UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

CENTRO DE INFORMÁTICA

2017.2

Trabalho de Graduação

Alternativa para Arquiteturas Monolíticas em Clouds Públicas

Aluno: Victor Nunes de Farias Neves (vnfn@cin.ufpe.br)

Orientador: José Carlos Cavalcanti (cavalcanti.jc@gmail.com)

Recife

2017

Victor Nunes de Farias Neves

Alternativa para Arquiteturas Monolíticas em Clouds Públicas

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: José Carlos Cavalcanti

Recife
2017

Victor Nunes de Farias Neves

Alternativa para Arquiteturas Monolíticas em Clouds Públicas

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em _____ de _____ de 201__.

BANCA EXAMINADORA:

Prof. Drº José Carlos Cavalcanti

_____

Prof Drª Carina Frota Alves

_____

“O esforço supera o talento quando o talento não se esforça”

Tim Notke

AGRADECIMENTOS

Gostaria de agradecer aos meus familiares que sempre acreditaram e continuam acreditando em mim, me dando suporte em toda a parte da graduação, principalmente a meus pais, Severino Nunes e Erivan Elvira, que acompanharam a trajetória diária e me ensinado valores que sempre carregarei comigo.

Deixo um agradecimento a todos os professores, os quais transmitiram o conhecimento para minha formação acadêmica e profissional, em especial ao meu orientador, José Carlos Cavalcanti, que se propôs a me auxiliar no fim dessa caminhada, me direcionando com afinco neste trabalho.

A todos os colegas que conheço do centro de informática, gostaria de agradecer a companhia desse tempo em que caminhamos juntos realizando trabalhos em grupo, discussões e com momentos de descontração, em especial a José Barbosa Neto, Valdi Junior e Vitória Maciel, os quais sempre estiveram ao meu lado no decorrer da graduação, compartilhando as alegrias e dificuldades durante a caminhada.

Aos demais que contribuíram no meu desenvolvimento acadêmico e profissional e a todos meus amigos, meu sincero agradecimento por tudo que fizeram por mim.

Resumo

O advento da computação em nuvem e sua capacidade de processamento computacional permitiram criar um ambiente onde as aplicações estão aptas para entregar uma grande quantidade de recursos para dar suporte aos serviços empresariais.

Por esta razão, qualquer empresa, emergente ou consolidada, que necessite de um ambiente para uma nova solução de mercado, ou aprimorar alguma já existente, pode contar com a cloud (nuvem) e não necessita gastar com uma infraestrutura física para tal. Esse fato tem elevado a competitividade entre as empresas.

A forma como as aplicações são construídas está passando por mudanças significativas nos últimos anos, fundamentalmente naquelas baseadas no modelo arquitetural monolítico. Por anos, essa solução foi capaz de atender às necessidades da empresa para que estas fossem capazes de entregar seus serviços com qualidade. No entanto, com o crescimento exponencial de informação na internet, elas estão se tornando obsoletas por não conseguirem suportar o processamento dessas informações, mesmo em ambientes de nuvem.

A nova maneira de arquitetar os softwares, os denominados microsserviços, vem tomando lugar das arquiteturas monolíticas exatamente para poder lidar com o crescente processamento das informações trafegando pelas aplicações em alta escala. Outro padrão, como a arquitetura orientada a eventos, também está sendo utilizado para atingir essa finalidade.

Nesse contexto, este trabalho tem como objetivo tentar entender as razões pelas quais tais arquiteturas estarem sendo capazes de realizar esse processamento, e como elas se encaixam no ambiente da nuvem.

Palavras-chave: Microsserviços, Arquitetura Orientada a Eventos, Computação em Nuvem

Abstract

The advent of cloud computing and its processing capabilities allows to build an environment where the applications are able to deliver a great amount of resources that gives support the company's services.

Thence, any company, new or consolidated, who needs an environment to make a new solution, or enhance one that already exists, can count on cloud and doesn't need to spend with a physical infrastructure. This fact increased the competitiveness among the companies.

The process to build application is going through meaningful changes over the last years, fundamentally based on monolithic architecture model. For years, this solution was able to attend the company's needs and make them qualified to deliver his services with quality. However, with the growth of the information in the internet, the monolithic architecture is getting obsolete and it can't handle too many information, even when the application resides on cloud

The new way to build software, known as microservice, has taken the place of monolithic architecture to make the application capable to handle the growing of information processing in high scale. Another pattern, as event-driven architecture, also is being used for this purpose.

In this context, the goal of this work is trying to understand the reasons that make these architectures able to do that processing and how they fit on cloud environment.

Keywords: Microservice, Event-Driven Architecture, Cloud Computing

Lista de Ilustrações

Sumário

1. Introdução

Neste capítulo é feita uma introdução a este Trabalho de Graduação. A Seção 1.1 faz uma breve discussão sobre a principal motivação para desenvolver este trabalho. Na Seção 1.2, são apresentados os objetivos específicos que se pretendeu alcançar. A Seção 1.3 explicita como o trabalho foi organizado.

1.1. Motivação

Nos últimos anos, com a expansão da tecnologia, as mudanças no cenário empresarial ocorrem com mais frequência. *Startups* (empresas emergentes) nascem com ideais de entregar produtos ou serviços de maneira eficiente e eficaz, empresas consolidadas focam em melhorias de produtos existentes e em alcançar novos nichos de mercado. O uso de ferramentas e métodos para ajudar a encontrar novos clientes e manter os antigos tornam as corporações mais competitivas no nível estratégico.

As ferramentas utilizadas até bem recentemente pelas corporações para entrega de serviço tinham como base uma infraestrutura nomeada monolítica. O uso de estruturas com características de arquiteturas monolíticas torna a escalabilidade, um dos fatores essenciais para a expansão de negócios, um desafio para gerenciar recursos de infraestrutura. [1][2]

Com a finalidade de poder gerenciar seus recursos melhor para suportar a alta demanda de seus negócios, empresas começaram a migrar para arquiteturas suportam o crescimento serviços, como microsserviços. Notáveis exemplos são aqueles como o da *Netflix*, que conseguiu se preparar para a entrega de seu serviço de streaming quando resolveu modificar sua arquitetura para que fosse escalável. [3]

As empresas emergentes não possuem ambientes robustos como as empresas consolidadas: por falta de recursos monetários, o custo de aquisição de equipamentos para uma infraestrutura própria de TI (servidores, equipamentos de comunicação e segurança, etc.) é um desafio para elas. Outro fator que dificulta o crescimento é a velocidade de entrega dos serviços: por não ter um modelo de negócio totalmente validado, nem todas as tarefas são automatizadas para entregar

valor do serviço em tempo hábil.

Estes fatores podem ser problemáticos para as startups que pretendem oferecer serviços com a mesma qualidade das empresas consolidadas, principalmente as que já utilizam ambientes escaláveis e automatizados (como o exemplo da *Netflix*).

1.2. Objetivo

O trabalho tem como objetivo realizar um estudo de caráter teórico e exploratório com a finalidade de investigação das arquiteturas alternativas para substituição do modelo de arquitetura monolítica existente no mercado, tomando como base a adoção dos conceitos como os de microsserviços e event-driven, padrões de software que tem uma relação entre si para construção de softwares [53] e oferecem ambientes robustos e escaláveis.

Os principais pontos de enfoque do trabalho são:

- Apresentar os conceitos dos padrões de software microsserviços e event-driven;
- Apresentar os conceitos das arquiteturas de nuvem;
- Realizar uma correlação das abordagens arquiteturais em um ambiente heterogêneo sobre uma plataforma cloud pública.

Através das arquiteturas estudadas, este trabalho visa apresentar uma alternativa para que empresas emergentes sejam capazes de concorrer com a infraestrutura dos ambientes corporativos consolidados. Para que isso seja possível, foram elaboradas duas questões que pode direcionar as startups a alcançar o mesmo patamar de infraestrutura das empresas consolidadas:

- Como é possível que as empresas emergentes construam um ambiente capaz de lidar com uma quantidade de informação que cresce abruptamente através da rede?
- De que forma as startups podem aproveitar a arquitetura em cloud junto às arquiteturas de microsserviços e event-driven para obter um ambiente com melhor custo benefício?

O trabalho não se propõe a realizar um estudo de desempenho dos padrões de software, embora haja referências a estudos relacionados. Também não é focado em

nenhum estudo de algoritmos com o intuito de mostrar a interoperabilidade de como os conceitos estudados são organizados.

1.3. Estrutura do Trabalho

Este trabalho está dividido em 6 (seis) capítulos, incluindo este capítulo de introdução. O segundo capítulo está focado em explicar os conceitos da arquitetura de microsserviços e como ela transforma o ambiente corporativo, apresentando as características em ambientes monolíticos e fatores que auxiliam na criação dessa arquitetura.

O terceiro capítulo está relacionado ao estudo da arquitetura event-driven, suas características e seu comportamento, indicando como pode auxiliar na comunicação entre componentes.

O quarto capítulo descreve os conceitos da arquitetura cloud, suas características básicas, os tipos de nuvem e modelos de entrega de serviço.

O quinto capítulo está focado em apresentar como as três arquiteturas podem colaborar formando um ambiente heterogêneo onde empresas consolidadas ou startups podem construir suas soluções.

O sexto e último capítulo são apresentadas as conclusões sobre a pesquisa realizada e possíveis trabalhos futuros.

