

# **EAD** **UNISANTA**

## **SISTEMAS DISTRIBUÍDOS**

Me. Joseffe Barroso de Oliveira

### **GUIA DA DISCIPLINA**

# 1. CONCEITUAÇÃO E CARACTERÍSTICAS DOS SISTEMAS DISTRIBUÍDOS

## Objetivo

O objetivo deste capítulo é apresentar o conceito e as características dos sistemas distribuídos.

## Introdução

Um sistema distribuído é uma coleção de programas de computador que utilizam recursos computacionais em vários pontos centrais de computação diferentes para atingir um objetivo comum e compartilhado. Os sistemas distribuídos visam remover gargalos ou pontos centrais de falha de um sistema.

### 1.1. Conceito

Um sistema distribuído é uma coleção de programas de computador que utilizam recursos computacionais em vários pontos centrais de computação diferentes para atingir um objetivo comum e compartilhado.



Também conhecido como computação distribuída ou bancos de dados distribuídos, ele depende de pontos centrais diferentes para se comunicar e sincronizar em uma rede comum. Esses pontos centrais costumam representar dispositivos de hardware físicos diferentes, mas também podem representar processos de software diferentes ou outros sistemas encapsulados recursivos. Os sistemas distribuídos visam remover gargalos ou pontos centrais de falha de um sistema.

## 1.2. Características

Compartilhamento de recursos: Um sistema distribuído pode compartilhar hardware, software ou dados

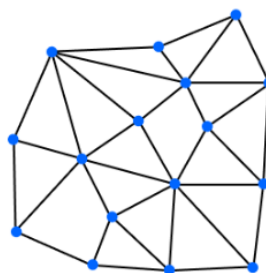
1. **Processamento simultâneo:** Várias máquinas podem processar a mesma função ao mesmo tempo
2. **Escalonamento:** A capacidade de computação e processamento pode evoluir conforme necessário quando estendida para máquinas adicionais
3. **Deteção de erros:** As falhas podem ser detectadas com mais facilidade
4. **Transparência:** Um ponto central pode acessar os e se comunicar com outros pontos centrais no sistema

## 1.3. Diferença entre um sistema centralizado e distribuído

Um sistema de computação centralizado é onde toda a computação é executada por um único computador em um único local. A principal diferença entre sistemas centralizados e distribuídos é o padrão de comunicação entre os pontos centrais do sistema. O estado de um sistema centralizado está contido em um ponto central que os clientes acessam de maneira personalizada. Todos os pontos centrais de um sistema centralizado acessam o ponto central, o que pode levar ao congestionamento e lentidão da rede. Um sistema centralizado tem um único ponto de falha, enquanto um sistema distribuído não tem um único ponto de falha.



Centralized



Distributed

## 1.4. Microsserviços são sistemas distribuídos?

Uma arquitetura de microsserviços é um tipo de sistema distribuído, pois decompõe um aplicativo em componentes ou “serviços” diferentes. Por exemplo, uma arquitetura de

microserviços pode ter serviços que correspondem a recursos de negócios (pagamentos, usuários, produtos etc.) em que cada componente correspondente lida com a lógica empresarial para essa responsabilidade. O sistema vai ter várias cópias redundantes dos serviços para que não haja um ponto central de falha para um serviço.

### 1.5. Benefícios, desvantagens e riscos dos sistemas distribuídos

Os sistemas distribuídos costumam ajudar a melhorar a confiabilidade e o desempenho do sistema. A confiabilidade é aprimorada removendo pontos centrais de falha e gargalos. Os pontos centrais de um sistema distribuído oferecem redundância para que, se algum ponto central falhar, haja outros pontos centrais prontos para cobrir e substituir a falha. O desempenho é aprimorado porque os pontos centrais podem ser escalados com facilidade no sentido horizontal e vertical. Se um sistema sofrer uma carga extensa, os pontos centrais extras podem ser adicionados para ajudar a absorver a carga. A capacidade de um ponto central individual [joseffe@unisanta.br](mailto:joseffe@unisanta.br) também pode ser aumentada para lidar com cargas extensas.

