

MBA
USP
ESALQ

MICROSERVIÇOS

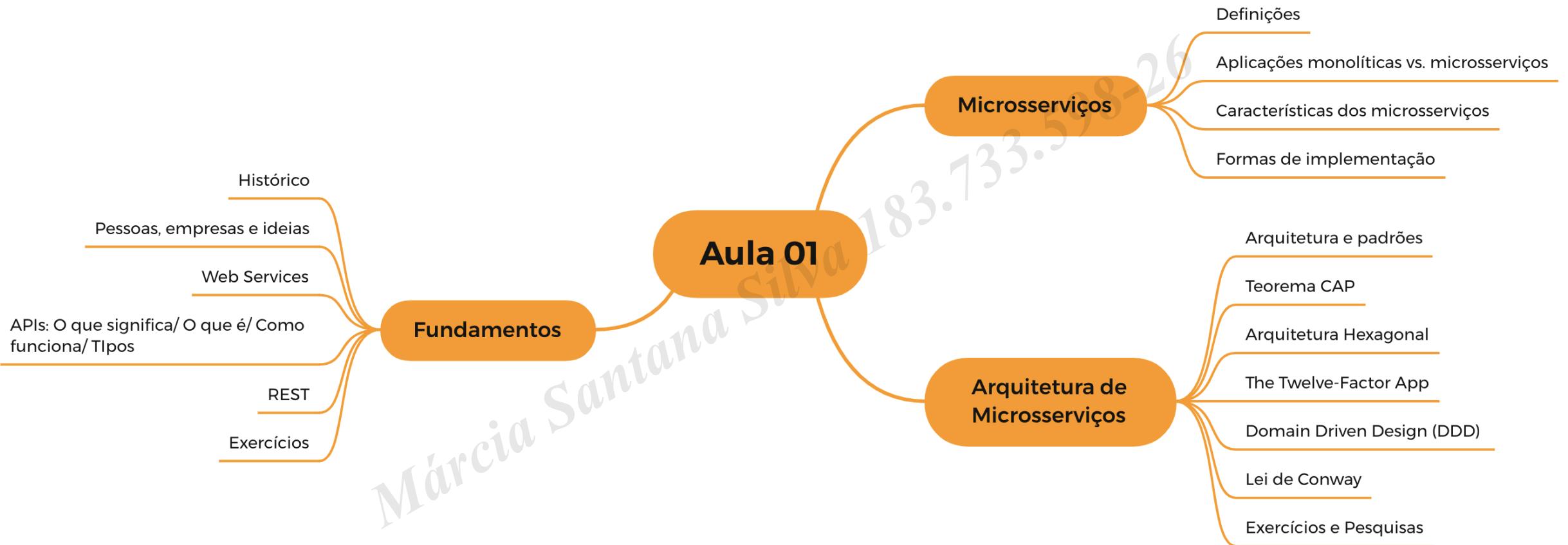
Prof. José Maria Cesário Jr.

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.

Proibida a reprodução, total ou parcial, sem autorização. Lei nº 9610/98

AGENDA MICROSERVIÇOS

Data	Conteúdo
01/02/2024	Fundamentos de microserviços Web Services APIs: Tipos, benefícios, segurança, casos de uso REST e RESTful APIs Microservices: características, aplicações monolíticas, padrões Twelve Factors apps Arquitetura Hexagonal Exercícios



ÍCONES

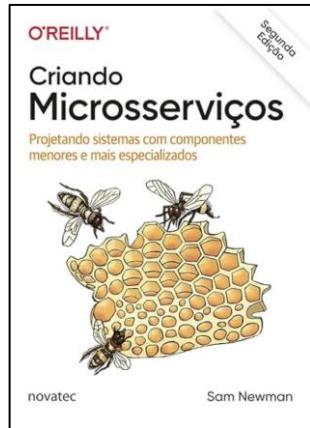


**DEFINIÇÃO OU
PONTO DE ATENÇÃO**



**EXERCÍCIO OU
ATIVIDADE PRÁTICA**

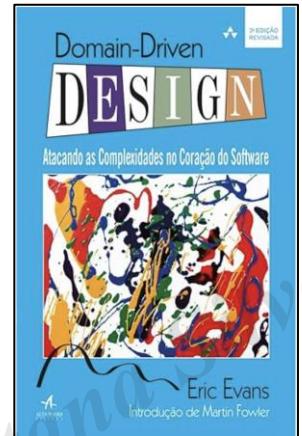
LIVROS



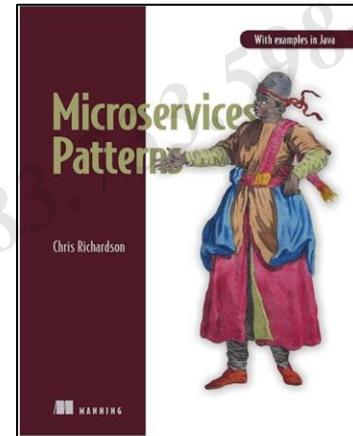
**Criando
Microsserviços
Sam Newman
2ª Edição 2017**



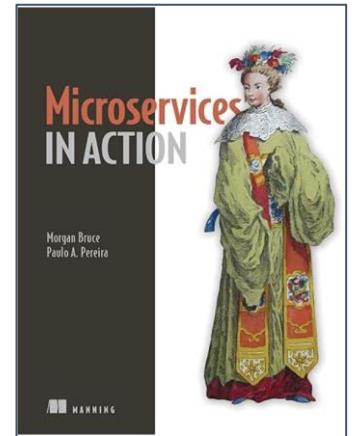
**Microsserviços Prontos
Para a Produção
Susan J. Fowler
1ª Edição 2017**



**Domain-driven design
Eric Evans
1ª Edição 2016**



**Microservices Patterns:
With Examples in Java
Chris Richardson
1ª Edição 2018**

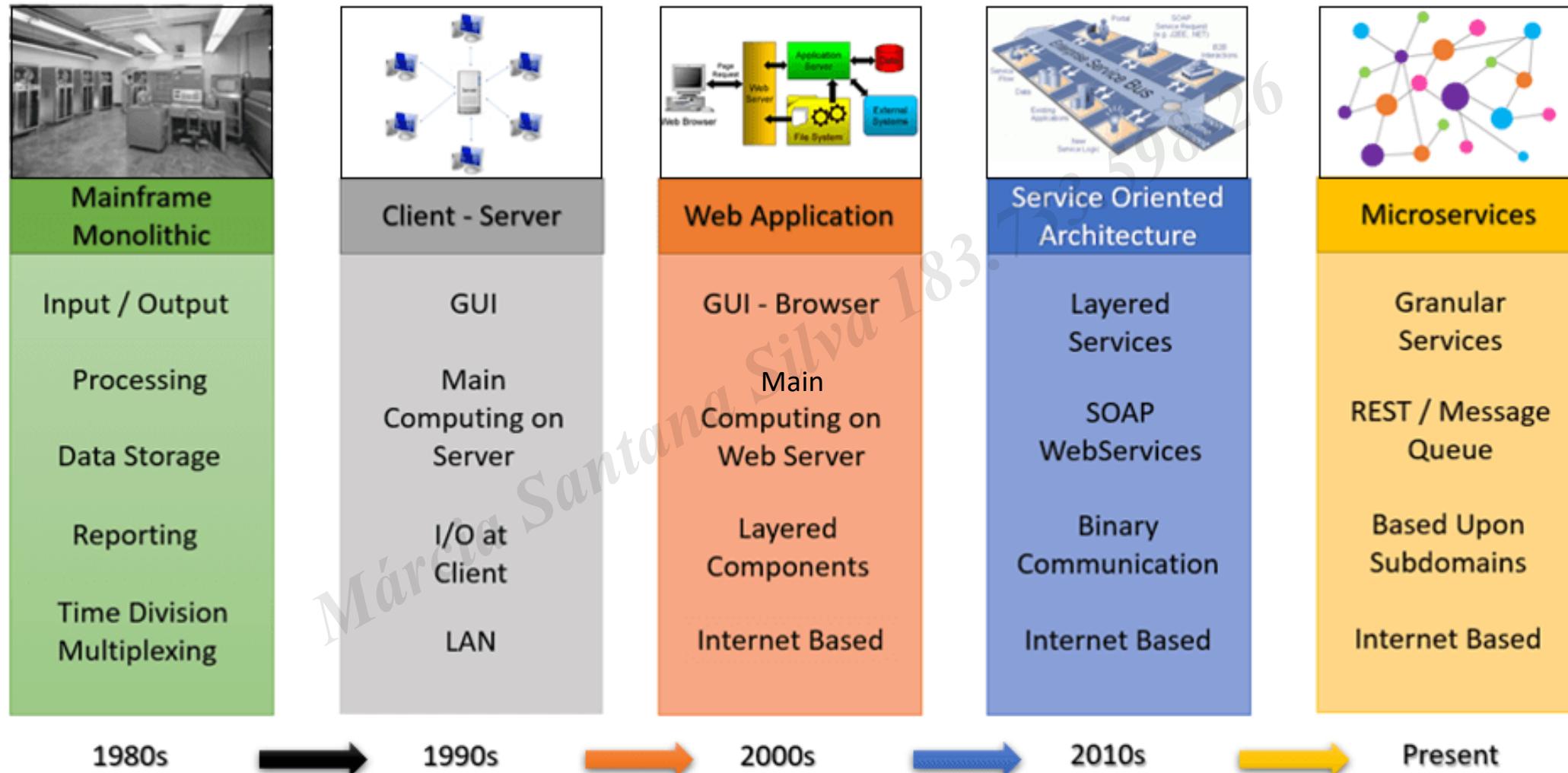


**Microservices in Action
Morgan Bruce
Paulo A Pereira
1ª Edição 2018**

FUNDAMENTOS

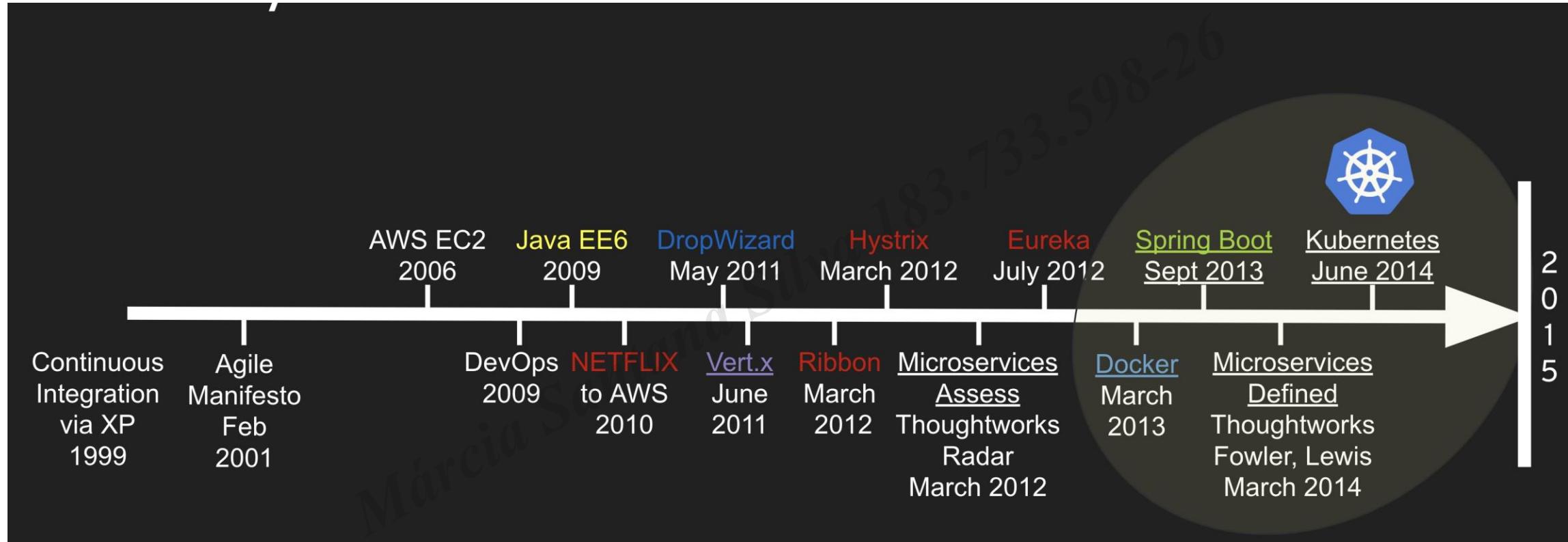
Parte 1

HISTÓRICO



Fonte: <https://www.dotnetcurry.com/microsoft-azure/microservices-architecture>

HISTÓRICO



Fonte: https://docs.google.com/presentation/d/1QeDOfgnJO5vCftBFsA8iU3xP1FUTRXYfeVCuY_ZJTdo/edit?usp=sharing

PESSOAS E IDEIAS



Fred George
ThoughtWorks
Bayesian Principles
2004



Peter Rodgers
Micro-Web-Services
2005



Alistair Cockburn
Hexagonal Architecture
2005



James Lewis
ThoughtWorks
Microservices
2011



Martin Fowler
ThoughtWorks
Microservices
2011



James Lewis
Case study "Microservices - Java, the Unix Way"
33rd Degree conference in Krakow
2012

Fonte: Fotos adaptadas Wikipedia

O TERMO MICROSERVIÇOS

"Microservices" - yet another new term on the crowded streets of software architecture. Although our natural inclination is to pass such things by with a contemptuous glance, this bit of terminology describes a style of software systems that we are finding more and more appealing. We've seen many projects use this style in the last few years, and results so far have been positive, so much so that for many of our colleagues this is becoming the default style for building enterprise applications. Sadly, however, there's not much information that outlines what the microservice style is and how to do it.

In short, the microservice architectural style [1] is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

The term "microservice" was discussed at a workshop of software architects near Venice in May, 2011 to describe what the participants saw as a common architectural style that many of them had been recently exploring. In May 2012, the same group decided on "microservices" as the most appropriate name. James presented some of these ideas as a case study in March 2012 at 33rd Degree in Krakow in [Microservices - Java, the Unix Way](#) as did Fred George [about the same time](#). Adrian Cockcroft at Netflix, describing this approach as "fine grained SOA" was pioneering the style at web scale as were many of the others mentioned in this article - Joe Walnes, Daniel Terhorst-North, Evan Botcher and Graham Tackley.