2. Microsserviços

2.1. Antes do Microsserviço

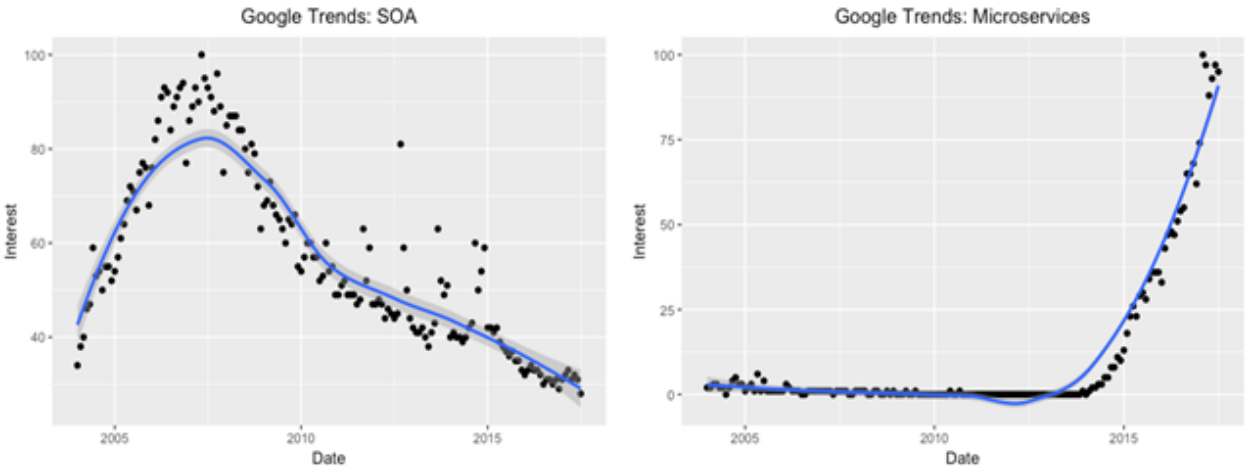
Crescer de forma rápida no mercado nunca foi tão importante como nos momentos atuais, onde a demanda por serviços para um negócio pode incrementar rapidamente. A tecnologia vem em auxilio para que as soluções comerciais possam ser entregues o mais rápido possível. Logo, a tecnologia e sua arquitetura correspondente precisam estar em conformidade com o negócio da empresa, e com isso promove-se uma entrega de serviço e/ou produto que esteja em nível de excelência. [4]

No início do milênio, companhias estavam procurando formas de se conectar com outras empresas e não existia um modelo que auxiliasse a escalabilidade de forma tão rápida na época. No entanto, um dos modelos, o *SOA – Service Oriented Architecture*, ou Arquitetura Orientada a Serviços – começou a ganhar força, visto que sua estrutura possibilitava provisionamento de escalabilidade, integração, interoperabilidade e demais necessidades que precisavam ser atendidas para realização de negócios entre empresas. Foi também uma das primeiras a ser dirigida a serviços, e que se preocupou em modelar uma arquitetura que pudesse crescer em ambientes distribuídos. [5]

Atualmente, o entusiasmo pela abordagem *SOA* diminuiu. É possível notar, pela figura 1, que enquanto o interesse pelo uso da abordagem já estava em queda no início da década atual, outro paradigma crescia. Microsserviço, como é chamado, possui muitos traços da abordagem *SOA*. As particularidades utilizadas pelo microsserviço podem proporcionar um ambiente onde o serviço principal é entregue com mais fluidez. Alguns argumentos podem ser utilizados para que SOA perdesse interesse em relação a Microsserviço:

- Excesso de coordenação: o impacto que causa utilizando muita coordenação gera muita dependência e gastos com esforço e tempo para desenvolver, testar, implantar e manter;
- Granularidade: muitas vezes, em SOA, são criados subsistemas que englobam mais de uma função e impactam tanto em transações quanto em desempenho. [6]

Figura 1 - Tendência do uso dos termos SOA e Microsserviços



Fonte: [42]

2.2. Conceito de Microsserviço

> Em resumo, a arquitetura de microsserviços é uma abordagem que busca desenvolver uma única aplicação como um conjunto de pequenos serviços, cada um executando seus próprios processos, e se comunicando através de mecanismos que transmitem pequenas mensagens, frequentemente por *API*. Os serviços são construídos sobre as capacidades de negócios e implantados independentemente por um conjunto de sistemas de automação de implantação. [MARTIN FOWLER e JAMES LEWIS, 2014]

Em sua narrativa, Lewis e Fowler deixam bem claro qual o significado e o objetivo de uma arquitetura voltada para serviços. A definição de microsserviços dá base a diversas características que abrangem toda uma questão de infraestrutura e de negócio que devem servir como pilares no momento em que a empresa decida utilizar a abordagem em seu ambiente. Portanto, nos próximos itens deste capítulo as características e a infraestrutura da abordagem são o foco da discussão.

2.3. Negócio e Organização

Três características de uma empresa, organização ou instituição sofrem significativo impacto no momento de adoção ou mudança para o paradigma de

microsserviços: comunicação, alinhamento de times e estímulo à inovação [8]. No momento em que ocorre o planejamento para implantar uma política adequada para microsserviços, essas características precisam estar presentes no *mindset* (mentalidade) dos colaboradores: a comunicação leva ao entendimento dos contextos do serviço, o alinhamento permite que o time esteja em sintonia e mantenha a produtividade, e o estimulo a inovação impulsiona na melhoria e/ou criação de um serviço diferenciado no mercado.

Outro ponto que pode ser levantado em relação a um negócio é a organização das equipes. Com microsserviços, há uma tendência a um número médio de seis pessoas por time para gerenciar os serviços, e cada integrante representa uma área de atuação que seja diferente, ou seja, cada equipe possui pessoas de competências distintas com o objetivo de desenvolver ou gerenciar os serviços regidos por eles [7]. A diversidade de informações permite preencher lacunas de conhecimento e serve como um apoio para o desenvolvimento ser o mais assertivo possível.

Um dos pontos mais importantes das organizações é a capacidade de automação dos processos. Processos automatizados, nas palavras de Ann All [52], auxiliam na entrega de novos produtos com qualidade e geram um aumento significativo na produtividade de tarefas.

Esse processo de automação está inteiramente ligado ao conceito de microsserviços (seja em tecnologia ou na organização de procedimentos organizacionais) e dessa forma, a utilização de processos que são feitos sem intervenção humana, reduzem o tempo de entrega manteando a competitividade. [51]

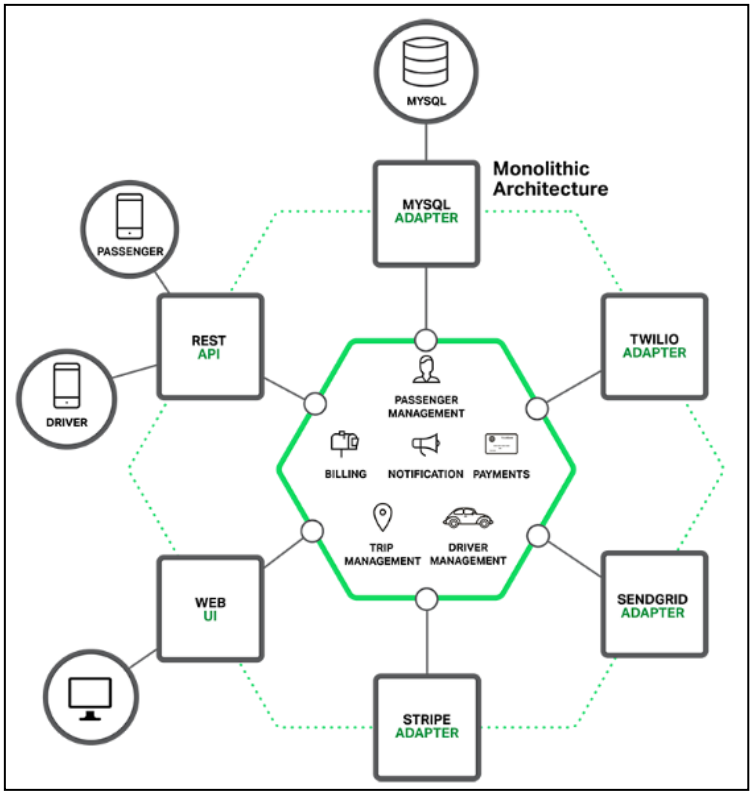
2.4. Características Principais – Um Comparativo com Ambientes Monolíticos

No que diz respeito à infraestrutura de tecnologia de informação e comunicação, os microsserviços trouxeram uma proposta diferente do que havia no mercado. Seguindo a definição sugerida por Lewis e Fowler, é possível listar alguns requisitos importantes na adoção dessa arquitetura. Para tal, uma comparação com a forma arquitetural monolítica é necessária.

2.4.1. Arquitetura Monolítica

Na figura 2, é possível inferir como uma arquitetura monolítica é estruturada e, a partir dela, como uma série de observações acerca de sua construção pode ser contrastada com relação à dos microsserviços, em termos de métricas de produtividade e escalabilidade.

Figura 2 - Arquitetura Monolítica



Fonte: [10]

O primeiro ponto a se notar pela figura 2 (que representa visualmente a arquitetura e seus componentes se comunicam) é que todos os serviços estão dentro de uma mesma área, o que pode indicar certo nível de dependência de um serviço para outro, por menor que seja.

Isso pode não ser um problema quando a aplicação é pequena, mas na medida em que há aumento de serviços acoplados, a alta dependência dos serviços tornará mais complexa a manutenção de seus códigos, e o entendimento de até onde cada serviço faz interface com outro [10]. Ou seja, para ambientes que crescem muito, a dependência força o gasto de esforço e tempo, prejudicando a produtividade.

A alta dependência dos módulos acaba desencadeando uma série de outros fatores, como a perda de confiabilidade do serviço, em caso de falha de execução, gerando impacto em outros serviços dependentes [10]. No caso de implantação de novos serviços, o impacto é em todo o ambiente que está em produção, uma vez que toda a aplicação sofrerá o processo de implantação a cada alteração.

Esses aspectos podem comprometer a empresa em relação à sua escalabilidade: se a empresa realizar uma avaliação de desempenho de uma aplicação e identificar que ela necessita escalar alguns serviços (não todos) infelizmente constatará a necessidade de escalar toda a aplicação, já que há acoplamento nos módulos dos serviços. Além disso, é preciso levar em conta que, ao escalar o serviço, uma quantidade grande de recursos de infraestrutura (que são escassos) será dedicada para uso.

No final, em função de alguns serviços, o preço pago para escalar um serviço em específico é um desperdício de recursos tecnológicos ou, em certos casos, o uso exorbitante de recursos financeiros, caso da empresa desejar adquirir mais recursos tecnológicos para essa finalidade.

Apesar de sua facilidade em implementação, possuir ambientes para desenvolvimento já consolidados no mercado e poder possibilitar os testes com mais facilidade durante toda a aplicação [11], o conjunto de fatores negativos vinculados ao ambiente monolítico faz dele uma opção não muito atraente para o mercado.