No entanto, a compensação para esses benefícios pode ser a “dispersão do desenvolvimento”, em que um sistema se torna muito complexo e a manutenção, desafiadora. À medida que um sistema cresce em complexidade, as equipes podem ter dificuldades para organizar, gerenciar e melhorar esses sistemas com eficiência. Parte do problema pode ser entender como os diferentes componentes se relacionam entre si ou quem tem um componente de software específico. Assim fica difícil entender como fazer alterações nos componentes para maximizar a integridade operacional e evitar o impacto negativo não apenas aos componentes dependentes, mas aos clientes.

## 2. ARQUITETURA DE SISTEMAS DISTRIBUÍDOS - PARTE 01

### Objetivo

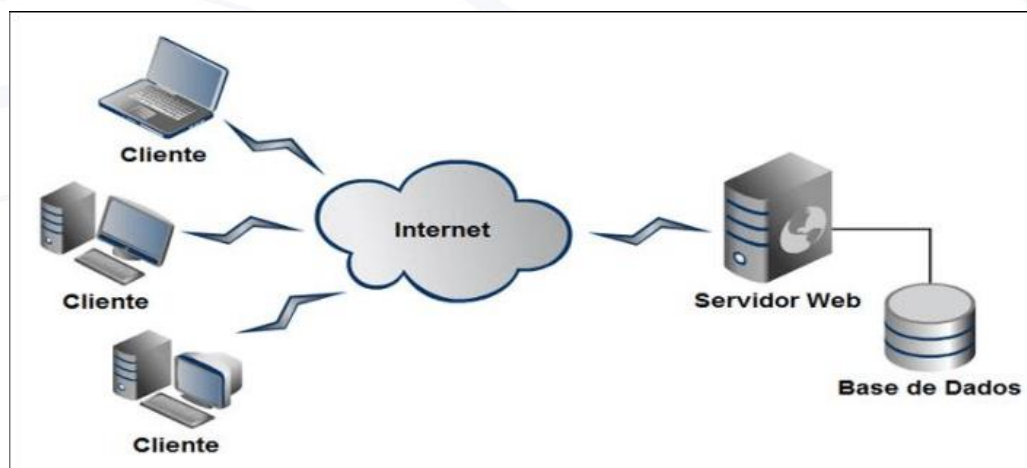
O objetivo deste capítulo é apresentar as principais arquiteturas de sistemas distribuídos.

### Introdução

A arquitetura de um sistema é sua estrutura em termos de componentes especificados separadamente e suas inter-relações. O objetivo global é garantir que a estrutura atenda às demandas atuais e, provavelmente, às futuras demandas impostas sobre ela. As maiores preocupações são tornar o sistema confiável, gerenciável, adaptável e rentável. O projeto arquitetônico de um prédio tem aspectos similares – ele determina não apenas sua aparência, mas também sua estrutura geral e seu estilo arquitetônico, fornecendo um padrão de referência coerente para seu projeto.

### 2.1. Cliente-servidor

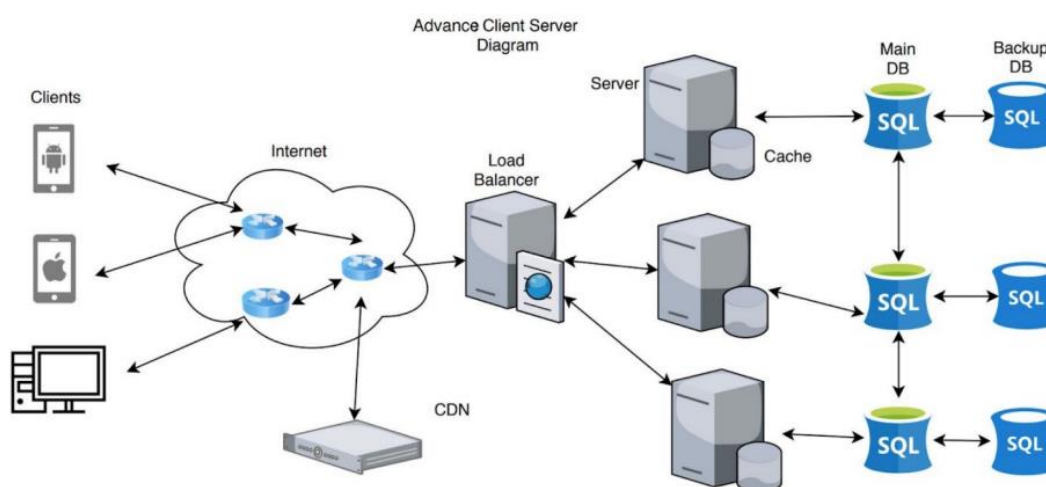
Uma arquitetura cliente-servidor é dividida em duas responsabilidades principais. O cliente é responsável pela apresentação da interface do usuário, que então se conecta pela rede ao servidor. O servidor é responsável por lidar com a lógica empresarial e o gerenciamento do estado. Uma arquitetura cliente-servidor pode ser degradada com facilidade para uma arquitetura centralizada se o servidor não for redundante. Uma configuração cliente-servidor de fato distribuída vai ter vários pontos centrais de servidor para distribuir as conexões do cliente. A maioria das arquiteturas cliente-servidor modernas são clientes que se conectam a um sistema distribuído encapsulado no servidor.