<https://martinfowler.com/articles/microservices.html>

EMPRESAS E IDEIAS



A Conversation with Werner Vogels

<https://queue.acm.org/detail.cfm?id=1142065>



<https://www.thoughtworks.com/insights/topic/microservices>



Redesigning the Netflix API

<https://netflixtechblog.com/redesigning-the-netflix-api-db5a7221fcff>



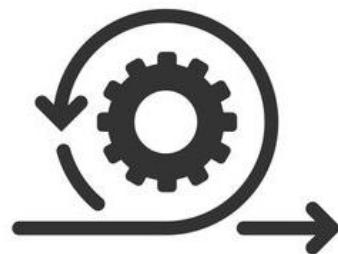
<https://cloudplatform.googleblog.com/2015/01/in-coming-weeks-we-will-be-publishing.html>

Fonte: Figuras adaptadas Wikipedia

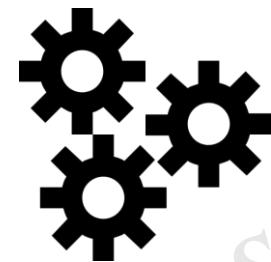
PREPARAÇÃO PARA MICROSSERVIÇOS



Arquiteturas de microserviços combinam vários conceitos, tecnologias e padrões:



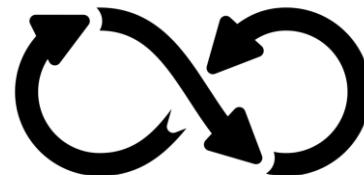
Desenvolvimento de
Software Ágil



Arquiteturas Orientadas à
Serviço



API-first Design



CI/CD

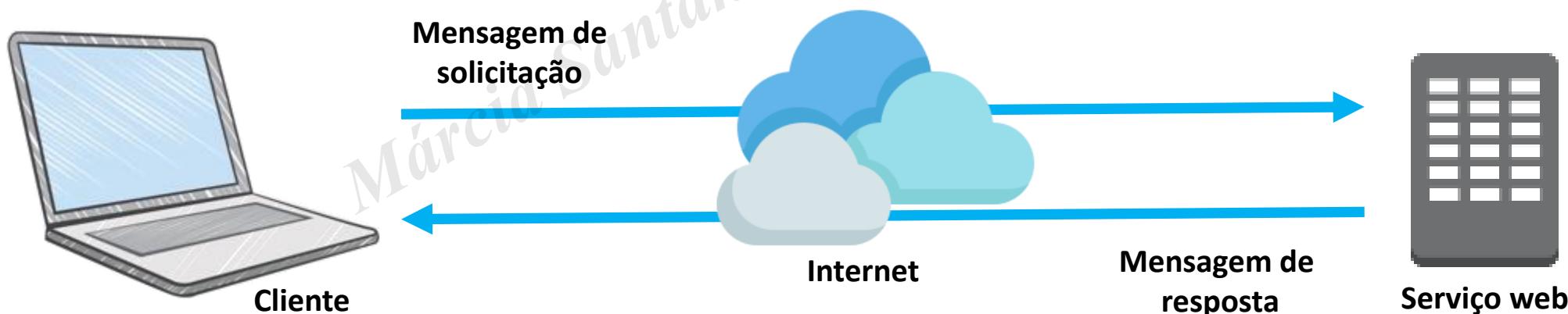


Design Patterns
12-Factor Apps

Fonte: <https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/microservices-on-aws.html>

O QUE SÃO SERVIÇOS WEB?

Um **serviço web** é qualquer software disponibilizado pela Internet que usa um **formato padronizado**, como eXtensible Markup Language (XML) ou JavaScript Object Notation (JSON), para a solicitação e resposta de uma interação através de ***Application Programming Interface (API)***.



Fonte: AWS Academy Foundations, Módulo 1

O QUE SIGNIFICA API?



- API significa Application Programming Interface (Interface de Programação de Aplicação). No contexto de APIs, a palavra Aplicação refere-se a qualquer software com uma função distinta.
- A interface pode ser pensada como um contrato de serviço entre duas aplicações. Esse contrato define como as duas se comunicam usando solicitações e respostas.
- A documentação de suas respectivas APIs contém informações sobre como os desenvolvedores devem estruturar essas solicitações e respostas.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

O QUE É UMA API ?

- APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.
- Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários.
- A aplicação para a previsão do tempo em seu telefone “fala” com esse sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

COMO AS APIs FUNCIONAM?

- A arquitetura da API geralmente é explicada em termos de cliente e servidor.
- A aplicação que envia a solicitação é chamada de cliente e a aplicação que envia a resposta é chamada de servidor.
- As operações podem ser síncronas ou assíncronas.
- Então, no exemplo do clima, o banco de dados meteorológico do instituto é o servidor e o aplicativo móvel é o cliente.

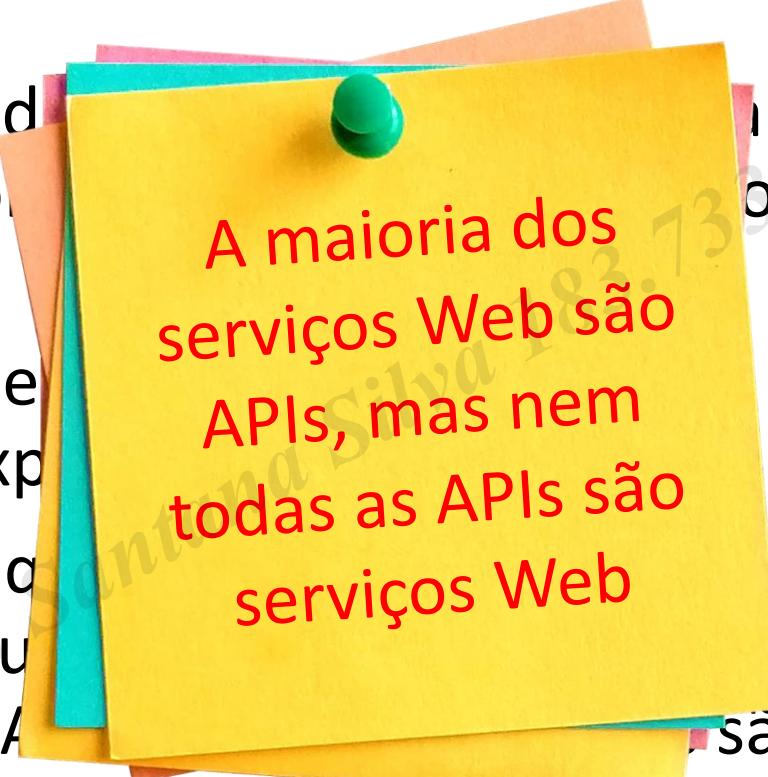
Fonte: <https://aws.amazon.com/pt/what-is/api/>

O QUE É API WEB?

- Uma API Web ou API de serviço da Web é uma interface de processamento de aplicações entre um servidor da Web e um navegador da Web.
- A API REST é um tipo especial de API Web que usa o estilo de arquitetura padrão explicado acima.
- Os diferentes termos que abrangem as APIs, como API Java ou APIs de serviço, existem porque, historicamente, as APIs foram criadas antes da World Wide Web. As APIs Web modernas são APIs REST e os termos podem ser usados de forma intercambiável.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

O QUE É API WEB?

- Uma API Web ou API é uma interface de processamento de aplicativos que pode ser usada por um navegador da Web.
 - A API REST é um tipo específico de arquitetura padrão exposta por APIs.
 - Os diferentes termos de API e serviço, existem porque existem muitas maneiras de expor serviços da World Wide Web. As definições desses termos podem ser usados de forma intercambiável.
- 
- A maioria dos serviços Web são APIs, mas nem todas as APIs são serviços Web

Fonte: <https://aws.amazon.com/pt/what-is/api/>

O QUE SÃO INTEGRAÇÕES DE API?

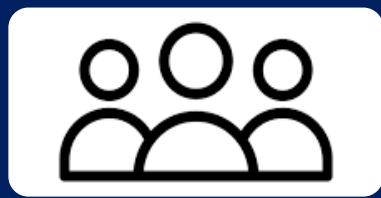
- As integrações de API são componentes de software que atualizam automaticamente os dados entre clientes e servidores.
- Alguns exemplos de integrações de API são quando os dados automáticos são sincronizados com a nuvem por meio da galeria de imagens do seu telefone ou a data e a hora são sincronizadas automaticamente no seu laptop quando você viaja para um local com outro fuso horário.

ACESSIBILIDADE DE APIs



APIs privadas

- Elas são internas a uma empresa e são usadas apenas para conectar sistemas e dados dentro da empresa.



APIs públicas

- Estas são abertas ao público e podem ser usadas por qualquer pessoa. Pode ou não haver alguma autorização e custo associado a esses tipos de APIs.



APIs de parceiros

- Estas são acessíveis apenas por desenvolvedores externos autorizados para auxiliar as parcerias entre empresas.



APIs compostas

- Estas combinam duas ou mais APIs distintas para atender a requisitos ou comportamentos complexos do sistema.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

O QUE É UM ENDPOINT DE API E POR QUE ELE É IMPORTANTE?

- Os endpoints da API são os pontos de contato finais no sistema de comunicação da API. Estes incluem URLs de servidores, serviços e outros locais digitais específicos de onde as informações são enviadas e recebidas entre sistemas. Os endpoints da API são fundamentais para as empresas por dois motivos principais:



1. Segurança

Os endpoints da API tornam o sistema vulnerável a ataques. O monitoramento da API é crucial para impedir o uso indevido.

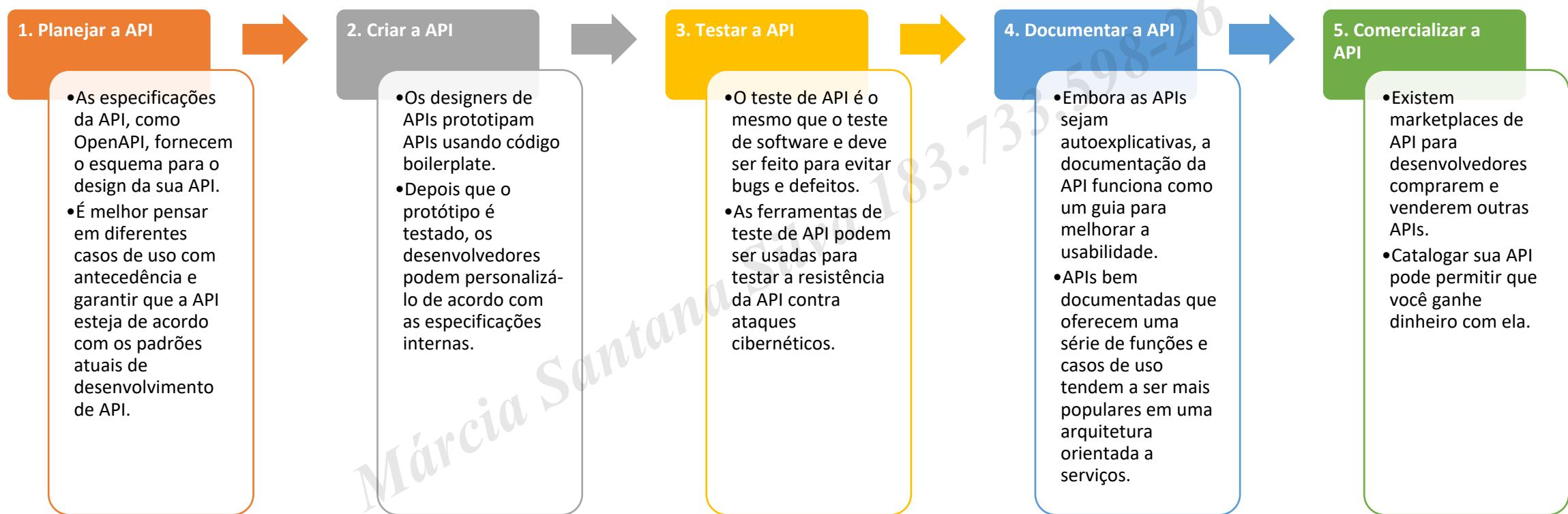


2. Performance

Os endpoints da API, especialmente os de alto tráfego, podem causar gargalos e afetar a performance do sistema.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

COMO CRIAR UMA API?



Fonte: <https://aws.amazon.com/pt/what-is/api/>

EXERCÍCIO: API



Acesse a API aberta do US National Weather Service e veja seu funcionamento e documentação. Teste os exemplos de chamadas de APIs no seu navegador e veja as respostas.



<https://api.weather.gov>

<https://www.weather.gov/documentation/services-web-api>

Exemplos de chamadas de API

<https://api.weather.gov/gridpoints/TOP/31,80/forecast>

<https://api.weather.gov/points/39.7456,-97.0892>

<https://api.weather.gov/alerts/active?area=NY>

PRINCIPAIS TIPOS DE APIs



- Principais tipos de APIs e seu funcionamento, dependendo de quando e por que elas foram criadas:

APIs REST	APIs RPC	APIs SOAP	GraphQL
<ul style="list-style-type: none">• Essas são as APIs mais populares e flexíveis encontradas na Web atualmente.• O cliente envia solicitações ao servidor como dados.• O servidor usa essa entrada do cliente para iniciar funções internas e retorna os dados de saída ao cliente.	<ul style="list-style-type: none">• Essas APIs são conhecidas como Remote Procedure Calls (Chamadas de Procedimento Remoto).• O cliente conclui uma função (ou um procedimento) no servidor e o servidor envia a saída de volta ao cliente.	<ul style="list-style-type: none">• Essas APIs usam o SOAP - Simple Object Access Protocol (Protocolo de Acesso a Objetos Simples).• Cliente e servidor trocam mensagens usando XML.• Esta é uma API menos flexível que era mais popular no passado.	<ul style="list-style-type: none">• GraphQL é uma especificação, uma linguagem de consulta de API e um conjunto de ferramentas. GraphQL opera em um único endpoint usando HTTP.• Ela prioriza fornecer aos clientes exatamente os dados que eles solicitam e nada mais.• Foi projetada para tornar as APIs rápidas, flexíveis e amigáveis ao desenvolvedor.• Como alternativa ao REST, o GraphQL oferece aos desenvolvedores de frontend a capacidade de consultar vários bancos de dados, microsserviços e APIs com um único endpoint de GraphQL.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

COMPARAÇÃO ENTRE OS PRINCIPAIS TIPOS DE APIs



	REST	RPC	SOAP	GraphQL
Fundamento	API design pattern	API design pattern	Protocolo para criação de APIs	Linguagem de consulta e runtime server-side para APIs
Características	<ul style="list-style-type: none"> Orientado a recursos Baseado em dados Flexível 	<ul style="list-style-type: none"> Orientado a ação Alto desempenho 	<ul style="list-style-type: none"> Padronizado Altamente seguro Foco em aplicações corporativas 	<ul style="list-style-type: none"> Endpoint único Requisições fortemente tipadas Buscas enxutas Auto-documentável
Formato de Dados	JSON/XML/YAML/HTML/plain text	JSON/XML/Thrift/Protobuf/FlatBuffers	XML	JSON
Curva de Aprendizado	Fácil	Fácil	Desafiadora	Média
Comunidade	Grande	Grande	Pequena	Em crescimento
Casos de Uso	<ul style="list-style-type: none"> Todos os tipos de apps web Apps Cloud Serviços de computação em nuvem 	<ul style="list-style-type: none"> Sistemas de microsserviços complexos Aplicações IoT 	<ul style="list-style-type: none"> Serviços financeiros CRM Gerenciamento de identidade Integração de sistemas legado 	<ul style="list-style-type: none"> Apps móveis de alto desempenho Sistemas complexos Arquiteturas baseadas em microsserviços