2.4.2. Arquitetura em Microsserviços

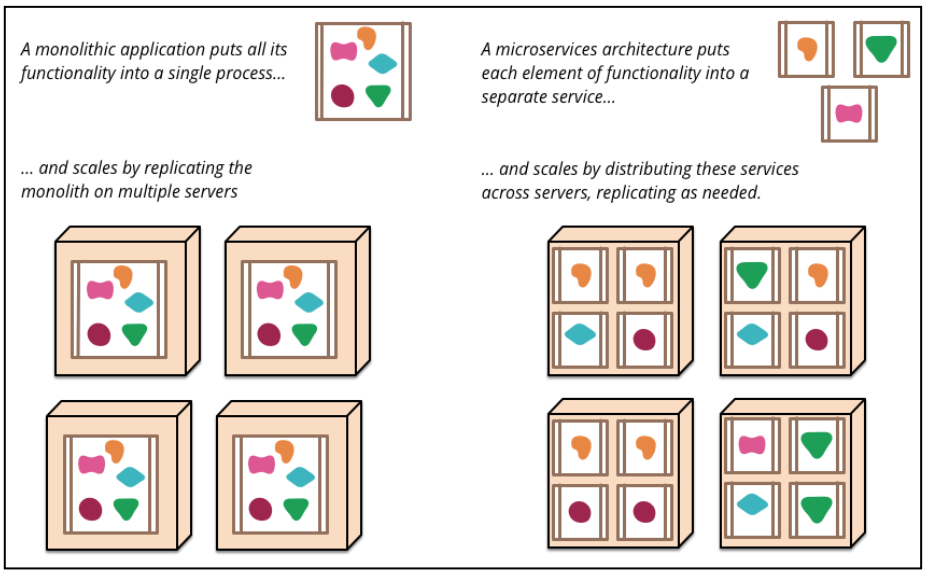
Microsserviços também possuem algumas complexidades, como a dificuldade de manter uma base de dados consistente (visto que elas são distribuídas), ou testes de integração dos serviços. No entanto, os benefícios acompanhados pela abordagem são vantajosos, quando se almeja que o negócio tenha crescimento robusto.

Dos principais ganhos em utilizar microsserviços, o primeiro que é possível citar é a independência dos serviços ou, em um termo mais técnico, perda de acoplamento. Na definição do termo, é defendido que a alteração de um serviço não irá ter impacto de mudanças noutro [9], o que é um ponto positivo, principalmente em caso de falhas de serviço: se algum momento uma falha e/ou inoperância ocorrer em um dos serviços,

os outros não serão afetados.

Por serem modularizados e serem pequenos, os grupos de trabalho poderão mensurar a extensão de seus serviços e entendê-los por completo. A divisão de responsabilidades para cada serviço fica clara para a equipe, ficando a cargo dela a decisão sobre qual padrão, qual tipo de banco de dados ou qual linguagem utilizar, não obrigando aos colaboradores a aprender tecnologias obsoletas [10]. Os colaboradores podem ser realocados para outros projetos com outras tecnologias que eles dominam, sem afetar a produtividade.

Figura 3 - Comparação de escalabilidade das arquiteturas monolíticas e de microsserviços



Fonte: [7]

Como já foi apresentado, o maior ganho na adoção é a escalabilidade. Observa-se na figura 3 que, no lado esquerdo, um ambiente construído em meio à abordagem monolítica torna difícil a escalabilidade de serviços específicos, enquanto no lado direito os serviços podem ser clonados de forma a facilitar o aumento de acesso a eles diminuindo a carga de requisições em um mesmo servidor, como também ser capaz de lidar com falhas. Os recursos podem ser mais bem aproveitados, evitando um gasto financeiro enorme dependendo da arquitetura.

2.5. Infraestrutura

Um aspecto relevante na estruturação da arquitetura de microsserviços são as ferramentas e técnicas para a montagem de sua infraestrutura. Tanto em uma migração quanto num projeto “*greenfield*” (começando do zero), a escolha certa de ferramentas e

técnicas, e de sua combinação, pode trazer resultados benéficos ou dores de cabeça futuras. Com uma grande quantidade de ferramentas no mercado, uma das que mais vêm chamando atenção para a comunicação entre os serviços é a denominada *API – Application Programming Interface*.

2.5.1. APIs

API, utilizando a descrição de Jacobson, Brail e Woods [2011], é uma forma de duas aplicações conversarem através da internet utilizando uma linguagem que ambos conhecem. A quantidade de *APIs* no mercado está crescendo exageradamente [14], possivelmente um indicativo de várias questões interrelacionadas, tais como o grande consumo delas nas companhias para focar no núcleo de desenvolvimento, no negócio, e, de certa forma, na terceirização de outras funções onde os colaboradores não têm expertise.

Entre os tipos de *API* no mercado, os microsserviços acabam utilizando muitas das vantagens das *API gateways*. Por sua troca de mensagens ser leve, as *APIs* evitam uso de recursos de banda trafegada na rede. Elas são capazes de realizar tarefas que vão desde segurança até transformação de informação, formando uma camada de abstração para o dispositivo que está consumindo as informações, e removendo a complexidade de construção da comunicação dos serviços [13]. Por vezes, pode ser considerado um orquestrador para organização interna ou um elemento de coreografia para se comunicar em sintonia com outros serviços externos a empresa (termos discutidos na seção 3.2).

O controle de tráfego é outro benefício que essas *APIs* trazem. Um exemplo é o *Load Balance*, que é uma técnica que auxilia no escalamento de serviços, podendo evitar um grande número de requisições seja direcionada a um servidor, diminuindo a latência do serviço. [11]

Como já mencionado, existem *APIs* para diversos tipos de ambientes, no qual a equipe de desenvolvimento precisa conhecer como alocá-las junto com outras ferramentas utilizadas no ambiente de desenvolvimento. Porém, além de manter o canal de comunicação entre os softwares abertos, é preciso montar ambientes para que as tarefas de desenvolvimento sejam automatizadas o máximo possível.

2.5.2. DevOps

Com essa finalidade em mente, algumas metodologias de *DevOps* (termo muito utilizado para indicar alinhamento de equipes de desenvolvimento e operacional, em relação a processos, ferramentas e responsabilidades, utilizando conceitos da metodologia ágil para acelerar entregas com elevado grau de qualidade) auxiliam na sistematização desse ambiente.

Através de *DevOps*, a construção de processos na infraestrutura de microsserviços se torna automática. A redução o tempo de entrega e atualização do serviço e automação através das ferramentas, possibilita um ambiente de produtividade e, por consequência, criar valor para o negócio [9]. Por isso, é valido destacar três das metodologias de automação de atividades utilizadas em *DevOps*.

A primeira dessas metodologias é chamada de integração contínua. Como descrito pela Amazon, a integração sugere um ambiente em que haja repositório compartilhado de código, onde se evita um ambiente com códigos conflitantes, além de manter o desenvolvimento conciso através de um conjunto de testes automáticos, tendo como resultado a minimização do gasto de tempo por parte da equipe para testar manualmente [15].

Outra metodologia é a entrega contínua, e pode ser considerada como extensão da anterior. A entrega foca em automatizar todo o processo, principalmente na área de testes (desde testes unitários até aceitação), até a fase de implantação, o que encoraja sempre o trabalho com pequenas mudanças, e evita-se conflito de versões ou atualizações [16].

Há ainda uma última metodologia que se compromete em realizar a fase de implantação de forma automática, chamada de implantação continua, ou seja, qualquer código alterado, automaticamente se passa pela fase de testes, e, se não for levantado nenhum erro, ele é implementado automaticamente em ambientes de produção [16]. A utilização de cada fase depende da organização da empresa em relação aos processos, principalmente no momento de implantar.

No momento de colocar em prática, as metodologias vão tornar o processo conhecido como *Deployment Pipeline* algo automático. Esse tipo de deployment é um conjunto de processos de desenvolvimento que vai desde a codificação até a entrega/atualização da aplicação; e ao chegar nessa fase, existem algumas técnicas que podem ajudar na transição de versões do sistema.

O *Blue-Green Deployment*, uma opção para a transição, foca em redirecionar o tráfego para a nova versão da aplicação de maneira completa e rápida, utilizando um ambiente quase

igual ao de produção [17], priorizando a velocidade da implantação. Outra forma de transição é a *Canary Release*, que possibilita a transição de versão implantada através de porcentagem, evitando que todo o tráfego seja direcionado para a nova versão da aplicação, e resguardando a aplicação de possíveis falhas em caso de *bugs* na integração [11].

2.5.3. Considerações finais da infraestrutura de Microsserviços

O último ponto, porém, não menos importante, em relação à infraestrutura, é como cada serviço é dividido. Levando em consideração que a infraestrutura precisa ser mais bem aproveitada possível, não poderiam faltar ferramentas que virtualizam servidores (discutidas na seção 5.2), visto que elas auxiliam no melhor aproveitamento de recursos de servidor e melhoria da gestão de cada componente.

O interessante é notar que, para microsserviços, todos os conceitos e técnicas apresentados anteriormente, desde *API gateway*, passando por metodologias de automação de trabalho, e entrega de contínua da aplicação, acabam facilitando a intercomunicação entre os serviços de maneira automática, evitando um esforço manual constante por parte da equipe desenvolvedora.

O modo como cada microsserviço é entregue depende de como os fatores apresentados até aqui são posicionados e, desde o início, um processo de transição para adoção dessa arquitetura exige um esforço ainda maior para ambientes que não estejam acostumados com a dinamicidade que o padrão demanda.

É notável, também verificar que os benefícios, através de mudanças de comportamento todas as equipes, afetam desde pessoas mais técnicas, como o desenvolvedor, até as pessoas interessadas no progresso do negócio, como alta cúpula das corporações.

3. Arquitetura Orientada a Eventos

3.1. Conceito

Ao lidar com o desenvolvimento de softwares, as equipes precisam estruturar a forma segundo a qual os componentes serão capazes de se comunicar uns com os outros envolvidos na arquitetura. Logo, a troca de mensagens é um ponto crucial para o bom desempenho da aplicação. Com base nisso, o uso de uma arquitetura que inclua características de flexibilização e perda de acoplamento de componentes é imprescindível.

Para que seja possível caracterizar uma abordagem que trate as mensagens, os eventos aos quais elas estão conectadas, e os significados que elas trazem, alguns conceitos precisam ser apresentados.

O primeiro de todos que pode ser definido é o de evento. Nas palavras de Michelson [2006], evento é descrito como algo que chamou atenção dentro ou fora da empresa e que pode conter informações como nome, tipo de evento, hora que ocorreu ou qualquer indicação de alteração de dados.

O segundo conceito é sobre a arquitetura em si: a arquitetura orientada a eventos (do inglês *Event Driven Architecture*, ou *EDA*), também conhecida como arquitetura orientada a mensagens, é uma técnica computacional que foca na comunicação dos componentes computacionais através de troca de mensagens de forma inteligente [20], e sua abordagem sugere que todo e qualquer evento possa ser disseminado de forma automática para todos os interessados na mensagem [19].