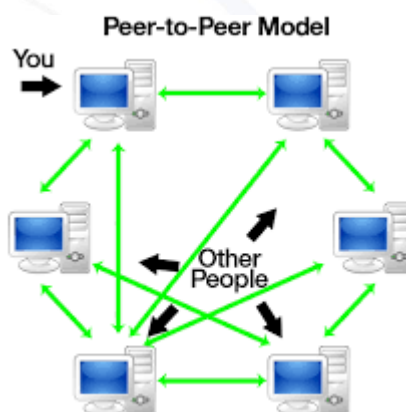
## 2.2. Vários níveis

Uma arquitetura de vários níveis expande a arquitetura cliente-servidor. O servidor em uma arquitetura de vários níveis é decomposto em pontos centrais granulares adicionais, que dissociam responsabilidades adicionais do servidor de back-end, como processamento e gerenciamento de dados. Esses pontos centrais adicionais são usados para processar trabalhos de longa duração sem sincronia e liberar os pontos centrais de back-end restantes para se concentrarem na resposta às solicitações do cliente e na interface com o armazenamento de dados.

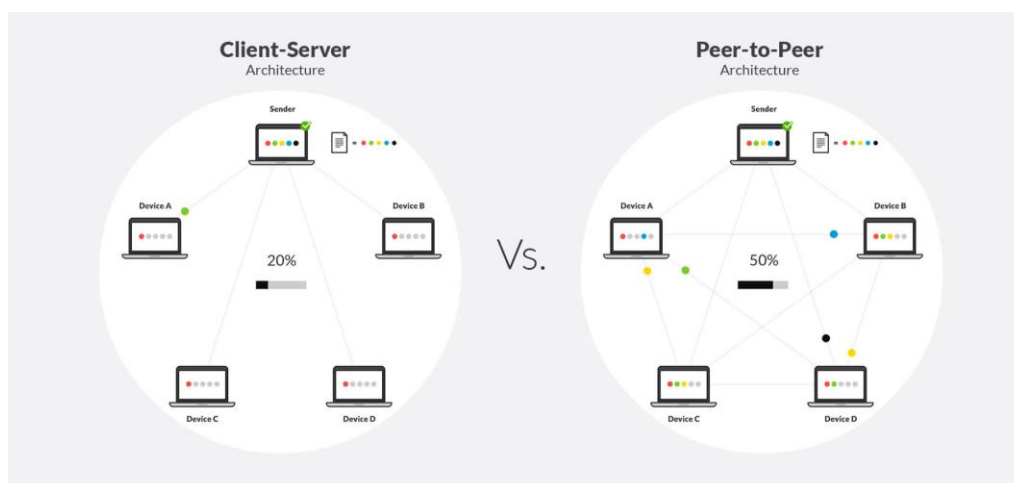


## 2.3. Par-a-par

Em um sistema distribuído par-a-par, cada ponto central contém a instância completa de um aplicativo. Não há separação de pontos centrais da apresentação e do processamento de dados. Um ponto central contém a camada de apresentação e as camadas de manipulação de dados. Os pontos centrais de mesmo nível podem conter todos os dados de estado de todo o sistema.