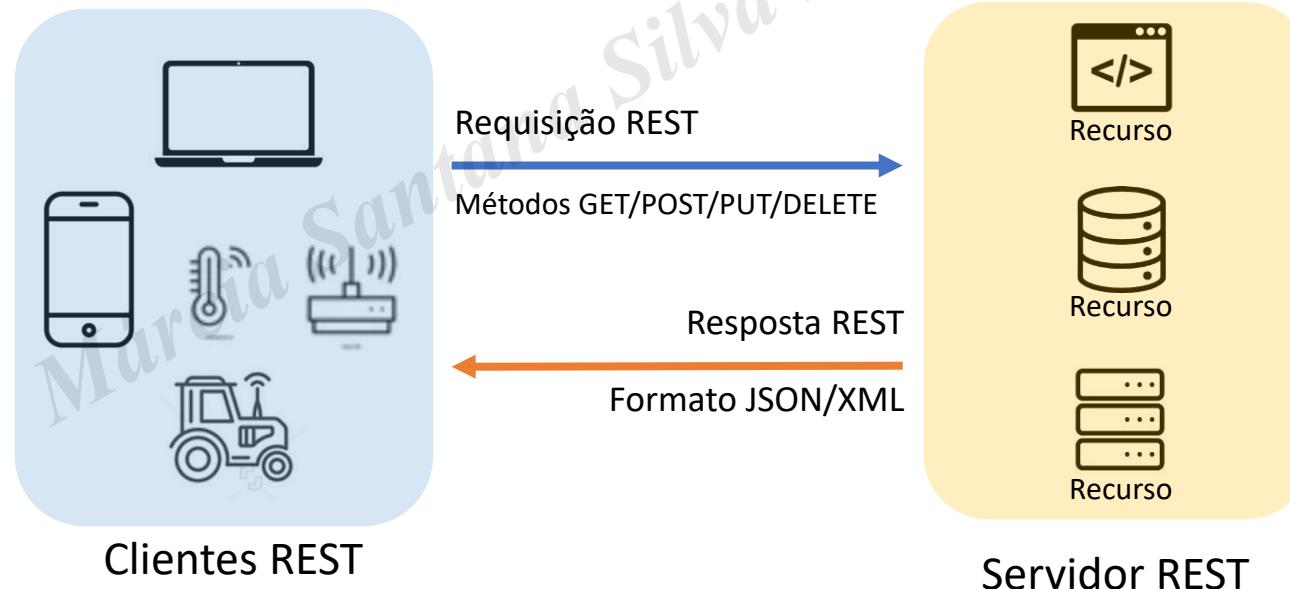
Fonte: Adaptado de <https://www.altexsoft.com/blog/rest-api-design/>

REST

- O modelo REST (REpresentational State Transfer) representa uma das possibilidades para a criação de web services, com utilização semântica dos métodos HTTP (GET, POST, PUT e DELETE)
- O criador do modelo REST foi [Roy Fielding](#), um dos principais autores da especificação HTTP e cofundador do projeto Apache HTTP Server.
- O termo REST foi apresentado pela primeira vez em 2000 na tese de doutorado de Roy Fielding [*Architectural Styles and the Design of Network-based Software Architectures*](#)

REST

- Uma requisição HTTP é equivalente a uma chamada de um método (operação) em um objeto (recurso) residente no servidor.
- Principais características de uma requisição REST:



Fonte: Adaptado de <https://www.altexsoft.com/blog/rest-api-design/>

O QUE SÃO APIs REST?

- A principal característica da API REST é a ausência de estado. A ausência de estado significa que os servidores não salvam dados do cliente entre as solicitações.
- As solicitações do cliente ao servidor são semelhantes aos URLs que você digita no navegador para visitar um site. A resposta do servidor corresponde a dados simples, sem a renderização gráfica típica de uma página da Web.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

QUAIS SÃO OS BENEFÍCIOS DAS APIs REST?

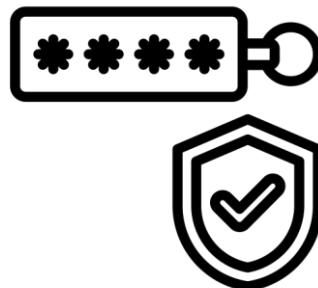
Integração	Inovação	Expansão	Facilidade de manutenção
<ul style="list-style-type: none">As APIs são usadas para integrar novas aplicações com sistemas de software existentes. Isso aumenta a velocidade de desenvolvimento porque cada funcionalidade não precisa ser escrita do zero.Você pode usar APIs para aproveitar o código existente.	<ul style="list-style-type: none">Setores inteiros podem mudar com a chegada de uma nova aplicação. As empresas precisam responder rapidamente e oferecer suporte à rápida implantação de serviços inovadores.Elas podem fazer isso fazendo alterações no nível da API sem precisar reescrever todo o código.	<ul style="list-style-type: none">As APIs apresentam uma oportunidade única para as empresas atenderem às necessidades de seus clientes em diferentes plataformas.Por exemplo, a API de mapas permite a integração de informações de mapas por meio de sites, Android, iOS etc.Qualquer empresa pode fornecer acesso semelhante aos seus respectivos bancos de dados internos usando APIs gratuitas ou pagas.	<ul style="list-style-type: none">A API atua como um gateway entre dois sistemas.Cada sistema é obrigado a fazer alterações internas para que a API não seja afetada.Dessa forma, qualquer alteração futura de código feita por uma parte não afetará a outra parte.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

COMO PROTEGER UMA API REST?



- Todas as APIs devem ser protegidas por meio de autenticação e monitoramento adequados. As duas principais maneiras de proteger APIs REST incluem:



1. Tokens de autenticação

Eles são usados para autorizar os usuários a fazer a chamada de API. Os tokens de autenticação verificam se os usuários são quem afirmam ser e se têm direitos de acesso para aquela chamada de API específica. Por exemplo, quando você faz login em seu servidor de e-mail, seu cliente de e-mail usa tokens de autenticação para acesso seguro.

2. Chaves de API

As chaves de API verificam o programa ou a aplicação que faz a chamada de API. Elas identificam a aplicação e garantem que ela tenha os direitos de acesso necessários para fazer a chamada de API específica. As chaves de API não são tão seguras quanto os tokens, mas permitem o monitoramento da API para coletar dados sobre o uso. Você pode ter notado uma longa sequência de caracteres e números no URL do seu navegador ao visitar diferentes sites. Essa string é uma chave de API que o site usa para fazer chamadas de API internas.

Fonte: <https://aws.amazon.com/pt/what-is/api/>

APIS REST VS. RESTFUL

- Uma API RESTfull segue ou implementa todas as exigências e restrições do estilo de arquitetura definido pelo criador do modelo REST (Roy Fielding) no processo de construção de um aplicação.
- As exigências para uma API ser RESTful são:
 1. Arquitetura cliente-servidor
 2. Comunicação stateless
 3. Cache
 4. Interface uniforme
 5. Sistema de camadas

EXERCÍCIO



- Prática de REST APIs com JSON-Server

MICROSERVIÇOS

Parte 2

O QUE SÃO MICROSERVIÇOS?



São uma combinação de abordagens arquitetônicas e organizacionais para o desenvolvimento de software em que as aplicações são compostas por serviços independentes que se comunicam por meio de APIs bem definidas.

As arquiteturas de microserviços facilitam a escalabilidade e agilizam o desenvolvimento de aplicativos, habilitando a inovação e acelerando o tempo de introdução de novos recursos e os ciclos de implantação no mercado.

Fonte: <https://aws.amazon.com/pt/microservices/>

O QUE SÃO MICROSERVIÇOS?

Microsserviços em poucas palavras

Por James Lewis e Martin Fowler

Publicado: February 23, 2015

O termo "Arquitetura de Microsserviços (Microservice Architecture)" surgiu nos últimos anos para descrever uma maneira específica de desenvolver software como suites de serviços com deploy independente. Embora não exista uma definição precisa desse estilo de arquitetura, há certas características comuns em relação à organização, à capacidade de negócios, ao deploy automatizado, à inteligência nos terminais e ao controle descentralizado de linguagens e de dados.

O que segue foi retirado de um artigo que apareceu originalmente no [website do Martin Fowler](#).

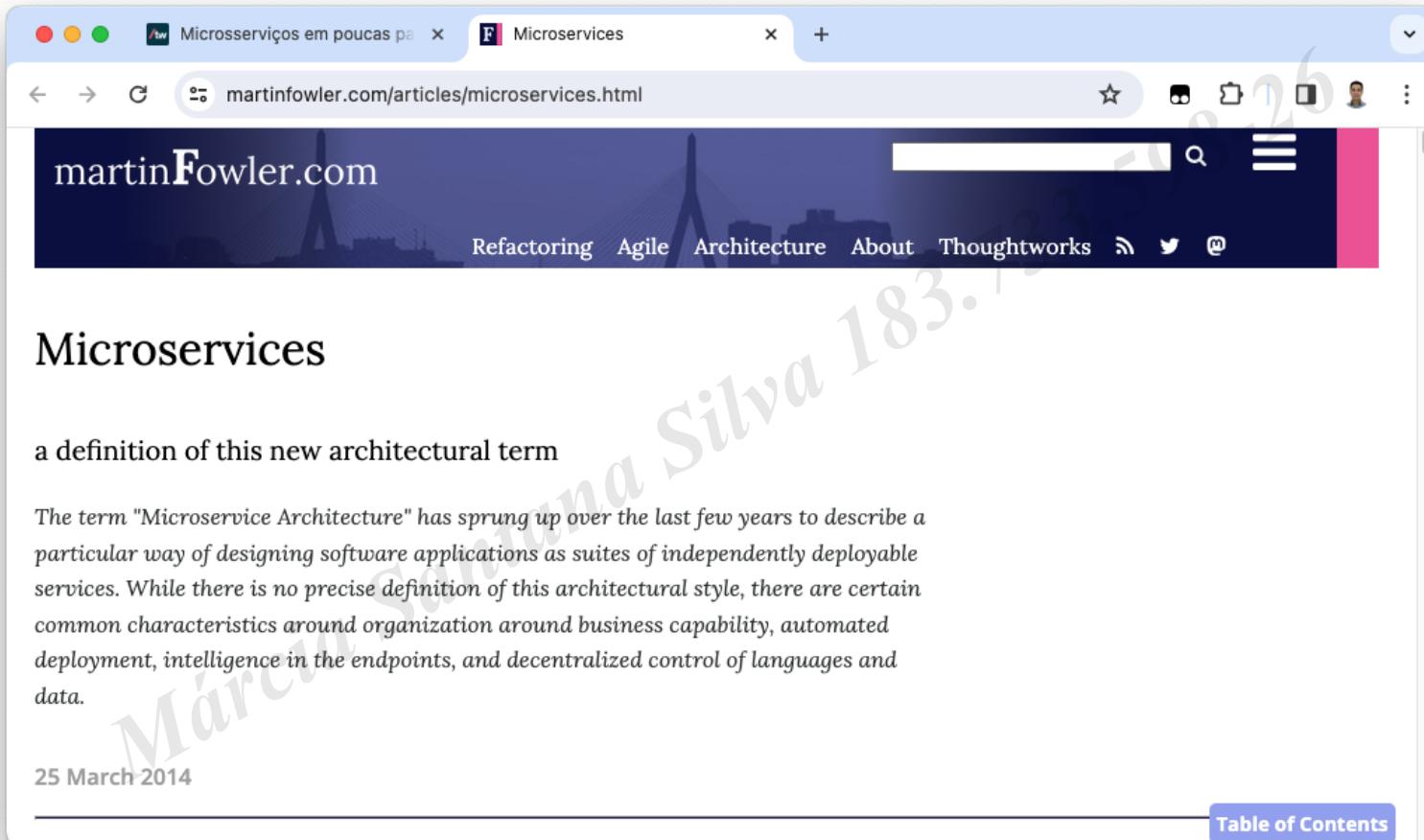
"Microsserviços" - mais um novo termo nas ruas lotadas da arquitetura de software. Embora a nossa inclinação natural seja olhar para essas coisas com um certo desprezo, a terminologia descreve um estilo de sistemas de software que temos achado cada vez mais atraente. Temos visto muitos projetos usando esse estilo nos últimos anos, e os resultados até agora têm sido positivos; tanto que, para muitos de nossos colegas, microsserviços vêm se tornando o estilo padrão para o desenvolvimento de aplicativos corporativos. Infelizmente, no entanto, não há muita informação que descreva o estilo de microsserviços e como aplicá-lo.

Resumindo, o estilo de arquitetura de microsserviços é uma abordagem que desenvolve um aplicativo único como uma suite de pequenos serviços, cada um executando seu próprio processo e se comunicando através de mecanismos leves, muitas vezes em uma API com recursos HTTP. Esses serviços são construídos em torno de capacidades de negócios e funcionam através de mecanismos de deploy independentes totalmente automatizados. Há o mínimo possível de gerenciamento centralizado desses serviços, que podem ser escritos em diferentes linguagens de programação e utilizam diferentes tecnologias de armazenamento de dados.

Para começar a explicar o estilo de microsserviços, é útil compará-lo com o estilo de aplicativo monolítico,

<https://www.thoughtworks.com/pt-br/insights/blog/microservices-nutshell>

O QUE SÃO MICROSERVIÇOS?



The screenshot shows a web browser window with two tabs open. The active tab is titled 'Microservices' and displays the URL martinfowler.com/articles/microservices.html. The page content is from martinFowler.com, featuring a dark header with the site's name and a navigation bar with links to Refactoring, Agile, Architecture, About, Thoughtworks, and social media icons. The main article title is 'Microservices', described as 'a definition of this new architectural term'. Below the title is a detailed paragraph explaining the concept of Microservice Architecture. At the bottom of the article, there is a date '25 March 2014' and a 'Table of Contents' button.

<https://martinfowler.com/articles/microservices.html>

MICROSERVIÇOS, UMA DEFINIÇÃO DESSE NOVO TERMO ARQUITETURAL

"O termo 'Arquitetura de microsserviços' surgiu nos últimos anos para descrever uma maneira específica de projetar aplicativos de software como conjuntos de serviços implementáveis independentemente. Embora não exista uma definição precisa desse estilo arquitetônico, existem certas características comuns em torno da organização, em relação à capacidade de negócios, implantação automatizada, inteligência nos terminais e controle descentralizado de linguagens e dados."