Diferente de como foi apresentado em microsserviços, em que a construção da arquitetura era com base nos serviços, o principal foco de construção da arquitetura para *EDA* são os eventos. Apesar da diferença de abordagem, as *EDAs* são capazes de prover um ambiente onde as aplicações são independentes e escaláveis para o negócio.

3.2. Características Principais

Como outros padrões arquiteturais, uma das principais características nesta arquitetura é a possibilidade de prover desacoplamento de aplicações, facilitando a

reusabilidade de módulos, com menor quantidade possível de dependências. Além disso, o que facilita o seu uso é que as mensagens não precisam carregar muitas informações para os componentes; ou seja, apenas um conjunto mínimo de dados é necessário para os componentes as tratem. [22]

O que realmente torna *EDA* uma abordagem singular é o conceito de *publisher/subscriber*, isto é, uma comunicação desse tipo permite que um mesmo componente seja tanto um publicador de informações quanto um consumidor destas. O comportamento dessa abordagem é uma comunicação assíncrona de muitos para muitos, o que sugere que a comunicação de quem vai passar as informações não necessariamente precise chegar a todos os outros ambientes, tampouco quem recebe precise processar todas as informações. [21]

Por fim, há dois conceitos que precisam ser definidas: orquestração e coreografia. Em outros ramos de conhecimento, essas duas palavras podem ter outros significados, mas em TI, cada uma delas funciona como uma maneira para conduzir a informação entre os componentes.

Boris Lublinsky [2008] reuniu uma série de definições dos dois conceitos por vários estudiosos do assunto. Para ele, apesar desses conceitos divergirem em alguns pontos, a essência delas apresentava as seguintes características:

- Orquestração se refere a automação de execução de passos, que possui uma figura central capaz de realizar a coordenação desses passos;
- Coreografia, por outro lado, está relacionado a uma forma de comunicação entre duas ou mais partes, orientada a atingir os objetivos, coordenando os recursos para tal.

Os dois conceitos ficam mais óbvios quando o autor cita um trecho na expressão utilizada por John Evdemon: “*Numa perspectiva B2B, orquestração é interno a organização e coreografia é entre organizações. Uma organização não orquestra outra*”.

3.3. Infraestrutura

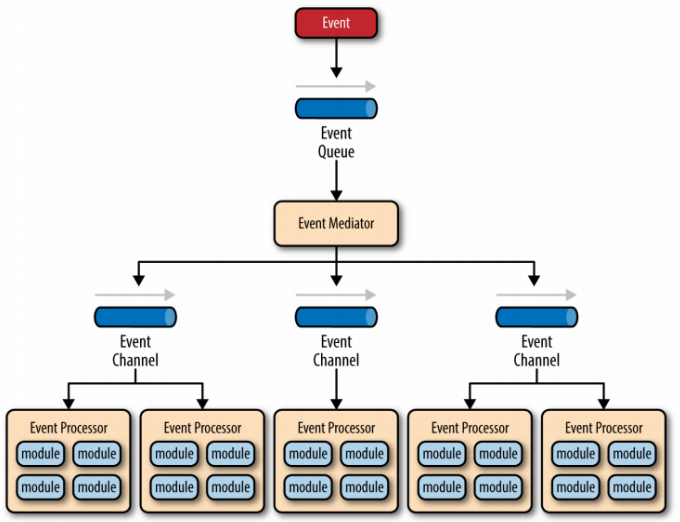
Apresentado e conceituado por Mark Richards [2015], a infraestrutura da arquitetura *EDA* pode ser construída principalmente de duas formas: com ou sem

mediador (topologia quebrada). É válido salientar que, independente das topologias utilizadas, toda e qualquer alteração em qualquer componente do sistema, seja ela uma adição, alteração de informação ou remoção de itens no banco de dados, são passíveis de gerar um evento.

3.3.1. Topologias

Seguindo para a primeira delas, a topologia com mediador (figura 4) necessita de certo nível de orquestração a fim de processar o evento. Para isso, existem quatro componentes que servem como canais condutores dos eventos: a fila de evento, o mediador, os canais de evento, e os processadores do evento.

Figura 4 - Topologia com Mediador



Fonte: [24]

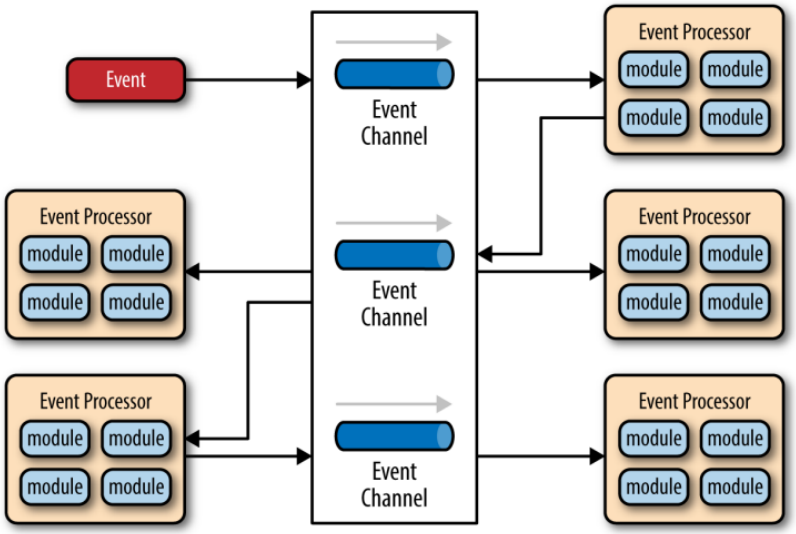
Nesta topologia, cada evento gerado por um componente segue uma sequência de passos até a informação chegar aos componentes que estão relacionados com a mensagem em questão:

- O processo inicia quando o evento é gerado e então encaminhado para uma fila de eventos iniciados. Este tipo de evento inicial vai ser transportado até o mediador do evento, um componente que gerencia os eventos;

- O mediador (um orquestrador) coleta as informações transpassadas pelo evento que iniciou e as repassa para os devidos canais de eventos. As informações que o mediador coletou serão transformadas em mensagens que serão processadas;
- Estas mensagens são encaminhadas para os canais de eventos específicos e as mensagens são repassadas para os processadores do evento, que devem ser construídos para realizar uma única tarefa de negócio sem depender de outras para concluir.

Por outro lado, topologia sem mediador (figura 5), o evento é enviado para todos os canais de eventos que serão responsáveis por repassar aos processadores que reivindicam para si os eventos e estes, além de processar o evento (quando a informação for de seu domínio), podem publicar outro evento informando que o processamento sobre tal evento foi encerrado. Funciona como um encadeamento de processamento do evento, cada processador realiza sua tarefa e encerra, avisando aos outros que terminou seu trabalho e outros podem iniciar, caso haja necessidade.

Figura 5 - Topologia sem Mediador



Fonte: [24]

Richards [2015] conclui que as duas topologias têm suas vantagens de uso, sendo necessário levar em consideração alguns aspectos. Por exemplo, se é importante que as tarefas sejam realizadas de forma sequenciada, o uso de um

mediador se torna mais apropriado. Caso contrário, em um ambiente de comunicação mais simples, com menos componentes, não é necessário usar o mediador por este ter uma maior complexidade de implantação do componente.

3.3.2. Componentes

O que as topologias não levam em consideração é o impacto que vai ter nas bases de dados. Como manter a consistência dos dados quando há a necessidade de escalar seus repositórios? Uma forma de responder essa pergunta foi descrita por Fowler [22], em seu artigo dedicado a arquitetura de eventos. Ele cita que, ao desenvolver um ambiente com eventos, é interessante utilizar *Event-Sourcing*.

Event-Sourcing procura guardar as mudanças de estados dos eventos. Por causa disso, há a possibilidade de guardar todas as mudanças de eventos. Essa característica se torna muito importante quando alterações indevidas são realizadas e precisam ser desfeitas [22]. Ela funciona como a centralizadora de informações (nas topologias, ela é implementada como o canal de eventos) e enviar as mensagens para os módulos, possibilitando que a informação chegue às bases de dados com consistência.

Para complementar a funcionalidade do último componente, o *Command Query Responsibility Segregation (*ou *CQRS*) pode ser utilizado em conjunto. Tecnicamente, este padrão divide procedimentos de consulta e gravação de dados independentes [25]. Dessa forma, é possível escalar procedimentos como consultar, alterar, inserir ou remover informação de dados em mais de um banco de dados sem perda de consistência.

4. Arquitetura em Nuvem

4.1. Conceito

Construída sobre servidores, roteadores, computadores pessoais e diversos outros dispositivos, os ambientes de TI oferecem conexão aos recursos tecnológicos e proveem as necessidades que os colaboradores precisam para realizarem suas tarefas e alcançar os objetivos da empresa. No entanto, esses ambientes podem custar uma quantidade massiva de recursos financeiros para as corporações por causa de aquisição e manutenção dos equipamentos.

Dessa forma, partindo do entendimento que o conceito de *cloud* inicialmente era abstrato, e metafórico, uma equipe do *National Institute of Standards and Technology* (NIST), conceituou a nuvem de uma forma bem aceita pela comunidade:

"*Cloud computing, é um modelo que permite o acesso sobre demanda a um conjunto de recursos compartilhados e configuráveis de acesso à rede (servidores, aplicações, serviços, etc.) que são providos e liberados com o mínimo de esforço gerencial ou interação com o provedor de serviços*" [26].

Em outras palavras, o mesmo ambiente que se pode construir fisicamente, adquirindo locais físicos, comprando equipamentos, contratando uma grande quantidade de pessoas para tomar conta dos dispositivos e os riscos ligados a todas as variáveis envolvidas na manutenção do parque tecnológico, serão transferidos para a empresa provedora de serviços cloud. É possível até calcular o custo de investimento em nuvem: empresas como Amazon, Microsoft, Google e tantas outras possuem ferramentas do tipo *cloud calculator* para ajudar a comparar o investimento. [27]

Para se ter uma ideia do crescimento, uma pesquisa realizada em 2013 constatou o ganho da utilização de cloud pelas organizações: o acréscimo na mobilidade dos funcionários, a redução de custos operacionais, melhoria na eficiência, entre outros fatores, apenas reafirmam o ganho no investimento em cloud [28]. Um relatório do *Gartner* chega a mostrar que o investimento em nuvem será bastante elevado nos próximos anos, chegando a quase duplicar em 2020 [29], mostrando o quanto de relevância a tecnologia que era concebida no fim dos anos 2000 passou a ter.

