes.[29] Com os microsserviços, qualquer serviço pode ser atualizado, com ciclos de urgência sobrepostos, para que as atualizações na arquitetura geral aconteçam diariamente.

Implantações avançadas e inovação

Os estágios que levam à CI/CD estão relacionados com a rapidez e a qualidade na disponibilização de aplicativos. No entanto, como Ron disse, a maioria das ideias não alcança o propósito inicial. Em uma apresentação, ele mostrou como isso acontece fornecendo uma série de imagens de diferentes designs do mecanismo de busca Bing e pedindo ao público que adivinhasse, por instinto, quais versões os usuários preferiam. Ele fez quatro perguntas em sequência e, de centenas de participantes, apenas uma pessoa permaneceu em pé.[30]

A ideia que ele quis passar é de que os dados superam a intuição. O design do Bing foi finalizado por meio de muitos testes de usuários, e eles descobriram que apenas um terço das ideias melhorou a experiência do usuário (e a mesma quantidade a danificou ativamente).

---

28 "The Emergence of Microservices as a New Architectural Approach to Building New Software Systems", da série "INDUSTRY DEVELOPMENTS AND MODELS", IDC. Al Hilwa, junho de 2015.

29 Spazzoli, Raffaele. "The Fast-Moving Monolith: How We Sped-up Delivery from Every Three Months, to Every Week". Red Hat Developers, 27 de outubro de 2016, https://developers.redhat.com/blog/2016/10/27/the-fast-moving-monolith-how-we-sped-up-delivery-from-every-three-months-to-every-week/.

30 "Online Controlled Experiments: Introduction, Insights, Scaling, and Humbling Statistics". SOBACO University of Minnesota, 18 de outubro 2013, https://sobaco.umn.edu/content/online-controlled-experiments-introduction-insights-scaling-and-humbling-statistics.

Infraestrutura de aplicativo para experimentação

Em 2006, quase uma década antes do termo "microsserviços" se popularizar no desenvolvimento, o desenvolvedor Neal Ford definiu o que é a programação poliglota em seu blog Meme Agora.[31] Ele identificou uma mudança no ambiente de programação.

Nos aplicativos corporativos mais antigos, havia apenas uma única linguagem. Conforme os aplicativos começaram a adotar arquiteturas de cliente servidor ou baseadas na web, algumas linguagens eram específicas para determinadas ações (como o JavaScript para uma interface web de usuário ou SQL para bancos de dados). No entanto, o aplicativo principal ainda era escrito em uma única linguagem central.

A programação poliglota de Neal é a ideia de que não haverá mais uma linguagem de programação central. No entanto, dentro da arquitetura geral de aplicativos, os serviços ou as funções individuais poderão ser escritos em linguagens totalmente diferentes e que melhor atendam às necessidades de um serviço específico.

A programação poliglota representa uma das necessidades básicas de um ambiente de desenvolvimento ágil: os desenvolvedores precisam ser capazes de testar algo novo e fora do design principal de um aplicativo. Isso é válido para aplicativos monolíticos efetivos e arquiteturas de microsserviços.

É importante considerar a flexibilidade e as opções dentro do ambiente de desenvolvimento ao criar uma plataforma para experimentação. Uma experimentação precisa oferecer suporte para:

- Várias linguagens.
- Vários tempos de execução.
- Ambientes de implantação flexíveis, como cloud física ou híbrida.
- Arquiteturas de aplicativo flexíveis e alteráveis, que permitam a adaptação do aplicativo às mudanças do ambiente ao longo do ciclo de vida.
- Padrões abertos ou habilidade de fazer padronização com iteratividade.

Os containers são um bom exemplo desse suporte, embora seja possível alcançar o mesmo resultado com outras plataformas. As bibliotecas, linguagens e versões de um container individual são prescritivas de maneira inerente. No entanto, um catálogo de containers pode ter centenas ou milhares de imagens diferentes, com suporte para linguagens e ambientes de execução distintos. Com esse tipo de plataforma, os desenvolvedores encontram a linguagem e o ambiente de execução exatos para realização do serviço como desejado. Além disso, as equipes de operações podem implantar esses serviços com mais eficiência.