Martin Fowler

O QUE SÃO MICROSERVIÇOS?

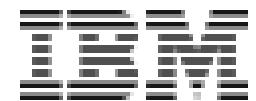


Azure

<https://azure.microsoft.com/pt-br/solutions/microservice-applications>

Google Cloud

<https://cloud.google.com/learn/what-is-microservices-architecture?hl=pt-br>



<https://www.ibm.com/br-pt/topics/microservices>



<https://aws.amazon.com/pt/microservices/>

vmware[®]
by Broadcom

<https://www.vmware.com/br/topics/glossary/content/microservices.html>



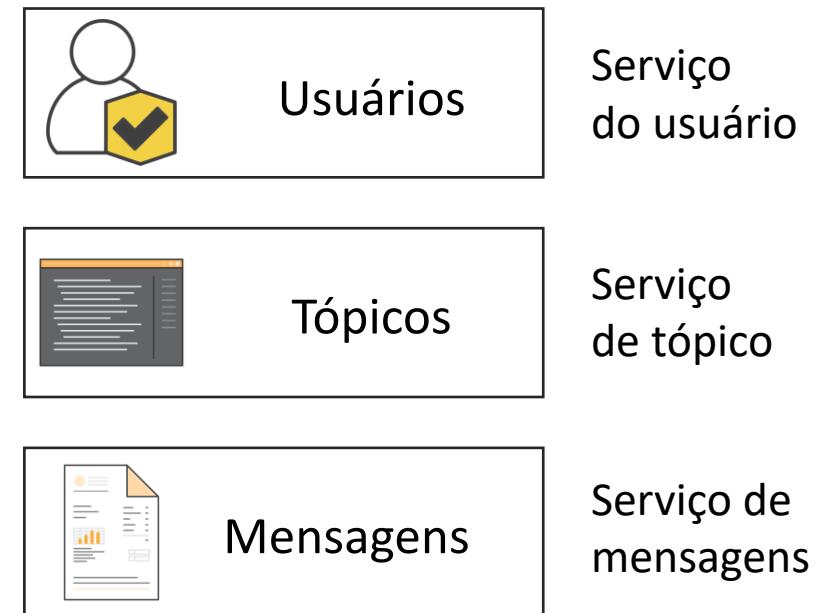
<https://www.redhat.com/pt-br/topics/microservices/what-are-microservices>

APLICAÇÕES MONOLÍTICAS VS. MICROSSERVIÇOS

Aplicação
monolítica



Aplicação
microsserviços



Fonte: <https://aws.amazon.com/pt/microservices/>

APlicações MONOLÍTICAS VS. MICROSSERVIÇOS

Aplicação monolítica

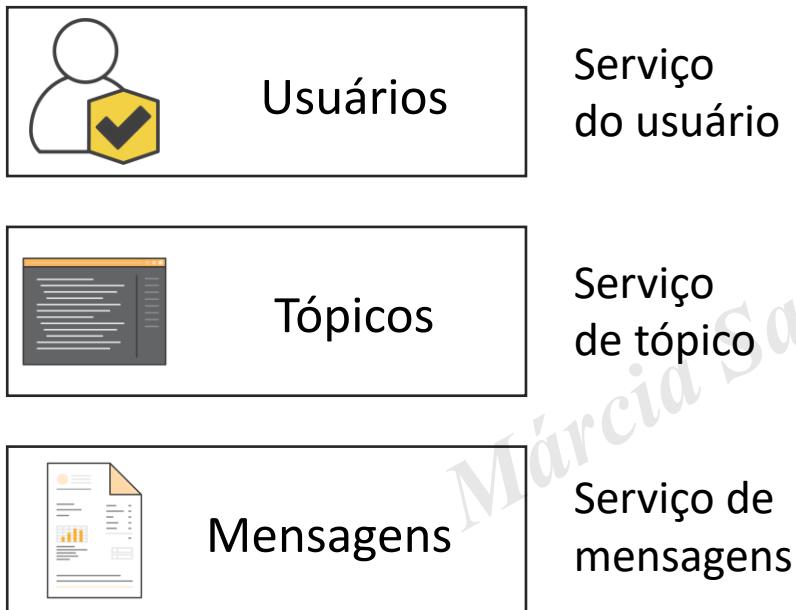


- Para entender os benefícios dos microsserviços, considere primeiro uma aplicação monolítica.
- Os três processos (usuários, tópicos e mensagens) de uma aplicação monolítica estão altamente acoplados. Eles funcionam como um único serviço.
- Se um processo da aplicação apresentar um pico de demanda, toda a arquitetura deverá ser dimensionada. Adicionar ou melhorar recursos torna-se mais complexo à medida que a base de código cresce, o que limita a experimentação e dificulta a implementação de novas ideias.
- A disponibilidade de aplicações monolíticas também fica em risco porque muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.

Fonte: <https://aws.amazon.com/pt/microservices/>

Aplicações monolíticas vs. microsserviços

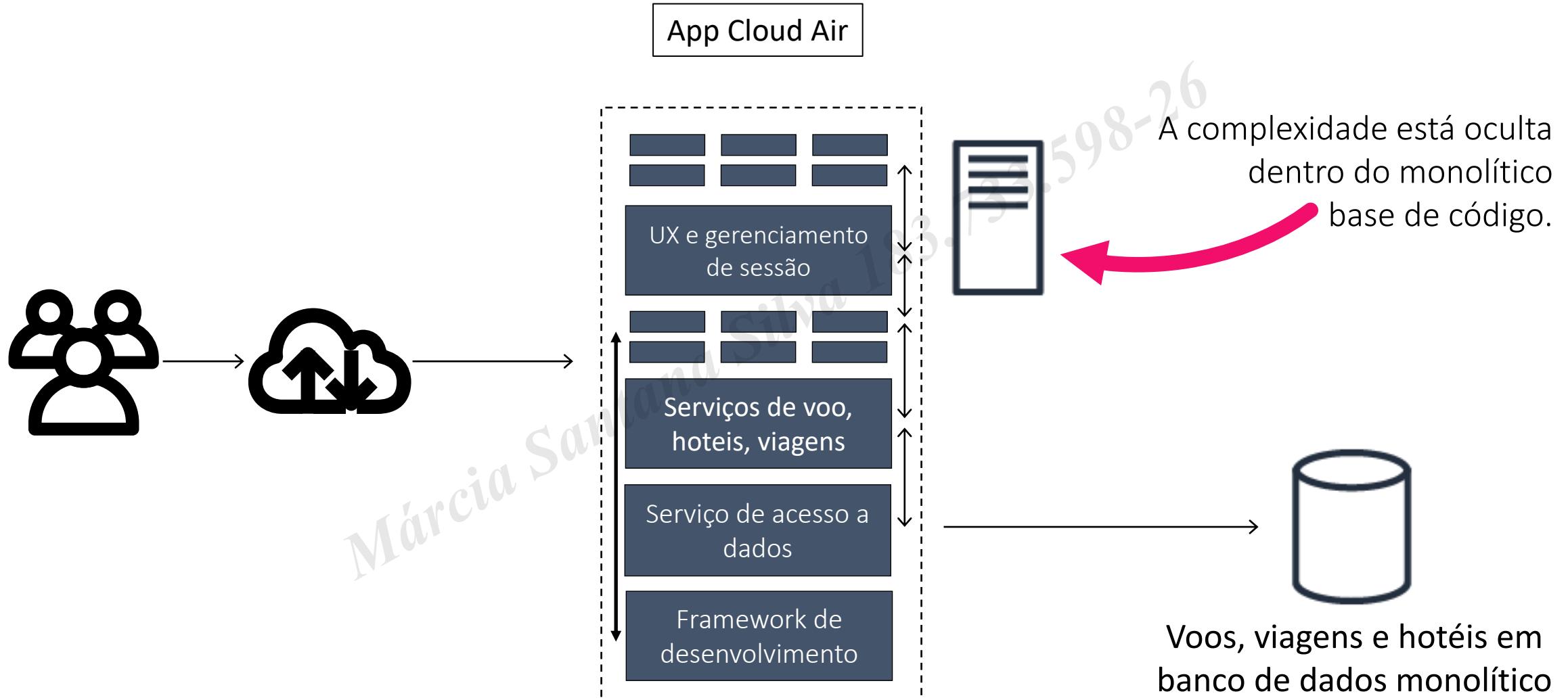
Aplicação microsserviços



- Agora, suponha que a mesma aplicação seja executada em uma arquitetura de microsserviços.
- Cada processo da aplicação é criado como um componente independente que é executado como um serviço. Os serviços se comunicam usando operações de API leves.
- Cada serviço executa uma única função que pode oferecer suporte a várias aplicações. Como os serviços são executados de maneira independente, eles podem ser atualizados, implantados e dimensionados para atender à demanda por funções específicas de uma aplicação.
- Uma arquitetura de microsserviços permite iteração, automação e agilidade geral muito mais rápidas. Inicie, falhe e recupere: tudo rapidamente.

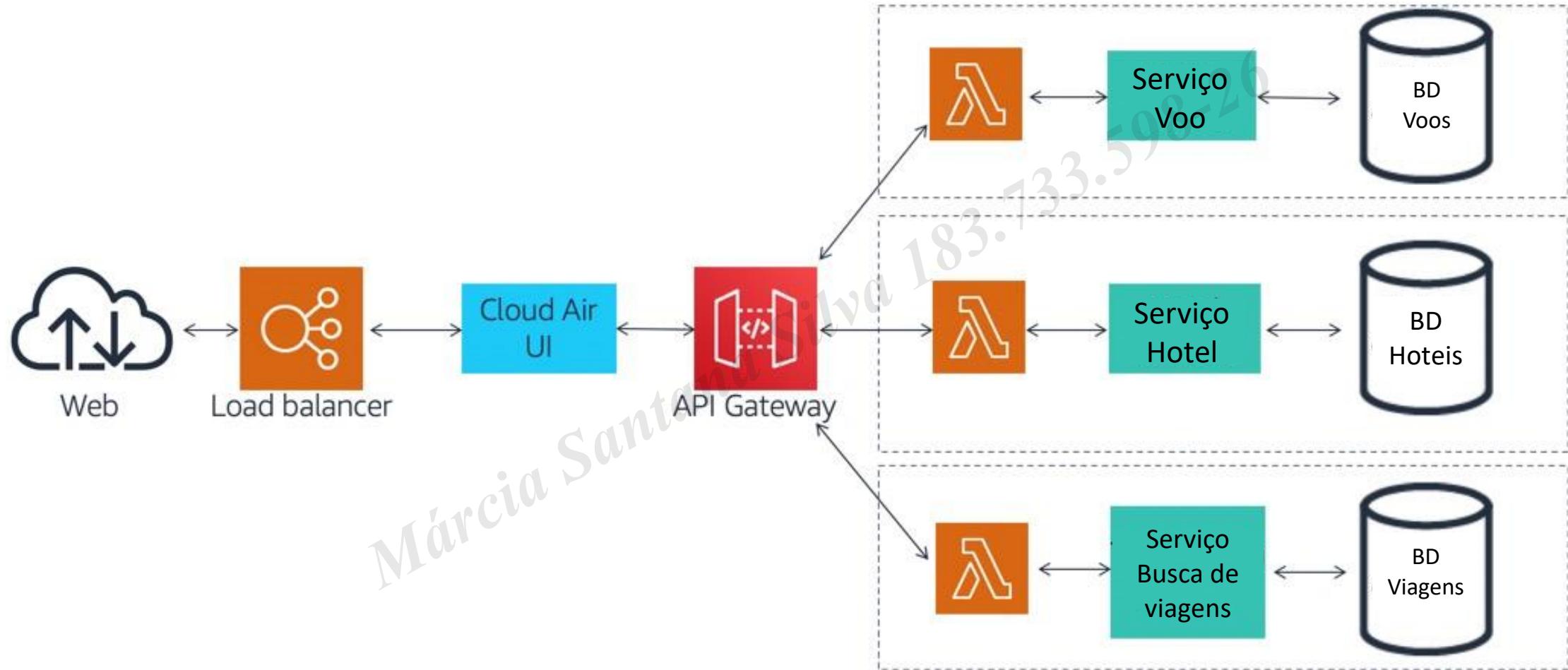
Fonte: <https://aws.amazon.com/pt/microservices/>

ARQUITETURA MONOLÍTICA



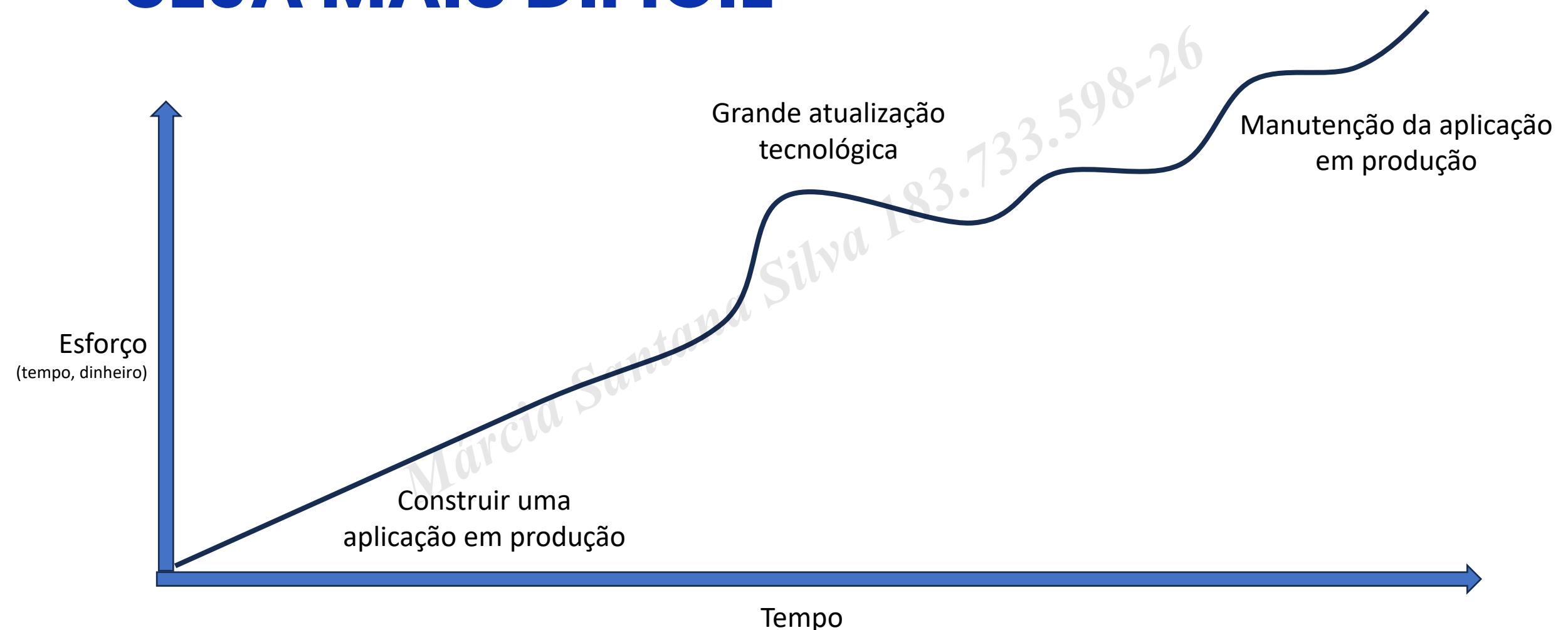
Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