4.2. Características Principais

Considerando o conceito apresentado, existe uma série de características que são utilizadas para a descrição do que uma cloud pode dispor. Segundo o *NIST*, a nuvem possui cinco principais características. São elas: *On-demand self-service*, *Broad network access*, *Resource pooling*, *Rapid elasticity* e *Measured service*.

Começando pelo *On-demand Self-service*, sua concepção define que um ambiente cloud deve ter o mínimo de interações entre um usuário e um gerenciador de recursos da cloud quando o usuário deseje utilizar o recurso. O poder de realizar alterações no consumo de recursos está na mão do cliente que deve estar ciente de que irá pagar pela quantidade que consumir. [30]

O *Broad Network Access* permite que o acesso seja realizado através de qualquer dispositivo que tenha permissão de conectividade com a infraestrutura, seja ela simples ou complexa, não importando em qual local esteja [26]. Isso permite flexibilidade das organizações contratar pessoas competentes de diversas regiões do globo para cuidar das aplicações.

Resource pooling é uma das características mais importantes e que deve chamar a atenção por causa de sua flexibilidade: o provedor de serviços utiliza um modelo arquitetural conhecido como *multi-tenancy* (ou multi-inquilino), que permite o acesso de diversos clientes ou empresas compartilhe recursos do servidor (plataforma de desenvolvimento, aplicação online, armazenamento, memória, entre outros) logicamente separados [31]. O cliente não tem ciência da localização da infraestrutura que disponibiliza os recursos para as tarefas.

Outro diferencial, *Rapid elasticity*, permite que o crescimento ou redução dos recursos de acordo com as necessidades do negócio. Para o consumidor, essa capacidade de escalabilidade deve transparecer ilimitada, mesmo que na realidade não seja. Os recursos devem estar disponíveis a qualquer hora para uso. [26]

Por último, a capacidade de verificar o quanto está sendo consumido para momentos de realizar o pagamento, *Measured service*. Realizar o controle do quanto é utilizado dos recursos, afinal, como já dito, através desse controle é possível medir o quanto será cobrado pela utilização do serviço. [32]

Essas cinco características são bem conhecidas e bem aceitas na composição

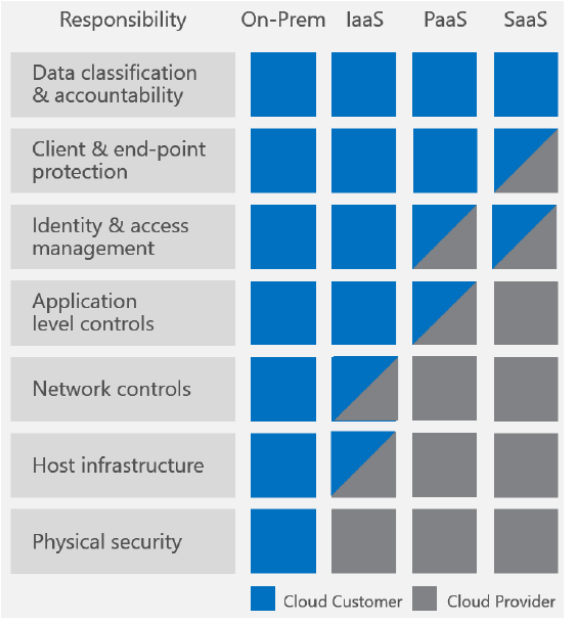
das características da cloud. No entanto, quando se fala em cloud, também se tem em mente o conceito de ser algo seguro. As empresas estarão contratando os serviços e precisam estar cientes de que o ambiente é capaz de suportar catástrofes, ou seja, que possui cópias de segurança [33], mecanismos de acesso e gerenciamento seguro da cloud e de garantir funcionamento ininterrupto. [34]

4.3. Modelos de serviços

Quando uma empresa contrata serviços de nuvem, é comum que o foco do contrato seja um ambiente que permita o aperfeiçoamento de alguma de suas soluções ou processos que levem a entrega do serviço. Para isso, há uma necessidade natural de definir o componente ou conjunto deles que serão adquiridos.

Pensando nisto, torna-se relevante avaliar os modelos de serviços para o ambiente com base nas necessidades da empresa. Os mais comuns a se avaliar são três: infraestrutura como serviço, plataforma como serviço e software como serviço.

Figura 6 - Classificação dos modelos de serviços em nuvem.



Fonte: [43]

Como é possível verificar na figura 6, cada um dos modelos (com exceção do "*on-prem*" que expressa a ideia em que a infraestrutura está localizada em um ambiente

tecnológico proprietário da companhia e, portanto, totalmente gerenciada por ela) possui um conjunto de características, controlados em parte pelo servidor que proporciona os serviços, e, em parte pelo cliente que os utiliza. Por terem propósitos diferentes, fica evidente a necessidade de conceituá-las.

Começando pela infraestrutura como serviço (*IaaS*), que pode ser definido como um ambiente que oferece recursos de rede, armazenamento, virtualização e qualquer outro tipo de recurso que seja fundamental para implementação das aplicações [26].

Outras definições chegam a complementar o que seria *IaaS*: *"a habilidade de criar, gerenciar e consumir elementos de infraestrutura*", "*obter infraestrutura cloud (rede, armazenamento) sobre demanda, elasticamente*", "*utilizar materiais brutos para executar os fluxos de trabalho de forma flexível, escalável e sem sobrecarregar ambientes*" [35]. Apesar de serem visões logicamente válidas, sempre utilizam os recursos fundamentais já mencionados para a construção desse modelo.

Partindo para o segundo modelo básico, plataforma como serviço (*PaaS*), com um poder menor de tipos de recursos disponíveis em relação ao *IaaS*, ele tem como objetivo permitir a implantação de aplicações criadas pelo próprio cliente ou pelo provedor do serviço utilizando qualquer tipo de linguagem, biblioteca ou ferramentas [26]. As aplicações do provedor podem ser alteradas de modo a se adequarem às necessidades do cliente. É vantajoso quando se pensa em criar equipes de desenvolvimento que não podem estar em um mesmo local físico, além de proporcionar agilidade no desenvolvimento das aplicações e portabilidade delas para outras plataformas. [36]

Chegando ao último modelo, o software como serviço (*SaaS*), que apresenta uma aplicação ou solução que o provedor disponibiliza ao cliente. Tal aplicação aproveita todo o ambiente cloud e pode ser acessada por várias pessoas em regiões geográficas distintas [26]. Talvez por não possuir controle sobre nenhum outro componente apresentado na figura, implique que este seja o mais restritivo, mas não deixa de ter suas utilidades, como não precisar se preocupar com configuração de aplicações em desktops ou licenciamento para uso. Um bom exemplo disso é o conjunto de aplicações do Google, como o *Docs* ou o *Presentation*.

Apesar dos três modelos serem os mais básicos, existe uma quantidade enorme de modelos de serviços, desde “tudo como serviço” (EaaS) que faz referência a qualquer produto se tornar um serviço, “banco de dados como serviço” (DBaaS) focando na estrutura de banco de dados, entre outros. Contudo, um dos modelos que vem chamando atenção e entrou na curva de inovação do *Gartner* foi o “Função como Serviço” (*FaaS* – pode ser conhecido também como *Serverless*) [37].

FaaS é uma nova forma de implantação de aplicação com o objetivo de se desprender dos servidores, no sentido de não se preocupar com a necessidade de provisionar novas máquinas virtuais quando o código demanda um alto processamento, assim deixando o foco de implementação apenas para o código e o resto estará na responsabilidade do provedor desse tipo de modelo. [38]

Diferente dos outros três, a cobrança pelo uso do modelo *FaaS* está no processamento de cada função na cloud: nada é cobrado do cliente quando ela está no ambiente, apenas quando ela é processada pelo servidor [38]. Além disso, os procedimentos são realizados automaticamente e podem ser escaláveis, sendo gerenciados pelo provedor de serviço. Apesar de ainda estar em processo de amadurecimento, o *Serverless* pode se tornar um modelo de serviço em que os padrões de arquitetura de software escaláveis, como microsserviços, utilizarão como base para entregar valor ao negócio em tempo hábil.

4.4. Tipos de cloud

Além dos modelos de serviço, é essencial considerar os tipos de nuvem em que os mesmos podem ser implantados. Para este estudo, três deles serão considerados: as públicas, as privadas e as hibridas. Todos eles se baseiam no conceito da *cloud computing,* entretanto cada um possui especificidades no uso de recursos (servidores, processamento, virtualização) e na forma como estes são provisionados, acessados e administrados.

Dessa forma, para realizar uma migração e manter a infraestrutura na nuvem, um estudo comparativo deve ser realizado, com foco na identificação das características específicas e o propósito para utilização de cada um dos modelos.

A nuvem pública é uma estrutura montada e gerenciada pelo provedor de

serviços a fim de oferecer recursos computacionais e cobrar de acordo com a quantidade de recursos utilizados. Através dela é possível criar ambientes onde os estes estão disponíveis podem ser aproveitados e realocados conforme a demanda dos clientes.

Para que isso ocorra, as clouds públicas são construídas em ambientes com alto poder de desempenho computacional, facilitando o provisionamento e escalabilidade de serviços. Além disso, toda a preocupação com a compra, gerência ou manutenção relacionada à estrutura do modelo de serviço ofertado fica sobre a responsabilidade do provedor de serviço. [40]

Todavia, a segurança nesse tipo de nuvem é um desafio, visto que, mesmo que haja processos destinados a isso, há uma limitação nos dados e a certeza de que eles estão protegidos adequadamente. Outro fator que pode complicar a adoção de clouds públicas se deve a conformidade com leis e regulações com algumas entidades, o que pode dificultar a migração. Nesses casos, a utilização de um acordo de serviço entre o cliente e o provedor que proporcione a não violação das regras de uso seria uma ação que pode atender tanto a conformidade quanto à segurança. [41]

Em contrapartida, a cloud privada é descrita como uma estrutura construída em que a infraestrutura e os serviços ofertados são destinados exclusivamente a um cliente, sendo administrada por um provedor de serviços externo ou pela própria organização [26]. Por causa disso, a computação em nuvem privada possui uma grande vantagem em relação às nuvens públicas: a flexibilidade o controle das políticas de segurança e da privacidade dos dados.