Padrões de implantação para inovação

As técnicas avançadas de implantação oferecem estrutura e clareza para o processo de inovação. As metodologias de implantação maduras criam um ambiente que possibilita a experimentação, feedback e análise. Quanto mais experimentação, melhor a inovação.

Estes são padrões comuns de implantação. Um deles ou todos podem ser apropriados, dependendo da natureza do aplicativo e do ambiente do usuário.

---

31 Ford, Neal. "Polyglot Programming". Meme Agora, 5 de dezembro de 2006, http://memeagora.blogspot.com/2006/12/polyglot-programming.html.

"*Os dados superam a intuição.*"

RON KOHAVI, GERENTE GERAL DE EXPERIMENTOS COM INTELIGÊNCIA ARTIFICIAL, MICROSOFT, UNIVERSIDADE DE MINNESOTA, DEPT. OF COMPUTER SCIENCE AND ENGINEERING TECHNOLOGY OPEN HOUSE 2013 [23]

Ambientes "blue-green"

Criar ambientes blue-green é uma forma de mitigar riscos ao implantar mudanças. Uma nova compilação é transmitida por todos os ambientes no pipeline de CI/CD. Há dois ambientes idênticos para produção (blue e green), no entanto, apenas um estará ativo. A mudança é implantada no ambiente ocioso na produção. Após a verificação desse ambiente, o roteador é trocado, e o tráfego é transmitido para o ambiente atualizado.

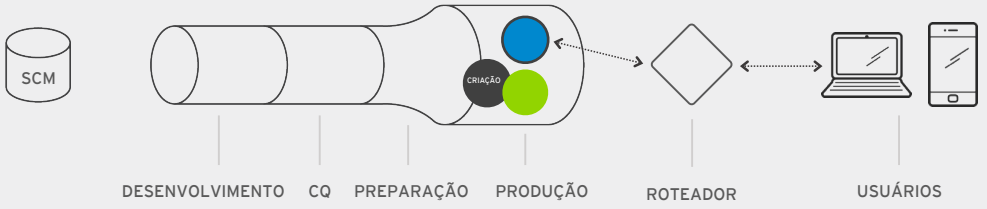

Figura 9. Pipeline da implantação blue-green

Lançamentos canários (Canary releases)

Eles são semelhantes à implantação "blue-green". A diferença é que o lançamento inicial vai apenas para um subconjunto de usuários no ambiente (os canários titulares na mina de carvão). Conforme o feedback é recebido dos usuários, esse subconjunto pode aumentar gradativamente até que todos os usuários sejam eventualmente trocados.

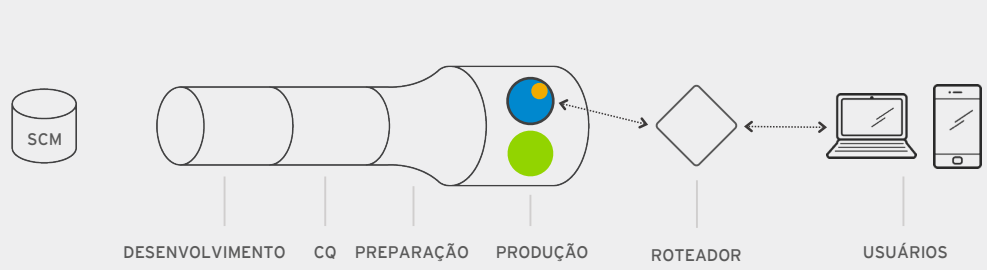

Figura 10. Pipeline da implantação canário

Isso pode ser usado como parte de uma técnica de teste para avaliar diferentes funcionalidades ou designs de aplicativos em pequenos grupos dentro de um ambiente de produção ativo e com padrões reais de tráfego e uso.