ARQUITETURA MICROSERVIÇOS



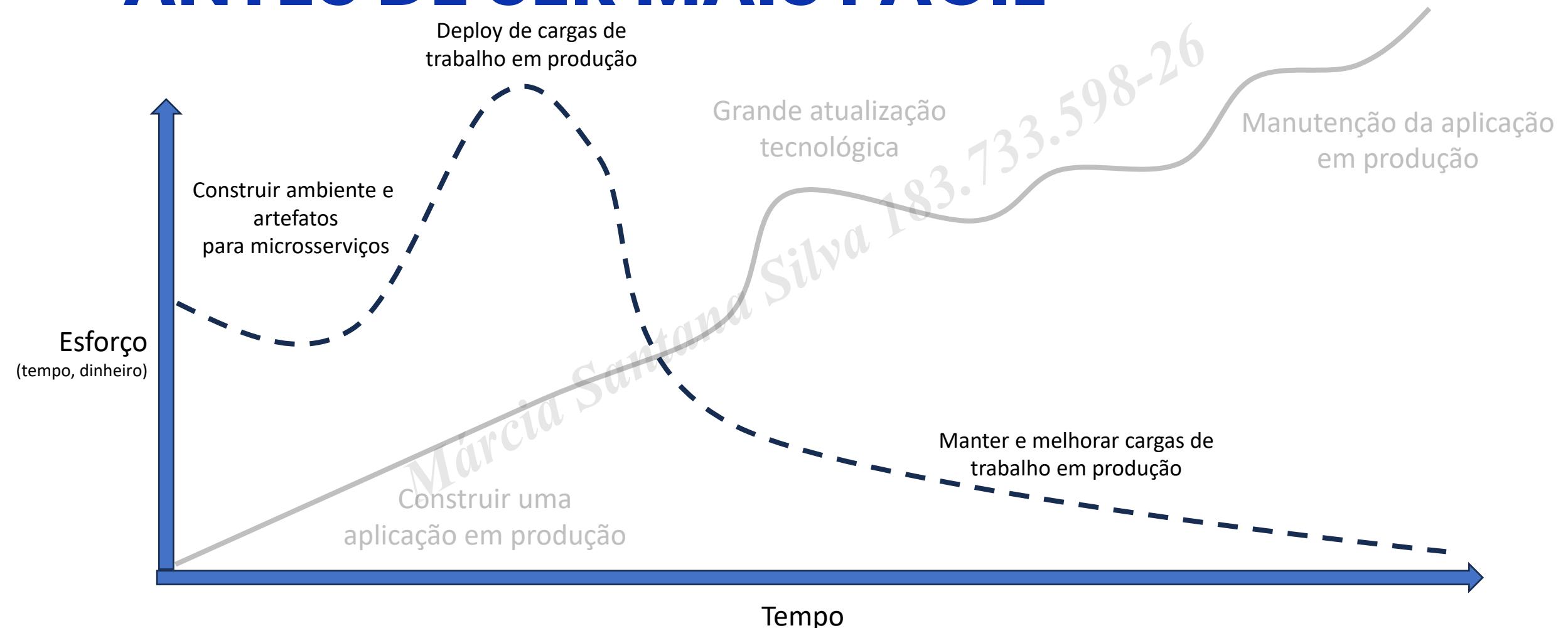
Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

MONOLITO É MAIS FÁCIL ANTES QUE SEJA MAIS DIFÍCIL



Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

MICROSSERVIÇOS É MAIS DIFÍCIL ANTES DE SER MAIS FÁCIL



Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>



APLICAÇÕES MONOLÍTICAS VS. MICROSSERVIÇOS

Tipo Aplicação	Prós	Contras
Monolítico v1 (toda funcionalidade em um aplicativo)	<ul style="list-style-type: none">Inicialmente simplesBaixas latências entre processosBase de código única, uma unidade de implementaçãoEficiente quanto a recursos em pequenas escalas	<ul style="list-style-type: none">A sobrecarga de coordenação aumenta à medida que a equipe cresceFraca imposição de modularidadeEscalonamento fracoImplementação tudo ou nada (paralisação)Longos tempos de build
Monolítico v2 (conjunto de camadas físicas monolíticas: front-end, servidor de aplicativos, camada de banco de dados)	<ul style="list-style-type: none">Inicialmente simplesFáceis consultas de uniãoEsquema único de implementaçãoEficiente quanto a recursos em pequenas escalas	<ul style="list-style-type: none">Tendência de maior acoplamento no decorrer do tempoEscalonamento e redundância fracos (tudo ou nada, somente vertical)Difícil de ajustarGerenciamento de esquema tudo ou nada
Microsserviços (Modular, independente, relação gráficas vs. Camadas físicas, persistência isolada)	<ul style="list-style-type: none">Cada unidade é simplesEscalonamento e desempenho independentesTeste e implementação independentesPode ajustar o desempenho com perfeição (cache, replicação, etc)	<ul style="list-style-type: none">Muitas unidades cooperandoMuitas recompras pequenasExige ferramentas mais sofisticadas e gerenciamento de dependênciasLatências de rede

Fonte: Manual de DevOps: como obter agilidade, confiabilidade e segurança em organizações tecnológicas, Capítulo 13

CASO PRIME VIDEO

The screenshot shows a dark-themed news article from the prime video | TECH website. At the top, there's a navigation bar with a menu icon, the 'prime video | TECH' logo with the Amazon smiley arrow, and a search icon. Below the header, the category 'Video Streaming' is shown in blue. The main title of the article is 'Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%'. A subtitle below the title reads: 'The move from a distributed microservices architecture to a monolith application helped achieve higher scale, resilience, and reduce costs.' The author of the article is Marcin Kolny, dated Mar 22, 2023. There are social sharing icons for LinkedIn, Twitter, Facebook, and Email. To the right of the article, there's a sidebar with a 'Most popular' section featuring a link to an article about Prime Video announcing Amazon.

Marcin Kolny
Mar 22, 2023



At Prime Video, we offer thousands of live streams to our customers. To ensure that customers seamlessly receive content, Prime Video set up a tool to monitor every stream viewed by customers. This tool allows us to automatically identify perceptual quality issues (for example, block corruption or audio/video sync problems) and trigger a process to fix them.

Most popular

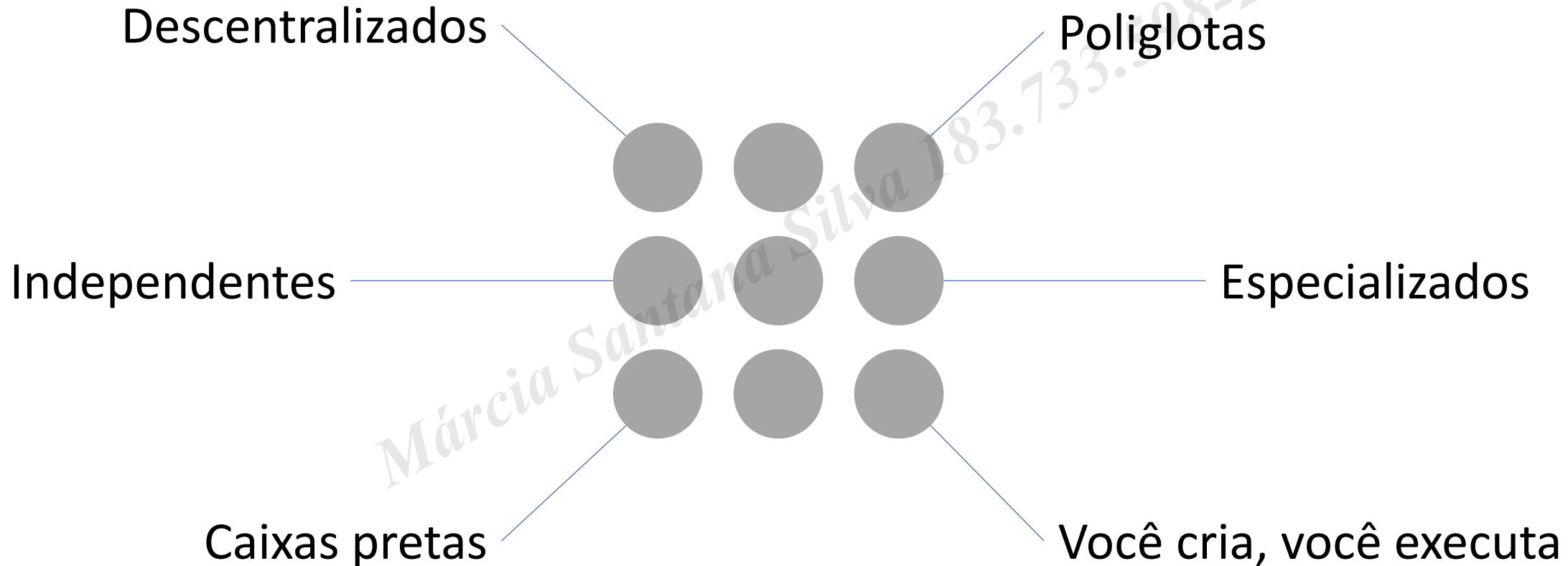
"We're just beginning to build the future of live sports streaming"

Feb 07, 2023

Prime Video announces Amazon

<https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90>

CARACTERÍSTICAS DOS MICROSSERVIÇOS



CARACTERÍSTICAS DOS MICROSERVIÇOS

Descentralizados	Independentes	Especializados	Poliglota	Caixas pretas	Você cria, você executa
<ul style="list-style-type: none">os microserviços são descentralizados na forma como são desenvolvidos, implantados, gerenciados e operados	<ul style="list-style-type: none">cada serviço do componente de uma arquitetura de microserviços pode ser desenvolvido, implantado, operado e dimensionado sem afetar a função de outros serviços	<ul style="list-style-type: none">cada serviço de componente é projetado para um conjunto de recursos e se concentra em resolver um problema específico	<ul style="list-style-type: none">as arquiteturas de microserviços adotam uma abordagem heterogênea para sistemas operacionais, linguagens de programação, armazenamentos de dados e ferramentas	<ul style="list-style-type: none">os detalhes da complexidade dos componentes de microserviços ficam escondidos de outros componentes	<ul style="list-style-type: none">o DevOps é um princípio organizacional fundamental para microserviços

CHECKPOINT



As aplicações de microsserviços são compostas por serviços independentes que se comunicam por APIs bem definidas

Os microsserviços compartilham as seguintes características:

Descentralizados

Independentes

Especializados

Poliglotas

Caixas pretas

Você cria, você executa

OS PRINCIPAIS DESAFIOS PARA IMPLEMENTAR ENTREGA CONTÍNUA (CONTINUOUS DELIVERY)

Principais desafios segundo o State of DevOps Report 2020 apresentado pela Puppet e CircleCI:

Cobertura de teste incompleta

Escrever bons testes que cubram todos os cenários possíveis é quase impossível, especialmente em ambientes complexos onde há um número infinito de comportamentos de usuário, dependências, arquiteturas dinâmicas e muito mais.”

Mentalidade organizacional

“Não é nenhuma surpresa que a mentalidade organizacional seja um desafio comum em cada grupo. Ouvimos muito isso em nosso trabalho com empresas, e é a única coisa com a qual os líderes seniores e profissionais estão de acordo.”

Arquitetura de aplicação fortemente acoplada

A arquitetura de aplicação fortemente acoplada é uma das principais restrições para as equipes de entrega. Atualizar sua aplicação ou serviço requer coordenação com outras equipes, retardando a entrega de cada equipe devido a dependências complexas. O acoplamento fraco significa que os aplicativos são mais modulares, para que as equipes possam entregar em seu próprio ritmo, usando seus próprios fluxos de trabalho

Fonte: <https://circleci.com/resources/state-of-devops-report-2020/>

ARQUITETURAS LEVEMENTE ACOPLADAS

"As organizações com esses tipos de arquiteturas orientadas a serviços, como o Google e a Amazon, têm flexibilidade e escalabilidade incríveis. Elas têm dezenas de milhares de desenvolvedores, em que pequenas equipes ainda podem ser incrivelmente produtivas."

Tradução livre de <https://dora.dev/devops-capabilities/technical/loosely-coupled-architecture/>



Randy Shoup, ex-diretor de engenharia do eBay, Stitch Fix, Google, WeWork



NÃO EXISTE UMA SOLUÇÃO ARQUITETURAL DEFINITIVA

Randy Shoup, também observou:

"Não há uma arquitetura perfeita para todos os produtos e todas as escalas. Qualquer arquitetura atende a um conjunto específico de metas ou intervalo de requisitos e restrições, como tempo de lançamento, facilidade de desenvolvimento de funcionalidades, dimensionamento etc. A funcionalidade de qualquer produto ou serviço quase certamente crescerá ao longo do tempo e nossas necessidades na arquitetura também mudarão. O que funciona na escala 1:X raramente funciona na escala 10:X ou 100:X."