Neste tipo de ambiente, a classificação dos dados, como são mensurados os riscos de acesso as tecnologias ou a forma de implantação de políticas próprias de segurança pode ser feitas de forma irrestrita, são mais abertas para adaptação frente às necessidades da organização [41].

No entanto, a nuvem privada sofre com a limitação de expansão ou redução de recursos: diferente da nuvem pública, que pode crescer no momento oportuno, a do tipo privado não segue o mesmo caminho, dependendo de uma comunicação com o provedor de serviços para realizar mudanças em um espaço de tempo adequado para prover os recursos necessários [40].

Já a nuvem hibrida é descrita pela união dos dois tipos descritos anteriormente em um único ambiente e incorpora as características de ambos. Esse aspecto faz com que essa arquitetura de nuvem necessite de comunicações padronizadas, fato que aumenta a complexidade de gerenciamento dela [40]. A padronização se faz ainda mais necessária quando há mais de um modelo de serviço em cada tipo de nuvem para que não haja perda de informações.

A utilização dessa terceira forma de arquitetar nuvens é indicada para infraestruturas que necessitam lidar com arquiteturas complexas, sendo indicadas quando as soluções corporativas necessitam de ambientes de alta performance, escalabilidade e segurança.

5. Ambiente de Múltiplas Arquiteturas

Pela busca de soluções que se adaptem à demanda, as soluções corporativas procuram construir sistemas que sejam capazes de ser modelados conforme as suas necessidades. Por esse motivo, as arquiteturas tecnológicas devem corresponder às expectativas, devem exercer um nível alto de segurança e robustez, e ser flexível para mudanças nos serviços.

O breve registro sobre microsserviço na seção 2.2 deste trabalho se encaixa nesse quadro e é uma solução que atende às exigências de um ecossistema ágil e que sempre está se transformando. Contudo, a complexidade em lidar com a escalabilidade das bases de dados torna a adoção dessa arquitetura um desafio. O motivo? Quando a aplicação está sob essa arquitetura, uma base de dados é construída de acordo com o serviço que está prestando, e ela usa a tecnologia que melhor se adequa ao contexto dos dados [44].

Os registros acerca dos modelos e tipos de nuvens nas respectivas seções 4.3 e 4.4 mostram que a infraestrutura por trás da *cloud computing* é vantajosa para que as empresas emergentes se equipararem frente à infraestrutura das empresas consolidadas.

Com base nestes fatores e com o objetivo de responder as questões estabelecidas na introdução deste trabalho, as próximas seções são focadas em como as startups podem construir uma infraestrutura que a permita crescer rapidamente e entregue o mesmo nível de serviço que empresas bem-sucedidas em seus ramos de negócios, como *Uber*, *Amazon*, *Spotify* ou *Netflix*.

A resposta para a primeira pergunta, sobre a construção de um ambiente com uma quantidade crescente de informação na rede, é simples: utilizando um ambiente que tenha as características de microsserviços e event-driven (discutido na seção subsequente). Para a resposta da segunda questão, que se refere ao aproveitamento das três arquiteturas apresentadas com o melhor custo benefício, pode ser descrita como um conjunto de dois fatores:

- Gerenciando a infraestrutura de microsserviços e event-driven por meio de virtualização dos serviços na nuvem (discutido na seção 5.2);

- Focando na utilização de nuvens públicas como base para infraestrutura que sustente as duas outras arquiteturas (discutido nas seções 5.3 e 5.4).

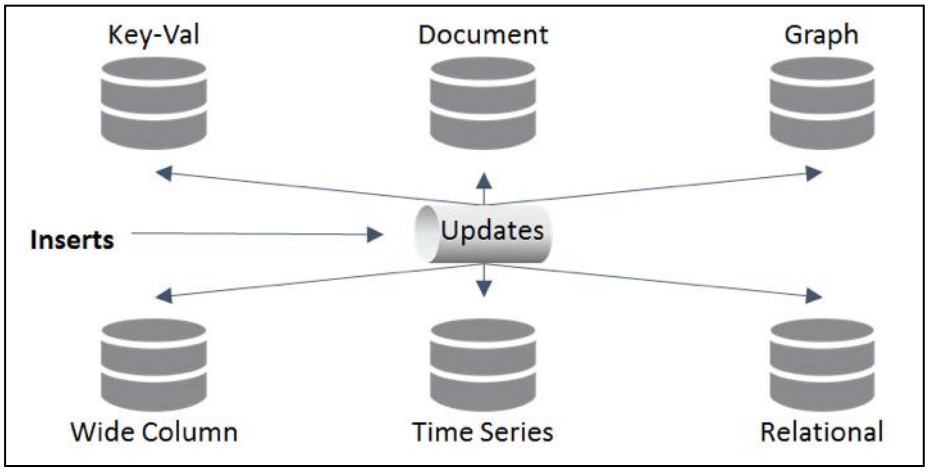
5.1. Correlacionando Microsserviços e Event-Driven

Sobre a condição da complexidade de armazenamento e consistência das bases de dados, fica claro que utilizando apenas a arquitetura em microsserviço não é suficiente para manter os serviços independentes.

Por causa disso, é interessante considerar a característica que faz a arquitetura event-driven única: ou seja, a capacidade de cada componente ser um *publisher/subscribe*. Fazer um serviço ter a capacidade de enviar, receber e processar eventos de acordo com suas informações é o caminho para maximizar o potencial de microsserviços, permitindo que os dados estejam consistentes.

Como já mencionado anteriormente, quando uma ação é executada, um conjunto de passos serão realizados pelo event-driven. Desse modo, cada serviço que receber uma interação e informações acerca do domínio que forem adicionados, alterados ou removidos, é criado um evento com as informações relevantes do que aconteceu e enviado para os serviços interessados nas informações. Cada serviço possui rotinas automáticas responsáveis por receber e interpretar as transações e realizam o processamento dessas informações gerando outros tipos de eventos (quando necessário).

Figura 7 - Distribuição de dados através do *Event-Sourcing*



Fonte [45]

Além disso, ao utilizar uma base de dados capaz de gravar todos os estados dos eventos (*Event-Sourcing*), permite-se criar um ambiente com dados consistentes, pois ele funciona como um centralizador de informações que permite a distribuição de informações dos eventos para as bases de dados, mesmo quando há serviços com mais de uma instancia [45]. É possível ter uma ideia visual do funcionamento do *Event-Sourcing* através da figura 7.

Apesar da complexidade de implementação da tecnologia, com ela é possível manter as bases de dados consistentes e escaláveis conforme a necessidade de escalar os serviços. Os benefícios de utilização do *Event-Sourcing* em conjunto com o *CQRS* (termo descrito na seção 3.3.2) vão de maximizar o desempenho na obtenção e persistência da informação, mantêm as características organizacionais de microsserviços como a liberdade de escolher o tipo de linguagem de programação, e banco de dados para construir o serviço, e, por oferecer um histórico de atualizações dos dados, servir para processos de auditoria [46].

Se levar em consideração uma arquitetura monolítica, onde não existe uma divisão de informações precisa, qualquer informação que é alterada gera uma série de processos de troca de mensagens com diversos componentes que, por muitas vezes, não tem relação com o domínio que a mensagem, desperdiçando processamento. Apesar de ser mais simples de desenvolver, o impacto negativo de informações, mesmo que simples, pode afetar áreas da aplicação de forma drástica.

Event-Driven é um dos padrões arquiteturais que focam no arranjo de informações para que sejam conduzidas de forma ordenadas e em grupos, a fim de evitar criar gargalos de informações em um ponto centralizado (uma única base de dados centralizada, por exemplo). O uso de *Event-Driven* permite que as informações possam ser controladas no contexto em que elas pertencem, facilitando o controle de implantação de novos componentes no contexto adequado, evitando desperdício de processamento.

Incorporar esse padrão de tratamento de informações como eventos ao ambiente de microsserviços é uma forma para manter os serviços e dados relacionados a eles independentes uns dos outros. A união de microsserviços e event-driven como um padrão de software é um caminho para as startups que desejam construir ambientes

preparados para serem escaláveis, flexíveis, com alta performance e automatizados com menor uso de recursos se comparado a arquiteturas monolíticas.

5.2. Infraestrutura virtualizada

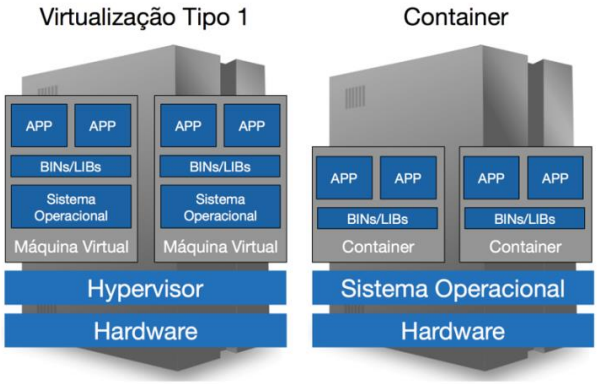
Quando se realiza a migração para a nuvem, a utilização de ambientes virtualizados é inevitável, tanto para o provedor do serviço quanto para o cliente. Os ambientes virtualizados permitem a gerência dos recursos de hardware, distribuindo todas as questões de armazenamento, processamento e memória proporcionalmente entre os componentes sobre essa tecnologia. No mercado, há duas formas de abordagem para criação desses ambientes: através de máquinas virtuais ou por container.

Para que os ambientes virtuais com máquinas virtuais (*VMs*) sejam criados, existe um componente que sempre está presente para dar o suporte a elas. Conhecido como *hypervisor*, ele permite realizar a divisão e compartilhamento dos recursos de processamento, memória e disco para os sistemas operacionais que estiverem instalados nele. O *hypervisor,* de acordo com Bill Kleyman [2012], possui duas versões para implementação:

- O primeiro tipo é implementado sobre o host físico, responsável por estar conectado a componentes de armazenamento e rede, permitindo comunicação direta com o hardware, possui alta performance de virtualização por causa disso;
- O segundo tipo é implementado via software, sobre um hardware lógico, acima de um sistema operacional anteriormente instalado em um sistema operacional. O acesso ao recurso físico é mais demorado, deixando a performance é menor em relação ao primeiro tipo.

A segunda opção de virtualização, por container, não exige que um *hypervisor* ou sistema operacional para cada aplicação estejam instalados no host para funcionar (ver figura 8). O propósito de um container é possuir uma aplicação e todas as suas configurações e dependências em um único local, semelhante a um pacote de aplicação. Por causa disso, a quantidade de recursos utilizados por cada aplicação é minimizada e, por ter uma quantidade menor de espaço ocupado, a portabilidade do container é melhor em comparação às *VMs* de tipo 1 [48].