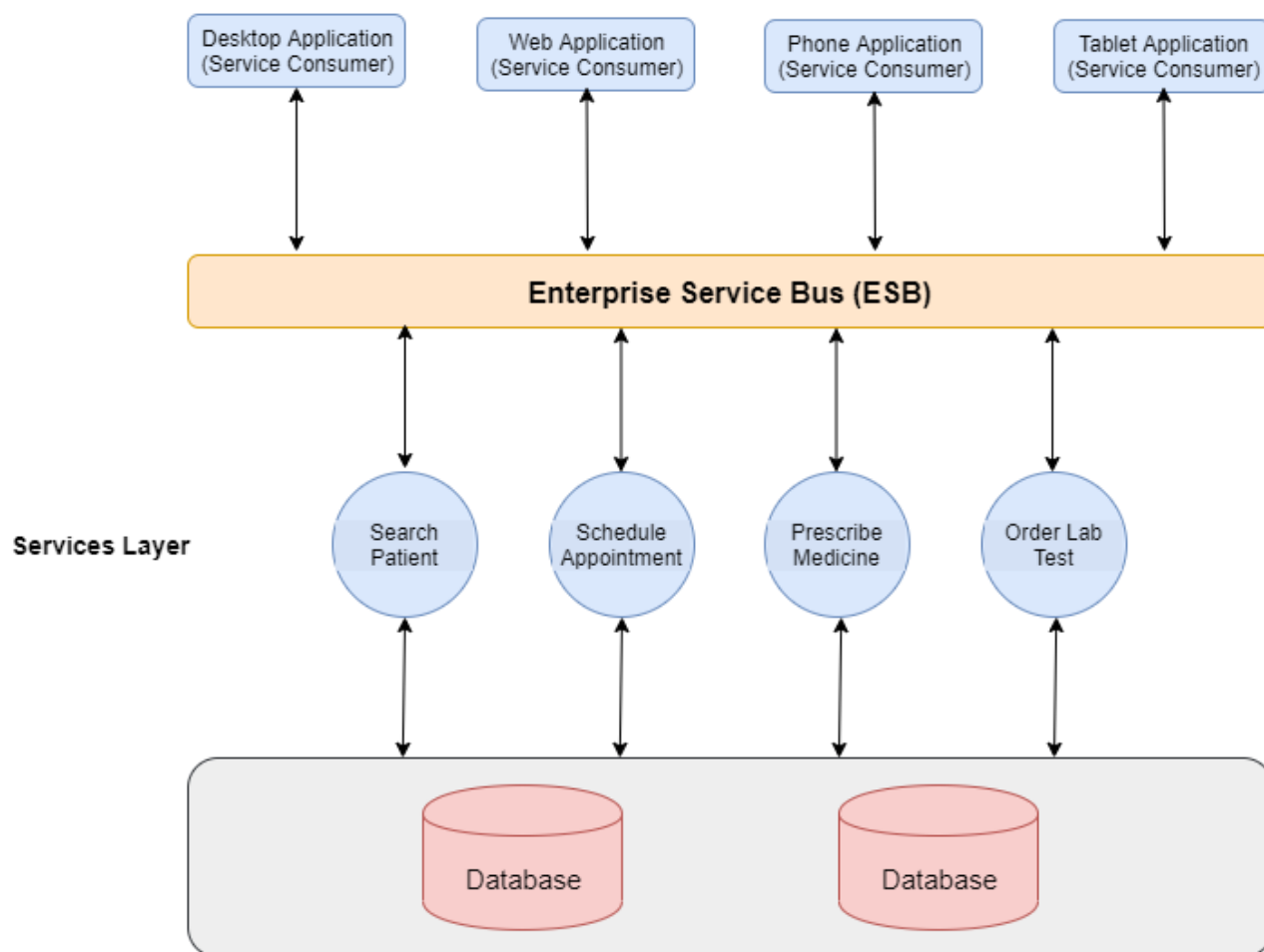


Os sistemas par-a-par têm o benefício de extrema redundância. Quando um ponto central par-a-par é inicializado e colocado on-line, ele descobre e se conecta a outros pares e sincroniza seu estado local com o estado do sistema maior. Com esse recurso, a falha de um ponto central em um sistema par-a-par não vai interromper nenhum dos outros pontos centrais, e o sistema par-a-par vai persistir.