Tradução livre de <https://dora.dev/devops-capabilities/technical/loosely-coupled-architecture/>

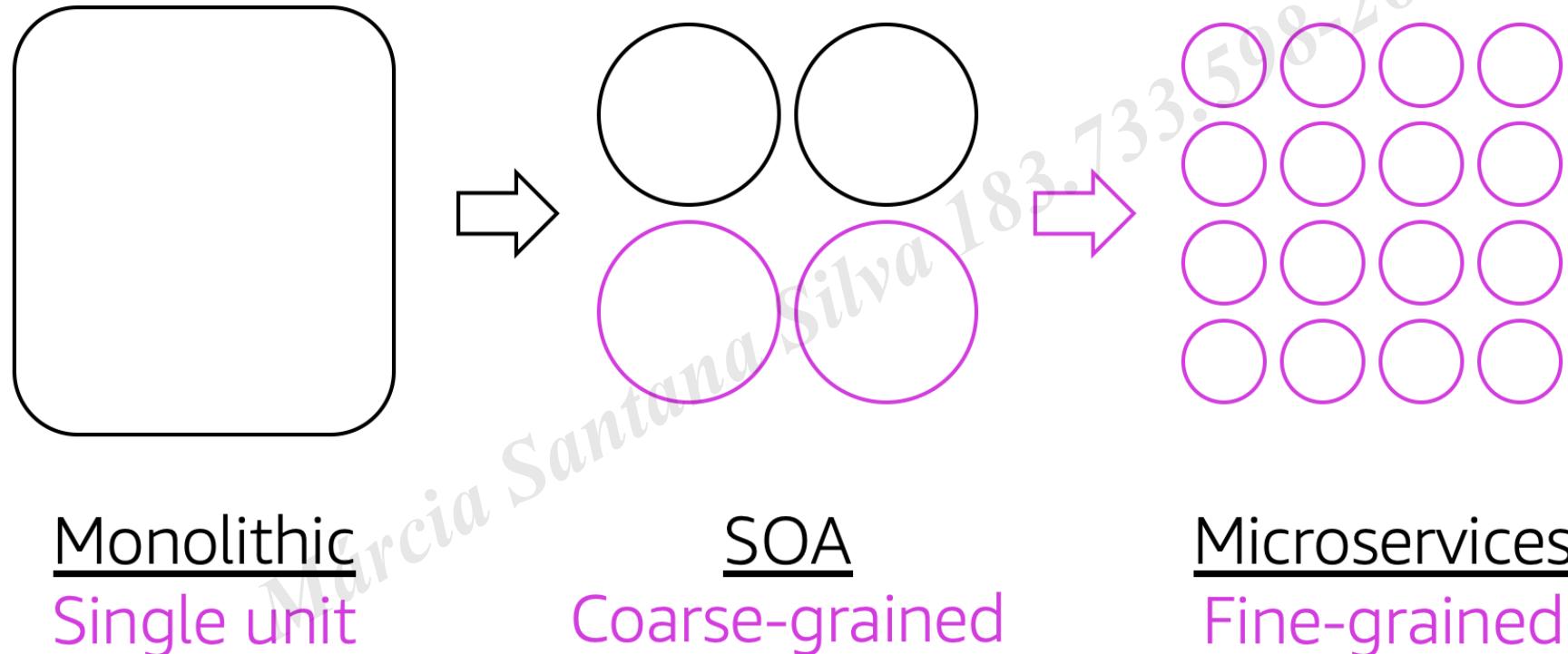


Randy Shoup, ex-diretor de engenharia do eBay, Stitch Fix, Google, WeWork

ARQUITETURA DE MICROSSERVIÇOS

Parte 3

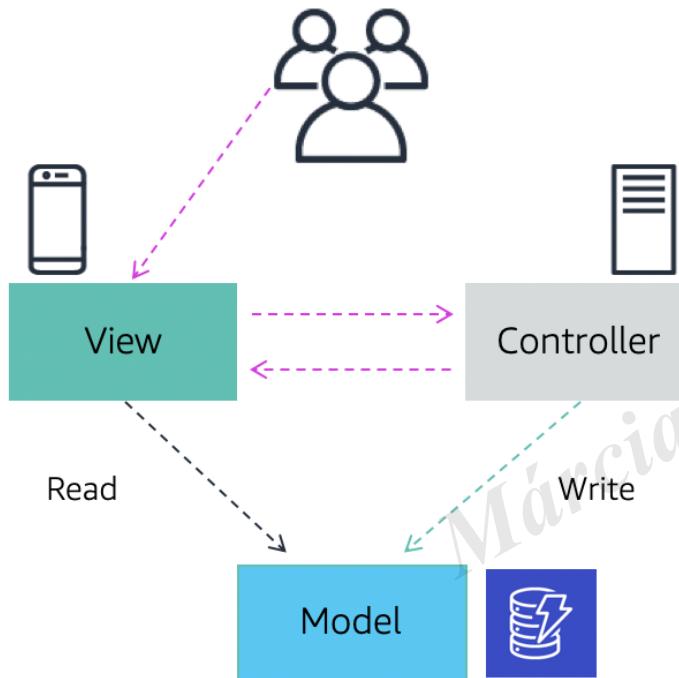
EVOLUÇÃO DA ARQUITETURA



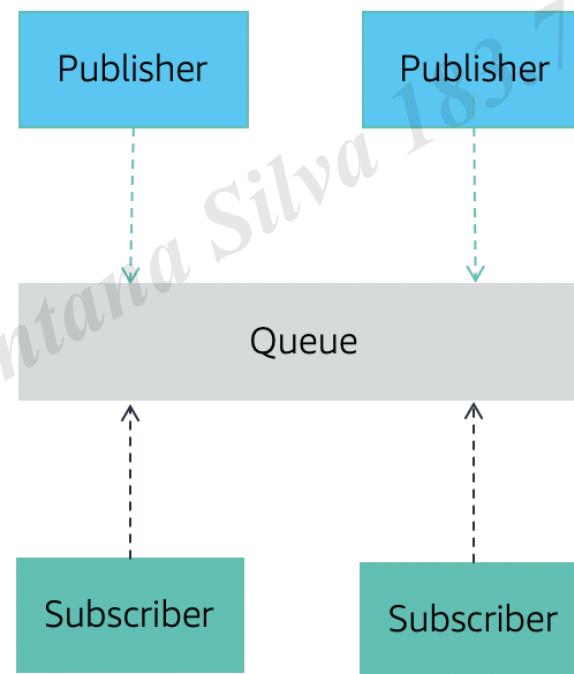
Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

DIFERENTES FORMAS DE IMPLEMENTAR MICROSERVIÇOS

Model/View/Controller (MVC)

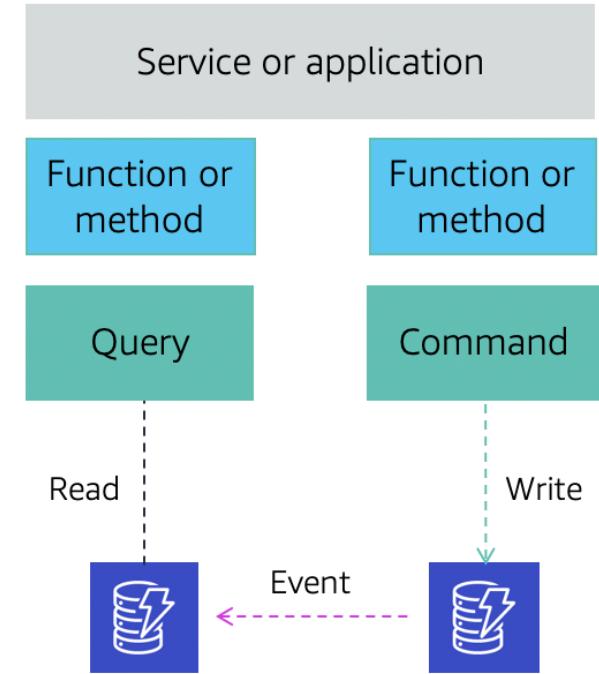


Pub-sub



CQRS

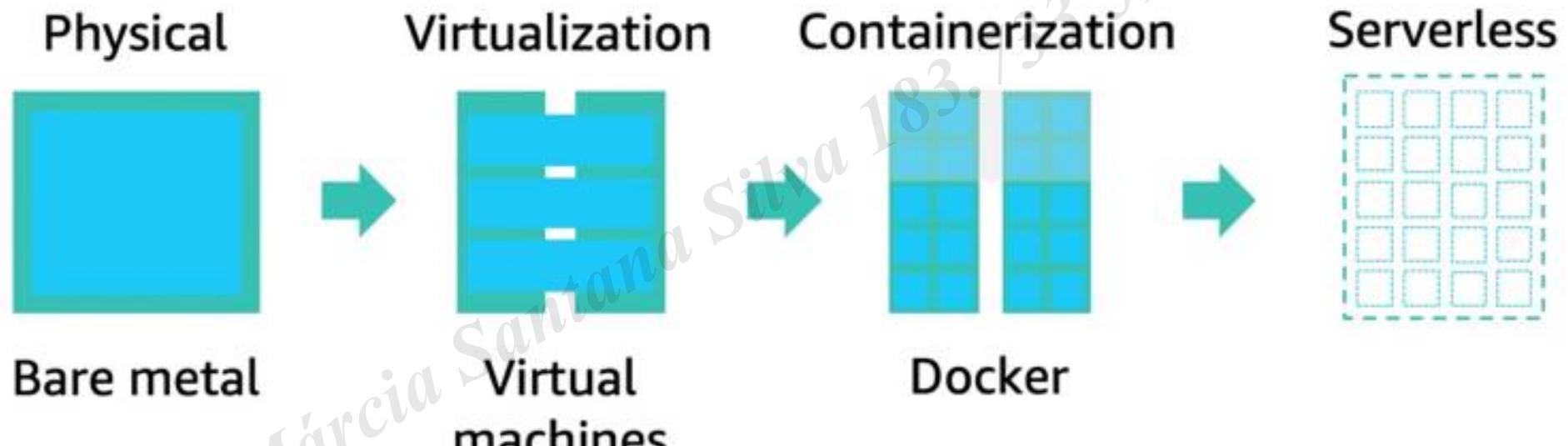
Command Query Responsibility Segregation



Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

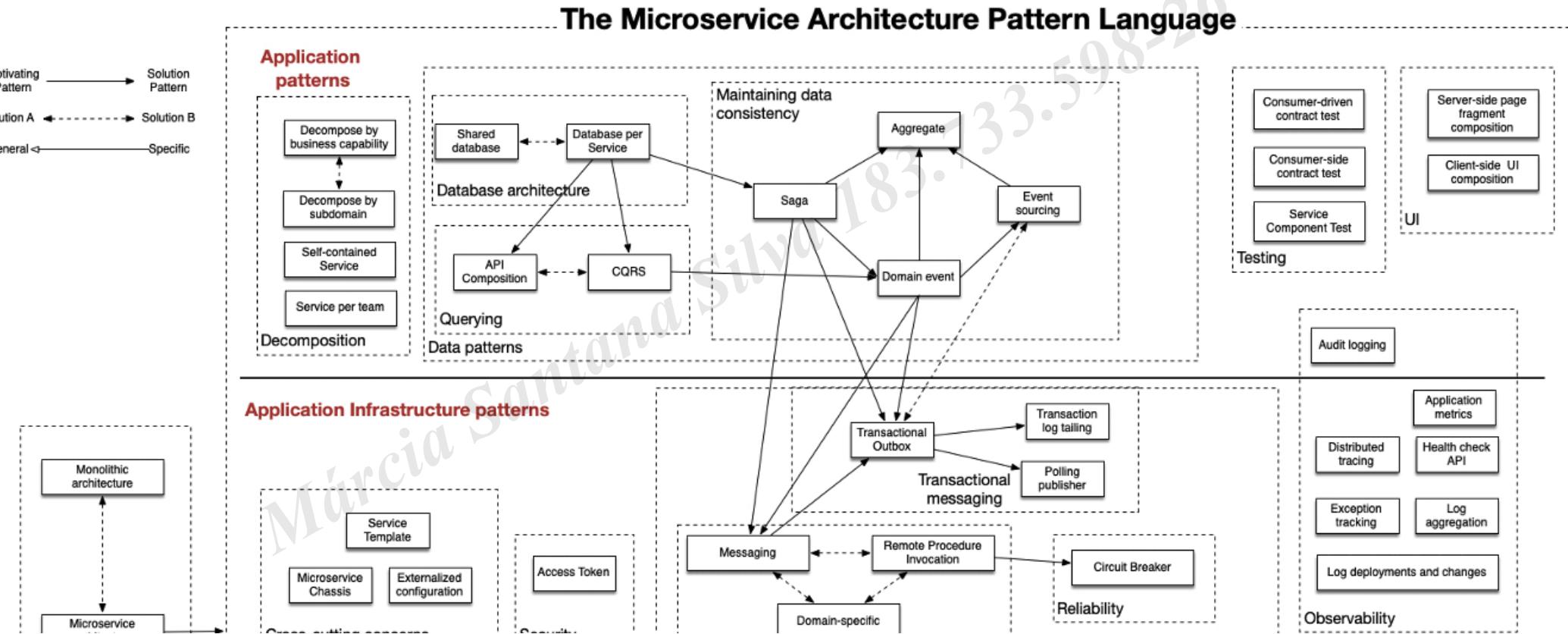
*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.
Proibida a reprodução, total ou parcial, sem autorização. Lei nº 9610/98

DIFERENTES FORMAS DE IMPLEMENTAR MICROSERVIÇOS



Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

ARQUITETURA E PADRÕES



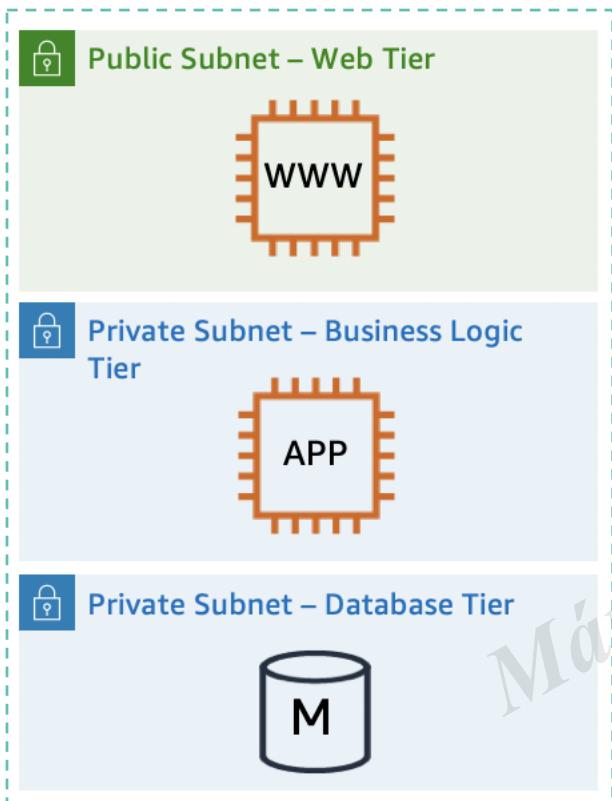
Chris Richardson, Livro Microservices Patterns

Fonte: <https://microservices.io/patterns/index.html>

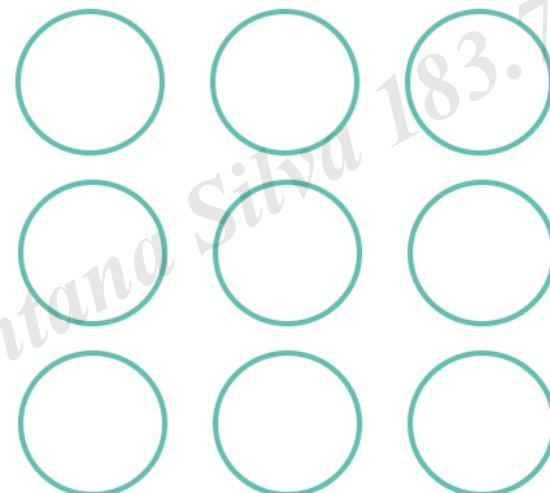
*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.
Proibida a reprodução, total ou parcial, sem autorização. Lei nº 9610/98

ARQUITETURA E PADRÕES

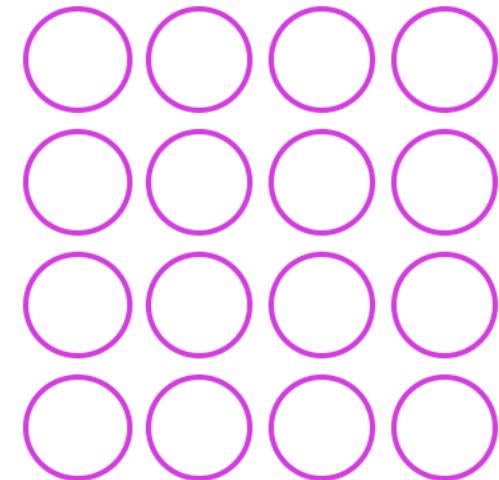
N-tier



Service-oriented architecture
(SOA)



Microservices



Fonte: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/modernization-pattern-list.html>

TEOREMA CAP

O teorema CAP afirma que um sistema distribuído, composto por vários nós que armazenam dados, não pode fornecer simultaneamente mais de duas das três garantias a seguir:

Consistência

cada solicitação de leitura recebe a gravação mais recente ou um erro quando a consistência não pode ser garantida.