Figura 8 - Tipos de virtualização



Fonte: [48]

Um estudo entre os dois modelos foi realizado pela divisão de pesquisa da *IBM* [49] com a finalidade de verificar as suas performances. Para realizar os estudos, utilizaram-se ferramentas de benchmark (termo se refere à comparação de desempenho de hardware) para medir processamento (*PXZ* – realiza compressão de dados), computação de alta performance (*Linpack* – realiza cálculos de equações lineares), acesso e utilização de memória (*Stream* – utilizar memória e *RandomAccess* – acesso a memória), transferência e latência de dados, entre outras métricas utilizadas para comparar desempenho de rede e armazenamento de dados. Os testes, que foram executados sobre o mesmo ambiente, apresentaram resultados favoráveis para virtualização via containers.

Essas características tornam a virtualização por container uma escolha interessante para microsserviços. O pouco uso de espaço em disco permite que cada aplicação consuma menos espaço e consequentemente, reduzir gastos com recursos em nuvens públicas.

5.3. Comportamento dos gastos na Nuvem

Transferir o risco de investimento em infraestrutura para o negócio tem se tornado cada vez mais comum em função da relação custo benefício que uso da nuvem pode trazer. Essa premissa é maior em relação às pequenas empresas: com a possibilidade de evitar o gasto com infraestrutura adequada para o progresso do

negócio, investir neste tipo de ambiente auxilia a controlar os gastos e poder expandir o negócio. O desafio é entender que tipo de nuvem é vantajosa para as pequenas empresas e o capital necessário para utilizar o ambiente.

Figura 9 - Comportamento de precificação das clouds.

Fonte: [28]

Por meio da figura 9 acima, fica mais claro para uma startup como funciona o comportamento de cobrança pelo uso dos recursos em cada tipo de nuvem:

- Na região esquerda da figura, que representa a cloud pública, é possível verificar que os custos por armazenamento, processamento e tráfego de informação são repassados para o cliente conforme a intensidade do uso;
- Na região mais central, representando o tipo privado, todo o custo de armazenamento, processamento e tráfego de informação é fixado em um valor único, não importando o quão baixo ou alto é o uso pelos recursos;
- Na região direita da figura, a cloud híbrida, existe um valor mínimo de recursos alocados e custos estabelecidos, porém, se houver um aumento no fluxo de informações, há uma expansão deles para atender a demanda e o custo acrescido de acordo com o uso.

5.4. Focando na nuvem pública

Startups, como já mencionado no capítulo introdutório deste estudo, não possuem recursos monetários suficientes para arcar com a infraestrutura de ambientes e isso também reflete no investimento em clouds.

O estudo das características das clouds mostrou que as do tipo privado oferecem um ambiente mais aberto para adaptar políticas de segurança e oferecem um custo fixo para que haja investimento. Inicialmente, essa seria a escolha mais adequada para as startups, no entanto, existem duas situações para o qual ambientes em clouds privadas tem desvantagens:

- Quando os serviços são colocados em ambientes que possuam armazenamento, tráfego e processamento maior do que o necessário, ou seja, há um excesso desses recursos e desperdício de capital investido;
- Quando é necessário de mais recursos do que os já adquiridos, e nesse caso, a comunicação com os usuários é mais demorada (por vezes deixando indisponível) e a entrega dos serviços com qualidade fica comprometida.

Com limitações barreiras em relação à expansão e redução de recursos, a implantação de microsserviços (e event-driven) em clouds privadas é mais complexa: há uma carga maior de fluxo de informação [18], visto que muitos processos tomam decisões automaticamente e as melhorias dos serviços acontecem constantemente requerem maior armazenamento e processamento. Portanto, o uso de clouds privadas como ambiente de infraestrutura para estas arquiteturas não é a melhor indicação para as startups.

Apesar de ter alguns empecilhos de segurança, a capacidade de expandir ou limitar o uso de recursos faz da nuvem pública o ponto chave para a implantação de serviços por causa da liberdade de gerenciamento de recursos aos clientes, permitindo-os mensurar o quanto gastam e o quanto escalam através do processamento e armazenamento de informações.

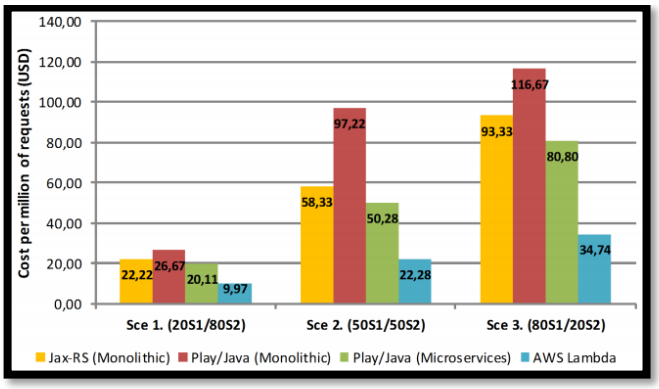
Microsserviço construídos sobre containers aproveitam dessa característica para realizar suas tarefas de forma eficaz e eficiente utilizando apenas o espaço necessário na cloud. Até em nuvens hibridas, a preocupação com a pública em relação ao custo benefício é maior no aspecto dos recursos.

Um dos exemplos de como a nuvem pública minimiza o custo de investimento na implantação de microsserviços pode ser visto na figura 10, onde há quatro cenários de implantação usados na *AWS* (nuvem pública da *Amazon*):

- O ambiente monolítico amarelo não possui framework;
- O ambiente monolítico vermelho possui um framework;
- O ambiente em microsserviço verde utiliza framework;
- O ambiente em microsserviço azul utiliza o AWS Lambda (*Serverless*).

O estudo é realizado com apenas dois tipos de serviços simples: o primeiro realizava uma consulta para saber o preço de pagamento e o segundo retornava os planos de pagamento existentes. Os testes mostraram que o desempenho de microsserviços é superior aos monolíticos em ambientes de nuvens.

Figura 10 – Custo de investimento na nuvem: Microsserviço x Monolítico



Fonte: [18]

O resultado mais expressivo dos testes foi o impacto de uso de microsserviços (utilizando o modelo de serviço *Serverless*) e a redução (até 77% de acordo com o autor do estudo) nos custos de infraestrutura, com ganho em performance e redução de respostas nas requisições em relação ao monolítico [18]. Isso mostra o quanto a arquitetura de microsserviço consegue usufruir melhor dos recursos da nuvem.

Um modelo chamado *CostHat*, proposto por Leitner, Cito e Stöckli [2016], pretende ir mais a fundo na questão de arquitetar os serviços: ele deseja criar um mecanismo para medir o custo do serviço mesmo antes de ser implantado na nuvem. Através de algumas métricas, como chamadas de *API* e utilização de *CPU*, o modelo pretende apresentar o custo do código no momento da implementação. Mesmo que

ainda não exista uma aplicação que realize tal função atualmente, este modelo pode servir de base para, em um futuro próximo, criar uma ferramenta que melhore a construção dos serviços, permitindo que desenvolvedores tenham consciência de custo de código por meio da criação de cenários para testar os custos das comunicações e, consequentemente, auxiliar a aprimorar a arquitetura entre os serviços em nuvem.

6. Conclusão e Trabalhos futuros

A utilização da nuvem para entrega de serviços é cada vez mais requisitada por ser base das novas formas de arquitetar sistemas. Os padrões de microsserviços e event-driven estão em processo de amadurecimento. No entanto, pode-se constatar que eles estão se tornando padrões bem aceitos à medida que mais casos de estudos e testes de desempenho comprovam sua relevância para uma era onde novos negócios e nichos de mercados surgem, e a demanda para suprir as necessidades desses nichos cresce de forma exacerbadamente alta.

O uso de microsserviço com event-driven consegue trazer flexibilidade e independência aos serviços e suas bases de dados, o que mostra ser uma escolha adequada para o atual cenário de negócios. A infraestrutura de clouds públicas mantem o ambiente propicio para que as soluções com alto desempenho.

A união entre essas tecnologias impacta, em diversos níveis, o modo como as empresas se comunicam internamente e externamente, seja por alterações organizacionais em relação a estruturar equipes para construção de soluções para o mercado que atuam, ou por causa do aprimoramento e dinâmica que essas arquiteturas trazem às aplicações, proporcionando alto grau de desempenho e competitividade as empresas emergentes.

Microsserviços, *Event-Driven* e *Cloud* podem ser considerados uma tríade de como as soluções arquiteturais de aplicações receberão adesões até pelas corporações menos visionárias, para não perderem clientes. *Amazon* e *Netflix*, alguns dos pioneiros em utilizar alguma dessas combinações, se tornaram espelho para diversas outras que aderiram a essas arquiteturas, e preenchem uma lista cada vez maior de marcas de sucesso pela qualidade da entrega de seus produtos e serviços.

6.1. Trabalhos Futuros

É notável perceber como uma grande quantidade de serviços oferecidos pelos provedores de nuvem está aumentando, com o intuito de manter seus clientes satisfeitos por escolherem seu ambiente, e para implementarem diversas soluções. Serverless é um desses serviços que pode trazer transformações de como os microsserviços serão construídos, pois evita que equipes se preocupem com o

provisionamento de máquinas virtuais ou containers, e se inquietando ainda mais com o código. Esse fato implica dizer de que *Serverless* pode auxiliar a construir aplicações, e se preocupar apenas com as regras de negócio.

Por outro lado, como foi apresentado, ferramentas que conseguem medir o custo financeiro de código, como o *CostHat*, podem surgir para garantir que o código esteja melhor custo benefício pela empresa que pretende utilizar esta tecnologia.

Por serem tecnologias novas, podem não ter um nível de maturidade ou uma implementação robusta para serem adotadas atualmente; porém é possível verificar que elas podem impactar na construção de soluções corporativas. Por esse motivo, há uma possibilidade de trabalhos futuros relacionando o impacto econômico/arquitetural que elas podem trazer para as companhias que resolverem adotá-las para aprimorar a entrega de bens e serviços.