Testes A/B

Os testes A/B apresentam dois designs diferentes aos usuários e, em seguida, avaliam qual tem o melhor desempenho de acordo com as métricas desejadas. É como se os usuários classificassem suas experiências ou enviassem feedback, mas isso também pode ser feito de forma mais sutil. Por exemplo, a combinação de testes A/B com lançamentos canários pode comparar dois possíveis designs ou até mesmo funcionalidades ocultas e depois avaliar como os usuários interagem com eles, com o ambiente atual como linha de base.

Por exemplo, na Figura 11, o aplicativo mobile parece ser idêntico para os usuários. No entanto, os ambientes A/B usam diferentes algoritmos para testar as recomendações de produtos.

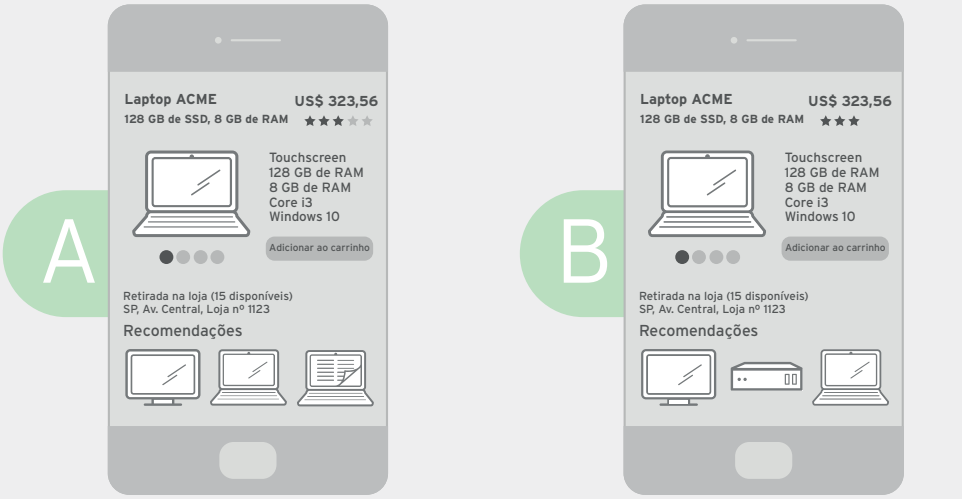

Figura 11. Teste com a interface A/B

Uma vez que um determinado design é bem-sucedido, ele pode ser lançado para um conjunto maior do ambiente, como seria feito com um lançamento canário.

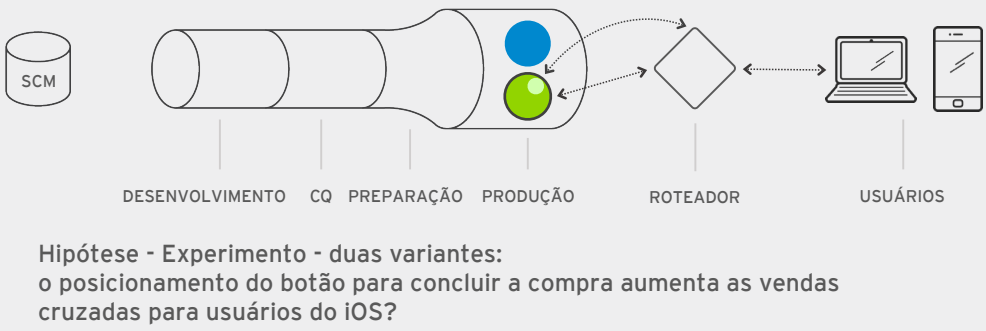

Figura 12. Pipeline do lançamento do teste A/B

Quando feito da forma correta, isso pode transformar um ambiente de produção em um ambiente de experimentação, possibilitando que as equipes sejam mais inovadoras e mais relevantes em seus projetos.

Esse tipo de teste é essencial para criar valor a partir de dados em vez de usar a intuição.

COMO ENSINAR UM ELEFANTE A DANÇAR

Escolha a etapa

Há muitas etapas entre um ambiente mais tradicional em cascata e microsserviços totalmente distribuídos. Rachel conta que toda arquitetura de software é o resultado da "tensão entre acoplamento e coesão".[32] Conforme você começa a planejar uma estratégia de transformação digital, essa tensão é representada na sua infraestrutura e cultura atual.