Disponibilidade

cada solicitação recebe uma resposta sem erros, mesmo quando os nós estão inativos ou indisponíveis.

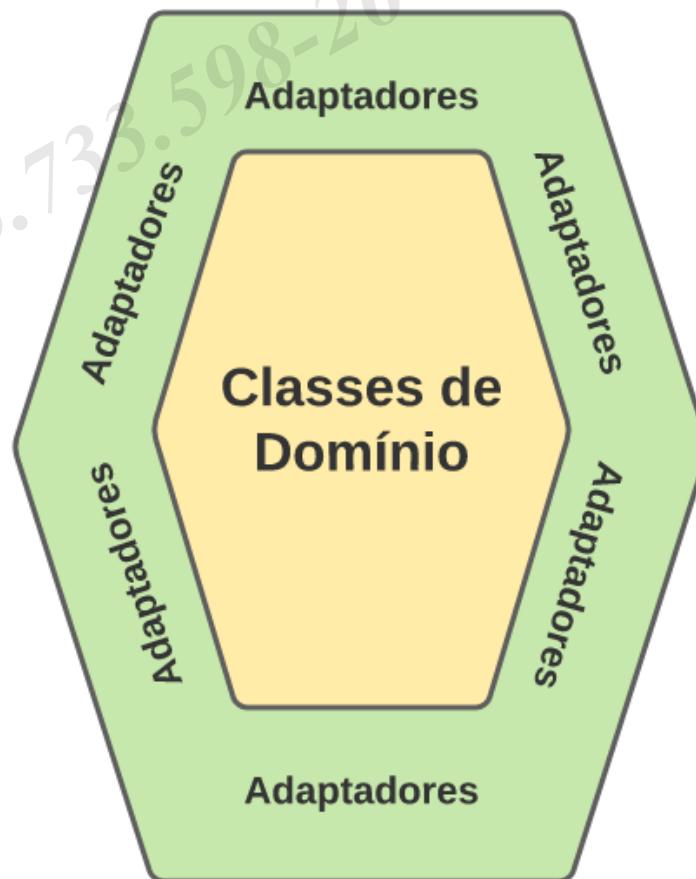
Tolerância à partição

o sistema continua operando apesar da perda de um número arbitrário de mensagens entre os nós.

Fonte: https://docs.aws.amazon.com/pt_br/whitepapers/latest/availability-and-beyond-improving-resilience/cap-theorem.html

ARQUITETURA HEXAGONAL

- Definida por Alistair Cockburn, em meados dos anos 90
- Ideia central: usar adaptadores para mediar a comunicação entre domínio e o restante do sistema
- Adaptadores: mesma ideia de padrões de projeto (GoF)
- Nome deriva do diagrama usado para ilustrar a arquitetura, formado por dois hexágonos concêntricos



Fonte: O que é uma Arquitetura Hexagonal?, Prof. Marco Túlio Valente, <https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html>



ARQUITETURA HEXAGONAL - OBJETIVOS

- Estabelecer uma separação clara entre:
 - Domínio limpo de tecnologia (classes do negócio)
 - Restante do sistema

Domínio limpo de tecnologia

- Camada de domínio não conhece:
 - Banco de dados usados pelo sistema
 - Frontend usados pelo sistema
 - Gateways de pagamentos usados pelo sistema
 - Serviços externos com os quais o sistema interage

Restante do sistema

- Interface com o usuário (Web, mobile, etc)
- Persistência e BDs
- Integrações com outros sistemas

Fonte: O que é uma Arquitetura Hexagonal?, Prof. Marco Túlio Valente, <https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html>



ARQUITETURA HEXAGONAL - VANTAGENS

- Permite que desenvolvedores se concentrem no domínio:
 - Representa o propósito do sistema
 - Responsável pela geração de valor
- Testabilidade (mais fácil testar domínio limpo de tecnologia)
- Maior facilidade de troca de bibliotecas, frameworks, BDs, etc.

Fonte: O que é uma Arquitetura Hexagonal?, Prof. Marco Túlio Valente, <https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html>



ARQUITETURA HEXAGONAL - ADAPTADORES

Domínio “limpo” de tecnologia



Adaptadores

<https://alistair.cockburn.us/hexagonal-architecture/>

Fonte: O que é uma Arquitetura Hexagonal?, Prof. Marco Túlio Valente, <https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html>

Restante do Sistema



Front-end



Banco de dados



Sistemas Externos



Cache



Bibliotecas Log

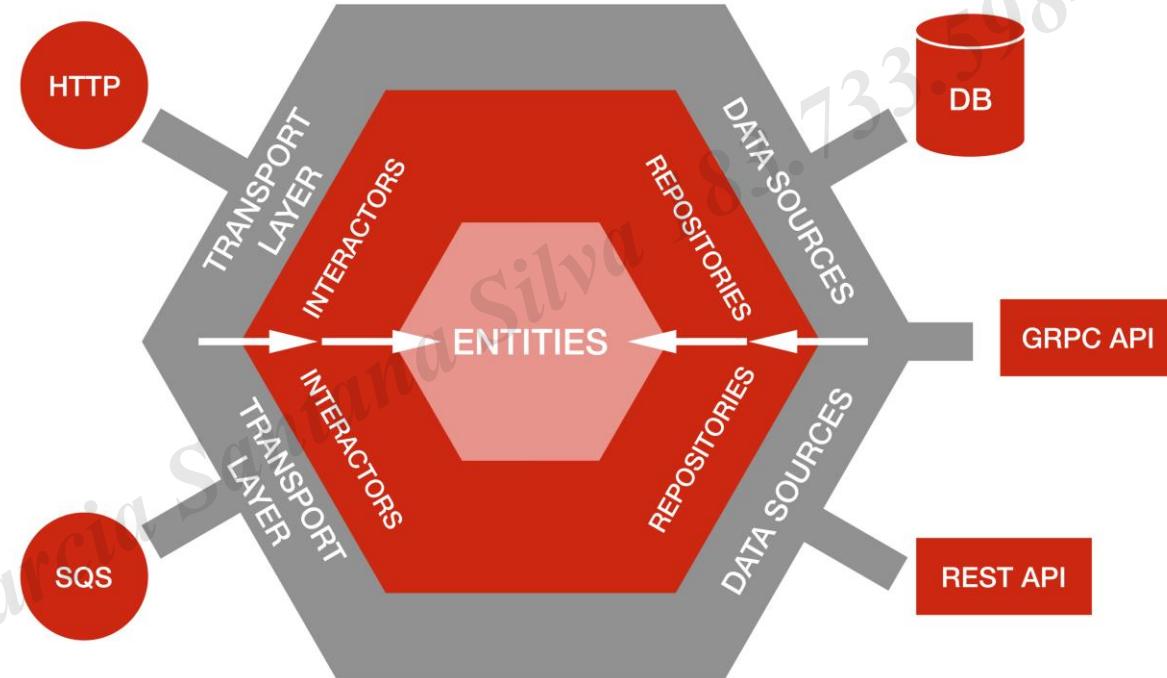


Gateways de Pagamento



Pub/Sub

ARQUITETURA HEXAGONAL – EXEMPLO NETFLIX



Fonte: <https://netflixtechblog.com/ready-for-changes-with-hexagonal-architecture-b315ec967749>

THE TWELVE-FACTOR APP



INTRODUÇÃO

Na era moderna, software é comumente entregue como um serviço: denominados *web apps*, ou *software-como-serviço*. A aplicação doze-fatores é uma metodologia para construir softwares-como-serviço que:

- Usam formatos declarativos para automatizar a configuração inicial, minimizar tempo e custo para novos desenvolvedores participarem do projeto;
- Tem um contrato claro com o sistema operacional que o suporta, oferecendo portabilidade máxima entre ambientes que o executem;
- São adequados para implantação em modernas plataformas em nuvem, evitando a necessidade por servidores e administração do sistema;
- Minimizam a divergência entre desenvolvimento e produção, permitindo a implantação contínua para máxima agilidade;
- E podem escalar sem significativas mudanças em ferramentas, arquiteturas, ou práticas de desenvolvimento.

A metodologia doze-fatores pode ser aplicada a aplicações escritas em qualquer linguagem de programação, e que utilizem qualquer combinação de serviços de suportes (banco de dados, filas, cache de memória, etc).

Fonte: <https://12factor.net/>



OBJETIVOS THE TWELVE-FACTOR APP

- Minimize o esforço de configuração do novo desenvolvedor.
- Maximize a portabilidade do aplicativo.
- Implante aplicativos facilmente em plataformas de nuvem modernas.
- Manter a paridade do ambiente entre o desenvolvimento e a produção.

1. Codebase

5. Build, release, run

9. Disposability

2. Dependencies

6. Processes

10. Dev/prod parity

3. Config

7. Port-binding

11. Logs

4. Backing services

8. Concurrency

12. Administrative processes

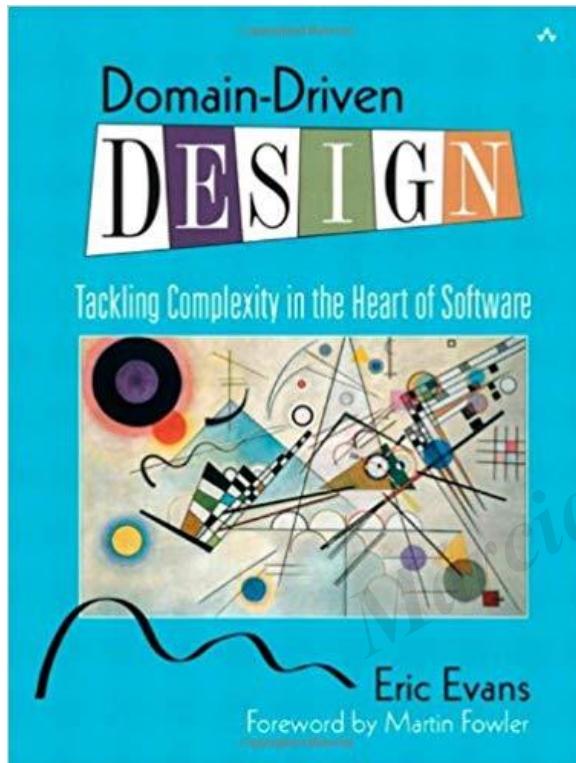
EXERCÍCIO



- Acessar o endereço <https://12factor.net/> para estudar cada um dos doze fatores.



Domain Driven Design

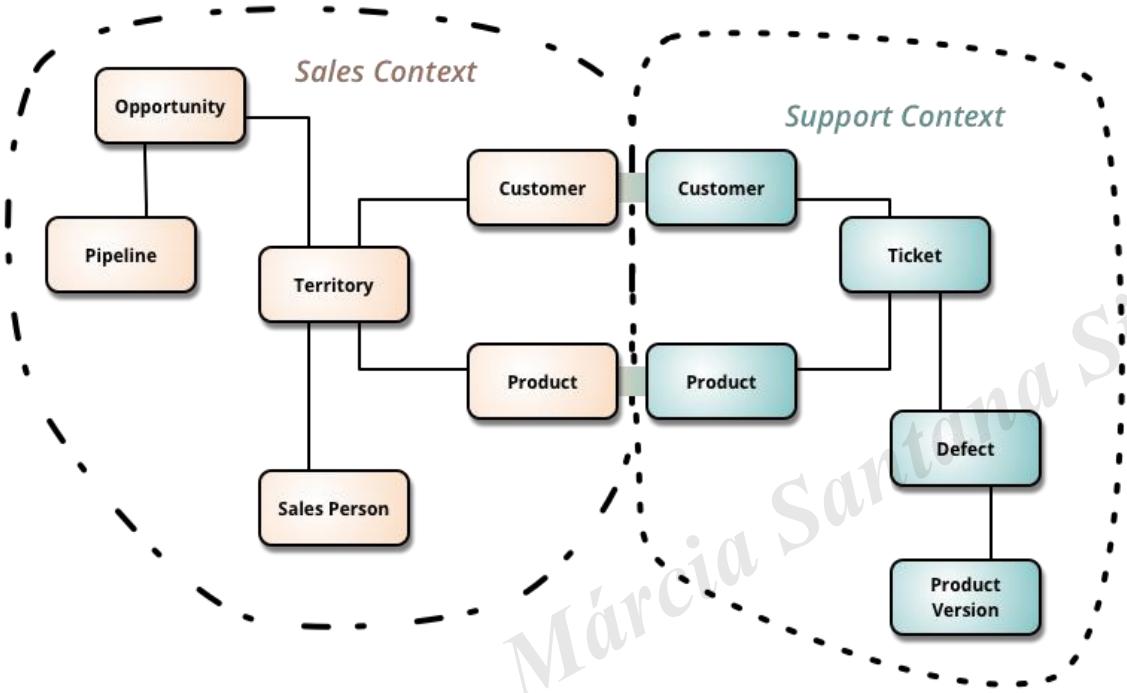


- O Domain Driven Design (DDD) é uma abordagem para o desenvolvimento de software para necessidades complexas, conectando a implementação a um modelo em evolução.
- Coloca o foco principal do projeto no domínio central e na lógica do domínio.
- Baseia projetos complexos em um modelo de domínio.
- Inicia uma colaboração criativa (usando uma linguagem ubíqua, ou UL) entre especialistas técnicos e de domínio para refinar iterativamente um modelo conceitual que trata de problemas de domínio específicos.