7. Bibliografia

[1] – SHET ,Shivshankar. “Limitations of a Monolithic application and need for adapting Micro services Architecture”. Disponível em: <https://www.javacodegeeks.com/2016/05/limitations-monolithic-application-need-adapting-micro-services-architecture.html>. Acessado em 21 de setembro de 2017

[2] – Endeavor Brasil. “Como fazer sua empresa ser maior que você”. Disponível em: <https://endeavor.org.br/negocio-escalavel/>. Acessado em 21 de setembro de 2017

[3] – EVANS, Josh. “Mastering Chaos - A Netflix Guide to Microservices”. Disponível em: <https://www.infoq.com/presentations/netflix-chaos-microservices/>. Acessado em 21 de setembro de 2017

[4] – MACHADO, Marcos Giacometti. “Micro Serviços: Qual a diferença para a Arquitetura Monolítica?”. Disponível em: <https://www.opus-software.com.br/microservicos-diferenca-arquietura-monoliticas/>. Acessado em 23 de setembro de 2017

[5] – HEWITT, Eden. Java SOA Cookbook. 1ª ed. O’Reilly Media, 2009

[6] – RICHARDS, Mark. Microservices vs Service-Oriented Architecture.1ª ed. O’Reilly Media, 2015

[7] – LEWIS, James; FOWLER, Martin. “Microservice: a definition of this new architectural term”. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acessado em: 20 de setembro de 2017

[8] – NADAREISHVILI, Irakli; MITRA, Ronnie; MCLARY, Matt; AMUNDSEN Mike. Microservice Architecture. 1ª Ed. O’Reilly Media, 2016

[9] – VERONA, Joakim. Practical DevOps. 1ª Ed. Packt Publishing, 2016

[10] – RICHARDSON, Chris. “Introduction to Microservices”. Disponível em: <https://www.nginx.com/blog/introduction-to-microservices/>. Acessado em: 26 de setembro.

[11] – PALLADINO, Marco. “Microservices & API Gateways, Part 1: Why an API Gateway?”. Disponível em: <https://www.nginx.com/blog/microservices-api-gateways-part-1-why-an-api-gateway/>. Acessado em: 28 de setembro

[12] – NEWMAN, Sam. Building Microservices.1ª ed. O’Reilly Media, 2015

[13] – JACOBSON, Daniel; BRAIL, Greg; WOODS, Dan. APIs – A Strategy Guide. 1ª ed. O'Reilly Media, 2011

[14] – Art Anthony. "Tracking the Growth of the API Economy". Disponível em: <https://nordicapis.com/tracking-the-growth-of-the-api-economy/>. Acessado em 04 de outubro de 2017.

[15] – Amazon. "O que significa integração contínua?". Disponível em: <https://aws.amazon.com/pt/devops/continuous-integration/>. Acessado em 02 de outubro de 2017.

[16] – ELLINGWOOD, Justin. "An Introduction to Continuous Integration, Delivery, and Deployment". Disponível em: <https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment>. Acessado em: 04 de outubro de 2017

[17] – BIRIBA, Rafael. "Blue-Green Deployment: entrega contínua e rollback imediato". Disponível em: <https://www.infoq.com/br/presentations/blue-green-deployment>. Acessado em: 05 de outubro de 2017.

[18] – M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures" 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, 2016, pp. 179-182.

[19] – MICHELSON, Brenda M. Event-driven architecture overview. Patricia Seybold Group, v. 2, 2006.

[20] – TAYLOR, Hugh, YOCHEM, Angela, PHILLIPS, Les, MARTINEZ, Frank. How SOA Enables the Real-Time Enterprise. 1ª Ed. 2010

[21] – GRAY, Alexander; JEFFERY, Keith G.; SHAO, Jianhua (Ed.). Sharing Data, Information and Knowledge: 25th British National Conference on Databases, BNCOD 25, Cardiff, UK, July 7-10, 2008, Proceedings. Springer, 2008.

[22] – FOWLER, Martin. "What do you mean by "Event-Driven"?" Disponível em: <https://martinfowler.com/articles/201701-event-driven.html>. Acessado em: 26 de outubro de 2017.

[23] – LUBLINSKY, Boris. "Orchestration vs. Choreography: Debate Over Definitions". Disponível em: <https://www.infoq.com/news/2008/09/Orchestration>. Acessado em:

27 de outubro de 2017

[24] – RICHARDS, Mark. Software Architecture Pattern. 1ª ed. O'Reilly Media, 2015. pp 12-19

[25] – NARKHEDE, Neha. "Event sourcing, CQRS, stream processing and Apache Kafka: What's the connection?". Disponível em: <https://www.confluent.io/blog/event-sourcing-cqrs-stream-processing-apache-kafka-whats-connection/>. Acessado em: 26 de outubro de 2017

[26] – MELL, Peter et al. The NIST definition of cloud computing. 2011.

[27] – HOLF, Todd. "Cloud AWS Infrastructure Vs. Physical Infrastructure". Disponível em: <http://highscalability.com/blog/2010/7/8/cloud-aws-infrastructure-vs-physical-infrastructure.html>. Acessado em: 02 de novembro de 2017

[28] – SORNOSO, Eric. "The Difference Between Public and Private Cloud". Disponível em: <https://www.business2community.com/cloud-computing/difference-public-private-cloud-0735598#ZoqLJJlUAowfg8MA.97>. Acessado em: 01 de novembro de 2017

[29] – GOASDUFF, Laurence; PETTEY, Christy. "Gartner Says Worldwide Public Cloud Services Market to Grow 18 Percent in 2017". Disponível em: <https://www.gartner.com/newsroom/id/3616417>. Acessado em: 01 de novembro de 2017

[30] – ZIMARA, Sabrina. "The Five Essential Characteristics of Cloud Computing". Disponível em: <http://erpbloggers.com/2013/07/the-five-essential-characteristics-of-cloud-computing/>. Acessado em: 02 de novembro de 2017

[31] – TAURION, Cezar. "Entendendo o modelo Multi-tenancy". Disponível em: <https://imasters.com.br/artigo/19067/cloud/entendendo-o-modelo-multi-tenancy/?trace=1519021197&source=single>. Acessado em: 02 de novembro de 2017

[32] – SCHOUTEN, Edwin. "Cloud computing defined: Characteristics & service levels". Disponível em: <https://www.ibm.com/blogs/cloud-computing/2014/01/cloud-computing-defined-characteristics-service-levels/>. Acessado em: 02 de novembro de 2017

[33] – Salesforce Brasil. "O que é Cloud Computing? Entenda a sua Definição e

Importância". Disponível em: <https://www.salesforce.com/br/blog/2016/02/o-que-e-cloud-computing.html>. Acessado em: 03 de novembro de 2017

[34] – VELTE, Anthony T. et al. Cloud computing: a practical approach. New York: McGraw-Hill, 2010.

[35] – FORK, Michael J. "What is infrastructure as a service (IaaS)?". Disponível em: <https://www.ibm.com/blogs/cloud-computing/2014/02/what-is-infrastructure-as-a-service-iaas/>. Acessado em: 04 de novembro de 2017

[36] – Microsoft Azure. "O que é PaaS? Plataforma como serviço". Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-paas/>. Acessado em: 05 de novembro de 2017

[37] – PANETTA, Kasey. "Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017". Disponível em: <https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/>. Acessado em: 06 de novembro de 2017

[38] – P. Castro, V. Ishakian, V. Muthusamy and A. Slominski, "Serverless Programming (Function as a Service)," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2658-2659.

[39] – ROBERTS, Mike. "Serverless Architectures". Disponível em: <https://martinfowler.com/articles/serverless.html>. Acessado em 07 de novembro de 2017.

[40] – HARVEY, Cynthia. "Private vs Public Cloud Computing". Disponível em: <https://www.datamation.com/cloud-computing/private-vs-public-cloud.html>.
Acessado em: 08 de novembro de 2017

[41] - GOYAL, Sumit. Public vs private vs hybrid vs community-cloud computing: a critical review. International Journal of Computer Network and Information Security, v. 6, n. 3, p. 20, 2014.

[42] – LITTLE, Mark. "A diferença entre SOA e microserviços". Disponível em: <https://www.infoq.com/br/news/2017/08/soaandmicroservices>. Acessado em 24 de setembro de 2017

[43] – SHINDER, Thomas. "What Does Shared Responsibility in the Cloud Mean?". Disponível em: <https://blogs.msdn.microsoft.com/azuresecurity/2016/04/18/what-

does-shared-responsibility-in-the-cloud-mean/>. Acessado em 02 de novembro de 2017

[44] – RICHARDSON, Chris. “Event-Driven Data Management for Microservices”. Disponível em: <https://www.nginx.com/blog/event-driven-data-management-microservices/>. Acessado em 14 de novembro de 2017

[45] – MCDONALD, Carol. “Event Driven Microservices Patterns” Disponível em: <https://mapr.com/blog/event-driven-microservices-patterns/>. Acessado em 15 de novembro de 2017

[46] – RICHARDSON, Chris. “Microservices + Events + Docker = A Perfect Trio - Wild Card Track”. Produção de Dockercon. Disponível em: <https://www.youtube.com/watch?v=pD0rEtEEwck> (43 min). Acessado em 18 de novembro 2017

[47] – KLEYMAN, Bill. “Hypervisor 101: Understanding the Virtualization Market”. Disponível em: <http://www.datacenterknowledge.com/archives/2012/08/01/hypervisor-101-a-look-hypervisor-market>. Acessado em 16 de novembro de 2017

[48] – MARCELO, João. “Container, o novo passo para a virtualização”. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/mp234?lang=en>. Acessado em 19 de novembro de 2017

[49] – FELTER, Wes et al. An updated performance comparison of virtual machines and linux containers. In: Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On. IEEE, 2015. p. 171-172

[50] – LEITNER, Philipp; CITO, Jürgen; STÖCKLI, Emanuel. Modelling and managing deployment costs of microservice-based cloud applications. In: Proceedings of the 9th International Conference on Utility and Cloud Computing. ACM, 2016. p. 165-174.

[51] – DAISYME, Peter. “For Startups, Automation is a Key Ingredient to Success”. Disponível em: <https://www.startupgrind.com/blog/for-startups-automation-is-a-key-ingredient-to-success/>. Acessado em 16 de dezembro de 2017.

[52] – ALL, Ann. “Process Automation a Key to Business Value”. Disponível em: <http://www.enterpriseappstoday.com/management-software/process-automation-a-

key-to-business-value.html>. Acessado em 16 de dezembro de 2017.

[53] – BONHAM, Andrew. “Microservices — When to React Vs. Orchestrate”. Disponível em: <https://medium.com/capital-one-developers/microservices-when-to-react-vs-orchestrate-c6b18308a14c>. Acessado em 09 de setembro de 2017.