Determine um objetivo que sua organização possa realmente alcançar. Isso não significa uma meta mais fácil, já que o propósito da transformação digital é mudar substancialmente a cultura, os processos, a arquitetura e a tecnologia. No entanto, significa entender o que você está tentando alcançar com essa mudança e, então, avaliar com clareza o que é necessário para conquistar esse objetivo. Pergunte a si mesmo:

- Como as equipes e grupos estão atualmente divididos?
- Quais são os padrões de comunicação entre eles?
- Quem está encarregado atualmente de planejar os ciclos?
- A respeito da funcionalidade, o que falta para a sua arquitetura de aplicativos atual alcançar o estágio desejado?
- Qual é o nível de risco ou tolerância a falhas na sua organização?
- Quão claro é o entendimento sobre seus fluxos de material e informações? (Este é o mapeamento de valor da organização.)
- Com que frequência você precisa lançar uma atualização para atender às necessidades dos clientes ou operações?
- Que nova funcionalidade é necessária para alcançar os objetivos corporativos ou necessidades de desenvolvimento?

Defina os princípios operacionais

As mudanças culturais são a base de todas as modificações de processo, tecnologia ou arquitetura que sua organização realizará para alcançar a transformação digital.

Embora seja básica, a criação de um conjunto de princípios fundamentais que tenha o suporte do gerenciamento e das equipes pode reforçar as iniciativas de transformação digital e unificar as equipes.

---

32 Laycock, Rachel. "Continuous Delivery". Apresentação da tarde. Red Hat Summit - DevNation 2016, 1º de julho de 2016, São Francisco, Califórnia. https://www.youtube.com/watch?v=y87SUSOfgTY

"Com uma sólida arquitetura de aplicativos, DevOps maduro, processos ágeis e uma equipe de gerenciamento de dados focada, é possível criar um ambiente de microsserviços eficaz. Esse ambiente diminui o tempo de provisionamento de desenvolvimento em 75%."

– GARTNER[33]

Esses princípios podem parecer simples, mas talvez seja útil ter clareza e ser explícito nas atitudes e comportamentos mais importantes para a mudança na cultura, especialmente se alguns deles (como incentivar o risco e a experimentação) são opostos à cultura atual. Por exemplo:

- Falhas acontecem.
- Experimentações são positivas.
- A organização (pessoas) é prioridade.
- Pratique a melhoria contínua.
- Comprometer-se com o aprendizado para a vida toda.
- Seja sempre responsável.
- Seja transparente.

Figura 13. Esforço reduzido em um lançamento bem orquestrado

33 "Innovation Insights for Microservices", Anne Thomas e Aashish Gupta, 27 de janeiro de 2017

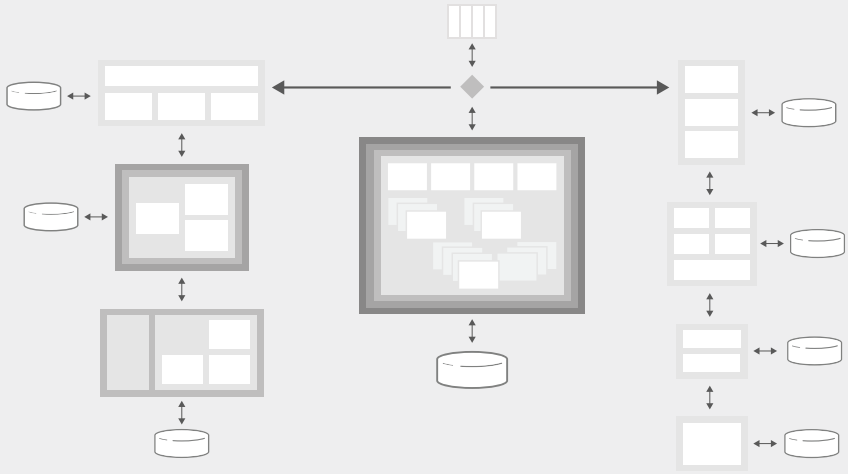

Figura 14. Separação de serviços individuais para "substituição" do monolítico

Substitua o monolítico

Em algum momento, a transformação digital afeta o aplicativo e a arquitetura atual. Se você tem um aplicativo monolítico, há duas formas relacionadas para começar a lidar com a dívida técnica:

- Separe os serviços existentes, uma vez que eles precisam de atualização e substituição ("strangling").
- Crie uma nova funcionalidade em serviços separados e independentes ("starving").