DDD E BOUNDED CONTEXT (CONTEXTO LIMITADO)

- Os Microserviços devem ter um contexto limitado bem definido e executar apenas uma tarefa.
- Um contexto limitado encapsula um único domínio.
- Um design orientado a domínio
 - Define os pontos de integração com outros domínios
 - Alinha-se bem com as características dos Microserviços
- Cuidado ao criar Microserviços monológicos!

DDD E BOUNDED CONTEXT (CONTEXTO LIMITADO)



- Vários fatores traçam limites entre os contextos, sendo a cultura humana dominante.
- Contextos limitados têm conceitos não relacionados (por exemplo, tíquetes de suporte em um contexto de suporte ao cliente).
- Eles também compartilham conceitos (ex. Produtos e clientes).

Fonte: <https://www.martinfowler.com/bliki/BoundedContext.html>

EXEMPLO: GOOGLE CLOUD DATASTORE

- 💡 Uma arquitetura **altamente acoplada** pode impedir a **produtividade de todos** e a capacidade de fazer mudanças com segurança.
- 💡 Por outro lado, uma **arquitetura levemente acoplada** promove produtividade e segurança com interfaces bem definidas que impõem como os módulos se conectam uns com os outros.
- 💡 Uma **arquitetura levemente acoplada** permite que equipes pequenas e **produtivas façam mudanças que possam ser implantadas de forma segura e independente**.
- 💡 Além disso, como cada serviço também tem uma API bem definida, ele facilita o teste de serviços e a criação de contratos de nível de serviço (SLAs) e de outros tipos entre equipes.

Google Cloud Datastore

Cloud Datastore: No SQL service

- ✓ Highly scalable and resilient
- ✓ Strong transactional consistency
- ✓ SQL-like rich query capabilities

Megastore: geo-scale structured database

- ✓ Multi-row transactions
- ✓ Synchronous cross-datacenter replication

Bigtable: cluster-level structured storage

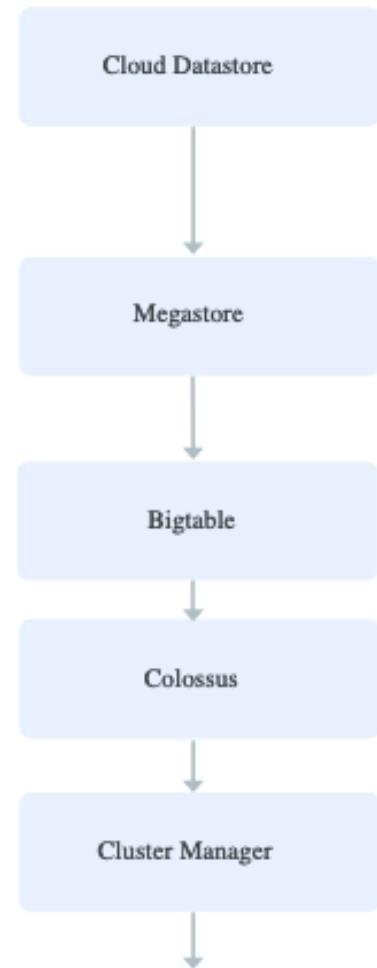
- ✓ (row, column, timestamp) -> cell contents

Colossus: next-generation clustered file system

- ✓ Block distribution and replication

Cluster management infrastructure

- ✓ Task scheduling, machine assignment





LEI DE CONWAY

Conway's Law

20 October 2022



Martin Fowler

- ❖ TEAM ORGANIZATION
- ❖ ENTERPRISE ARCHITECTURE
- ❖ APPLICATION ARCHITECTURE

Pretty much all the practitioners I favor in Software Architecture are deeply suspicious of any kind of general law in the field. Good software architecture is very context-specific, analyzing trade-offs that resolve differently across a wide range of environments. But if there is one thing they all agree on, it's the importance and power of Conway's Law. Important enough to affect every system I've come across, and powerful enough that you're doomed to defeat if you try to fight it.

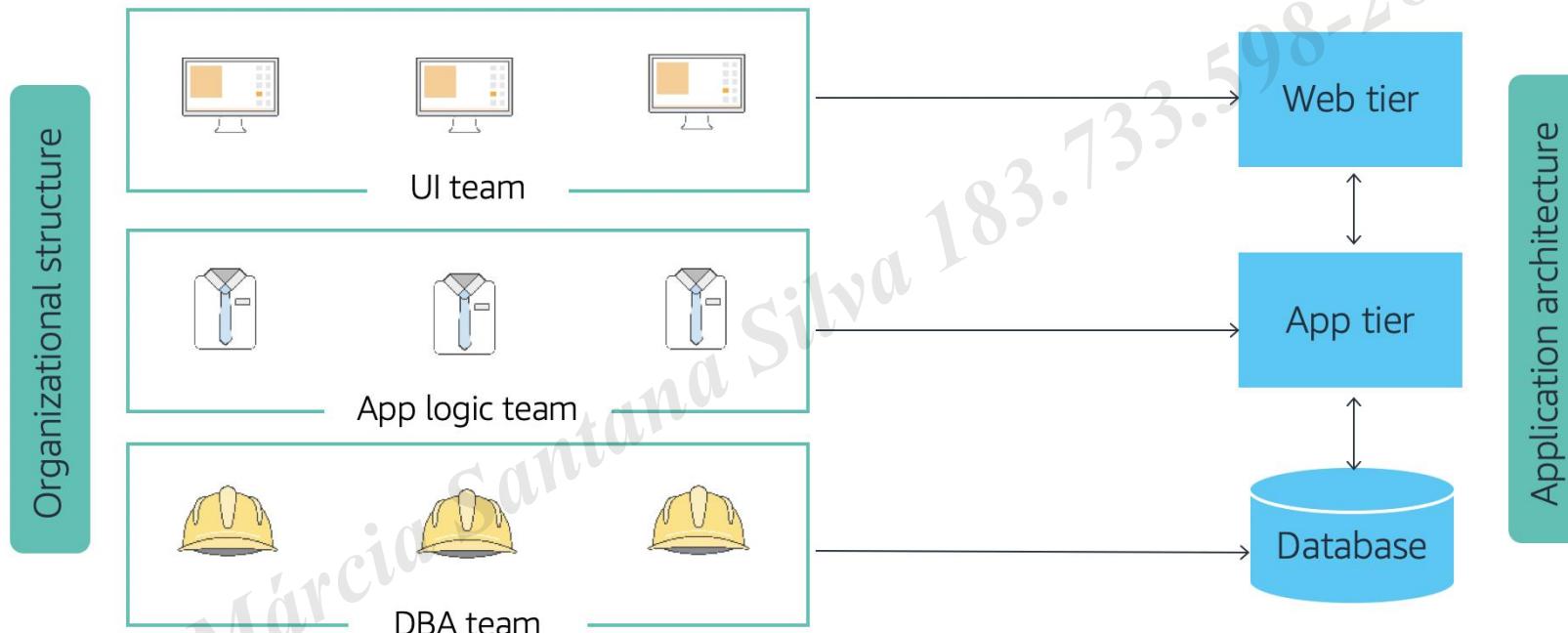
The law is probably best stated, by its author, as: [1]

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

-- Melvin Conway

<https://martinfowler.com/bliki/ConwaysLaw.html>

LEI DE CONWAY: ORGANIZAÇÃO MONOLÍTICA

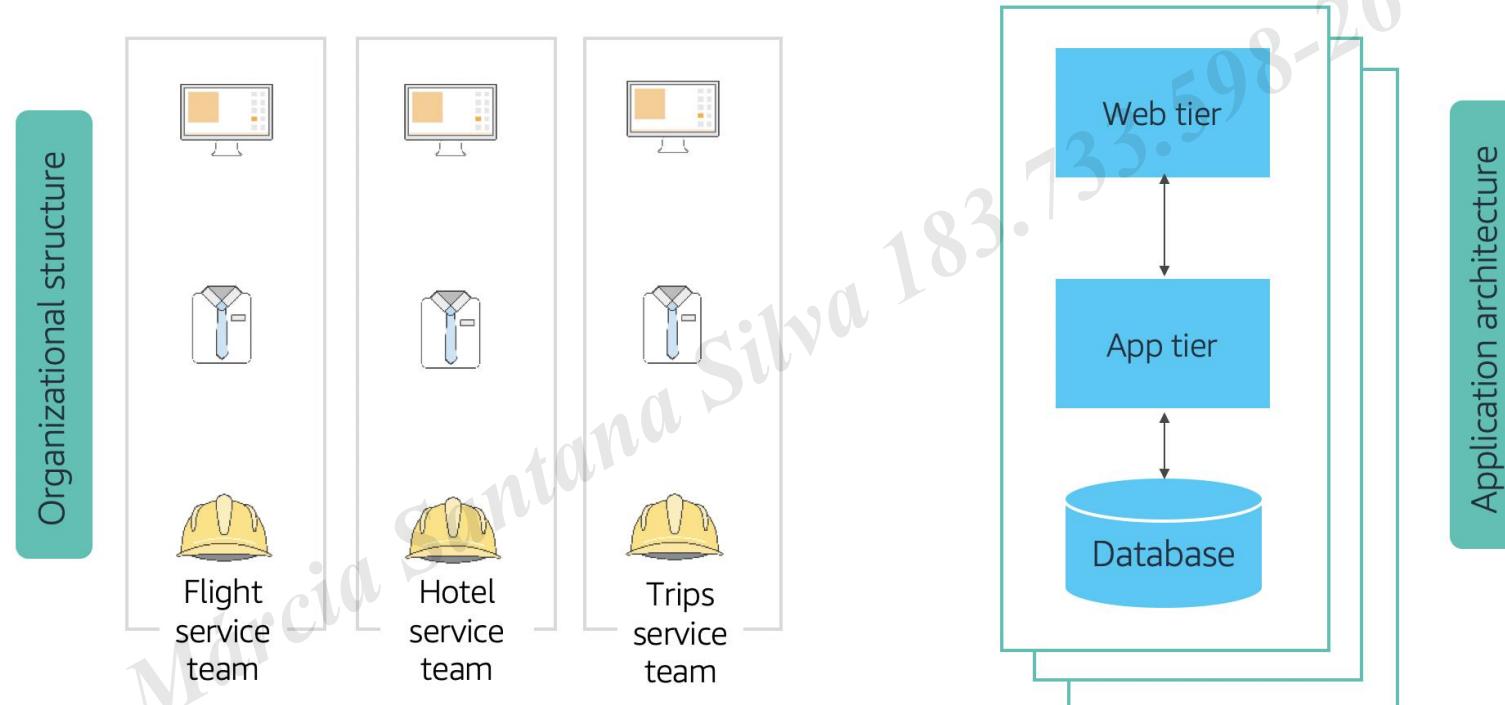


Lei de Conway

“Organizações que projetam sistemas (no sentido amplo usado aqui) são obrigadas [compelidas] a produzir projetos que sejam cópias das estruturas de comunicação dessas organizações.”

- Melvin Conway (1967)

LEI DE CONWAY: ORGANIZAÇÃO MICROSSERVIÇOS



Lei de Conway

“Organizações que projetam sistemas (no sentido amplo usado aqui) são obrigadas [compelidas] a produzir projetos que sejam cópias das estruturas de comunicação dessas organizações.”

- Melvin Conway (1967)

EXERCÍCIO



- Explorar uma aplicação baseada em microsserviços em uma Plataforma de Nuvem Pública.

EXERCÍCIOS E PESQUISAS

EXEMPLOS DE PADRÕES DE MICROSSERVIÇOS



- AWS - Implementação de microsserviços
- https://docs.aws.amazon.com/pt_br/whitepapers/latest/microservices-on-aws/microservices-on-aws.html
- https://docs.aws.amazon.com/pt_br/whitepapers/latest/microservices-on-aws/microservices-on-aws.pdf
- Microsoft Azure - Microsserviços do .NET
- <https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/>
- Google Cloud - Introdução a microsserviços
- <https://cloud.google.com/architecture/microservices-architecture-introduction?hl=pt-br>

EXEMPLO: CIELO



Cielo Developers

Portal de conteúdo Cielo. Documentos, artigos e tutoriais de todos os produtos Cielo estão disponíveis nesta página.

1. API E-commerce

- [Manual de Integração API e-commerce Cielo](#) >
- [Autenticação 3DS 2.0](#) >
- [Manual E-Wallets \(Carteiras Digitais\)](#) >
- [API de Chargeback](#) >
- [Integração VTEX](#) >
- [Integração Nuvemshop](#) >
- [Tutoriais](#) >
- [Prevenção contra Fraudes e Chargeback](#) >
- [Tokenização de Bandeira – Mandate Visa](#) >
- [Postman Cielo](#) >
- [FAQ](#) >

2. Checkout e Link de Pagamento

- [Integração API Checkout Cielo](#) >
- [API Link de Pagamento Cielo](#) >
- [Tutorial Checkout e Link de Pagamento](#) >
- [API de Controle Transacional](#) >
- [FAQ](#) >
- [Tokenização de Bandeira – Mandate Visa](#) >
- [Postman Cielo](#) >

3. API Cielo Conecta

- [Manual de Integração Cielo Conecta](#) >

<https://developer.cielo.github.io/>

EXEMPLO: EMBRAPA



A PLATAFORMA DE APIs DA EMBRAPA

A plataforma AgroAPI oferece informações e modelos agropecuários gerados pela Embrapa que podem ser utilizados por empresas, instituições públicas e privadas e startups para a criação de softwares, sistemas web e aplicativos móveis para o setor agropecuário, com redução de custo e de tempo. O acesso aos dados e modelos é realizado de forma virtual por meio de APIs (Interface de Programação de Aplicativos, na tradução do inglês) – um conjunto de padrões e linguagens de programação que permite, de maneira automatizada, a comunicação entre sistemas diferentes de forma ágil e segura.

ClimAPI



PlantAnnot



Agritec



<https://www.agroapi.cnptia.embrapa.br/portal/>

ARTIGO

- Microservices: The Journey So Far and Challenges Ahead

Journals & Magazines > IEEE Software > Volume: 35 Issue: 3 

Microservices: The Journey So Far and Challenges Ahead

Publisher: IEEE

Cite This

PDF

Pooyan Jamshidi ; Claus Pahl ; Nabor C. Mendonça ; James Lewis ; Stefan Tilkov [All Authors](#)

269

Cites in
Papers

27962

Full
Text Views



 Open Access

Abstract

Document Sections

» Microservices and Service-Oriented Architecture

» Microservices' Benefits

Abstract:

Microservices are an architectural approach emerging out of service-oriented architecture, emphasizing self-management and lightweightness as the means to improve software agility, scalability, and autonomy. This article examines microservice evolution from the technological and architectural perspectives and discusses key challenges facing future microservice developments.

Published in: [IEEE Software](#) (Volume: 35 , Issue: 3, May/June 2018)

<https://ieeexplore.ieee.org/document/8354433>

OBRIGADO!

<https://www.linkedin.com/in/josemariacesariojunior/>