Isso não requer um compromisso total com os microsserviços. O Key Bank, um cliente da Red Hat, precisava reduzir os tempos de lançamento de trimestres para semanas. A empresa conseguiu alcançar esse objetivo usando um aplicativo monolítico.[34] A lógica principal do aplicativo pode permanecer em um monolítico. No entanto, é possível que haja domínios lógicos em que os serviços são separados, normalmente para as interfaces do usuário. Por exemplo, criar uma camada separada para serviços, como sistemas de API, front-ends mobile e outras interfaces do usuário, possibilita que serviços orientados a clientes ou externos tenham iterações mais rápidas ou ciclos de vida separados do que o aplicativo principal. Isso gera mais inovações com menos risco corporativo.

O Gartner recomenda essa abordagem gradativa para as arquiteturas distribuídas porque ela é "iterativa e enfatiza áreas de valor".[35]

Seja qual for a etapa final da evolução digital da sua organização, há três áreas básicas que você precisa abordar:

- Agilidade no design da arquitetura.
- Experimentação.
- Automação.

---

34 Spazzoli, Raffaele. "The Fast-Moving Monolith: How We Sped-up Delivery from Every Three Months, to Every Week". Red Hat Developers, 27 de outubro de 2016, developers.redhat.com/blog/2016/10/27/the-fast-moving-monolith-how-we-sped-up-delivery-from-every-three-months-to-every-week/.

35 Olliffe, Gary. "How to Design Microservices for Agile Architecture". Insights do Gartner, 30 de janeiro de 2017.

Projete arquiteturas que ofereçam agilidade futura

Se o seu foco está na otimização de processos para aprimorar o monolítico ou na criação de microsserviços, a base da sua arquitetura precisa ser ágil. Frequentemente, agilidade significa ser híbrido. O Gartner recomenda começar até mesmo projetos novos como monolíticos e usar microsserviços à medida que eles amadurecem.[36] O Gartner afirma: "Talvez, inicialmente, você acredite que isso seja um esforço de desenvolvimento desnecessário. De fato, nossos estudos sugerem o contrário: a abordagem que prioriza monolíticos reduz o risco, aumenta a produtividade inicial e garante o desacoplamento e decomposição do seu aplicativo no conjunto certo de microsserviços."[37]

Lembre-se da teoria da força do design: crie um processo de desenvolvimento que enfatize a clareza e a simplicidade. O código precisa ser fácil de entender. A funcionalidade e a finalidade devem ser claras. Conforme o aplicativo amadurece, ele pode evoluir para arquiteturas mais distribuídas. Ter bons processos de desenvolvimento e implantação ajudará a manter a agilidade.

Reserve tempo e orçamento para experimentação

Você precisa ter um espaço no orçamento para experimentar novas tecnologias e recursos de aplicativos. Por exemplo, a IDC recomenda reservar 2% do orçamento de TI apenas para a experimentação com as tecnologias de container.[38]

O Gartner afirma que a tecnologia por si só não é um objetivo. Portanto, mudanças como "implantação na cloud" ou " adoção de microsserviços" certamente falharão devido a falta de clareza no propósito dessas mudanças.[39]

No entanto, objetivos digitais estratégicos, frequentemente, são suportados pelas mudanças tecnológicas. Reduzir o time to market pode exigir a migração para containers, por exemplo, e é possível executar aplicativos JavaTM EE em plataformas em containers, como o Red Hat JBoss® Enterprise Application Platform (EAP).

Reserve recursos que possibilitem aos seus desenvolvedores e grupos de operações identificar tecnologias úteis e aprender habilidades que ofereçam suporte a qualquer infraestrutura implantada.

Automatize tudo

A automação gera dois benefícios muito claros (dentre outros): maior eficiência ao eliminar as etapas manuais e consistência e reprodução que são inerentes. Automatizar todas as etapas do desenvolvimento (no início) e implantação (conforme os processos amadurecem) fornece às equipes ciclos de feedback sobre as mudanças em cada uma delas à medida que os stakeholders movem do desenvolvimento para as operações e para os clientes. Essa abordagem aprimora a qualidade geral do código.

Como primeira etapa, é importante definir uma linha de base do status atual da organização para alimentar a estratégia de automação. As etapas incluem:

- Definir métricas relevantes.
- Visualizar ou fazer a diagramação dos fluxos de trabalho atuais.
- Identificar os principais participantes em etapas diferentes.

---

36 Ibid.

37 Ibid.

38 Elliot, Stephanie, et al. "IDC TechBrief: Containers". IDC. Janeiro de 2017.

39 Knoernschild, Kirk. "Refactor Monolithic Software to Maximize Architectural and Delivery Agility". Principais Insights do Gartner, 18 de maio de 2017.

CONCLUSÃO

Com o tempo, os aplicativos corporativos costumam regredir para o monolítico estereotipado: indefinido, complexo para atualizar e lento para incorporar novas funcionalidades. Ainda assim, eles também fornecem as operações corporativas fundamentais que geram receita. Esse é o famoso elefante corporativo.

No entanto, o elefante pode ser treinado para garantir agilidade e transformação, contanto que haja uma visão objetiva de qual deverá ser o estado final. Isso é a transformação digital como um processo evolucionário. Não há um resultado ideal. Cada caminho da evolução reflete o propósito e a personalidade únicos da própria organização.

Avalie cada etapa da evolução digital: DevOps, autosserviço ou ambientes elásticos, automação, pipelines de CI/CD, implantações avançadas e microsserviços. Crie sua estratégia de transformação digital de acordo com o nível de evolução que melhor representa as suas necessidades corporativas.

Concentre-se em desenvolver sua cultura e equilibrar as mudanças na tecnologia com as modificações correspondentes nos processos. Assim, suas equipes poderão oferecer todo o suporte para a tecnologia.

Conforme os processos amadurecerem, comece a avaliar o aplicativo e a arquitetura. De acordo com a necessidade, isole ou desenvolva serviços independentes e crie uma arquitetura ágil que possa ser adaptada à medida que as prioridades corporativas mudam ou novas surgem.

Por fim, incentive a inovação. Isso significa tolerar alguns riscos e falhas, dentro dos limites dos objetivos corporativos e necessidades dos clientes. E é preciso ter disciplina para reservar recursos de tempo, dinheiro e infraestrutura. A inovação é proporcionada pela experimentação, o que garante mais chances de sucesso no processo de transformação digital. Essa experimentação também traz de volta aquilo que sempre atraiu os desenvolvedores e os profissionais de operação para o mundo da tecnologia: a capacidade de criar e ver suas criações em ação.

SOBRE A RED HAT

A Red Hat é a líder mundial no fornecimento de soluções de software open source, utilizando uma abordagem de parceria com as comunidades para oferecer tecnologias confiáveis e de alto desempenho de cloud, Linux, middleware, armazenamento e virtualização. A Red Hat conta com premiados serviços de suporte, treinamento e consultoria. Como um hub de conectividade em uma rede global de empresas, parceiros e comunidades open source, a Red Hat ajuda a criar tecnologias relevantes e inovadoras que permitem a ampliação recursos disponíveis e preparam os clientes para o futuro da TI.

Saiba mais em ***http://www.redhat.com/pt-br.***

AMÉRICA LATINA
+54 11 4329 7300
latammktg@redhat.com

BRASIL
+55 11 3629 6000
marketing-br@redhat.com

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

br.redhat.com
#8980_0917