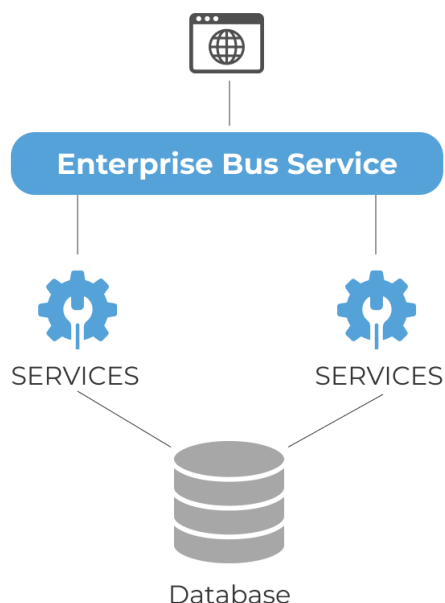
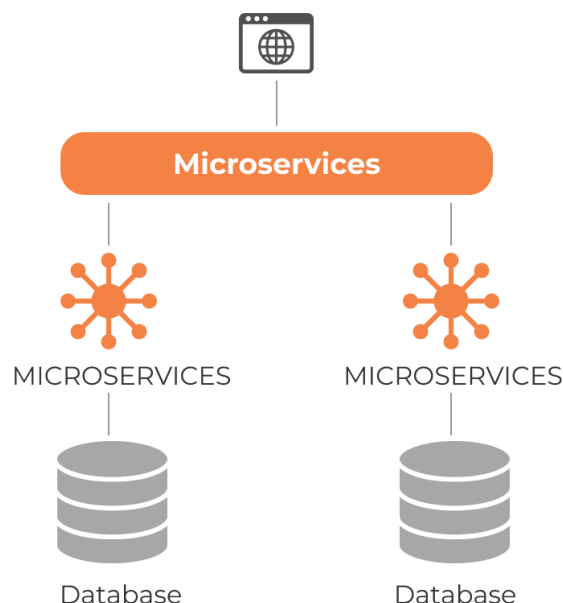
## 2.4. Arquitetura orientada a Serviço

A arquitetura orientada a Serviço (SOA) é uma antecessora dos microserviços. A principal diferença entre a SOA e os microserviços é o escopo do ponto central — o escopo dos pontos centrais de microserviço existe no nível do recurso. Em microserviços, um ponto central encapsula a lógica empresarial para lidar com um conjunto específico de recursos, como o processamento de pagamentos.

**Service Consumer Layer**

Os microserviços contêm vários pontos centrais de lógica empresarial diferentes que interagem com pontos centrais de banco de dados independentes. Em comparação, os pontos centrais da SOA encapsulam um aplicativo inteiro ou uma divisão corporativa. O limite de serviço para pontos centrais da SOA, em geral, inclui um sistema de banco de dados inteiro dentro do ponto central.

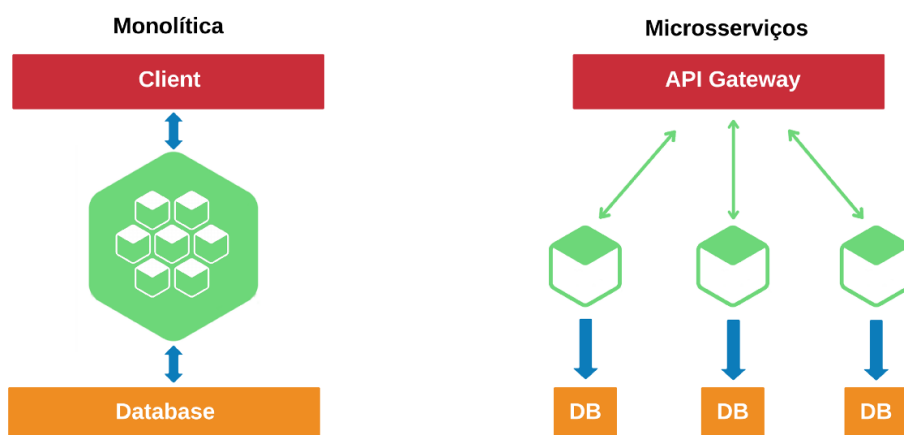


**Service Oriented Architecture****Microservices**

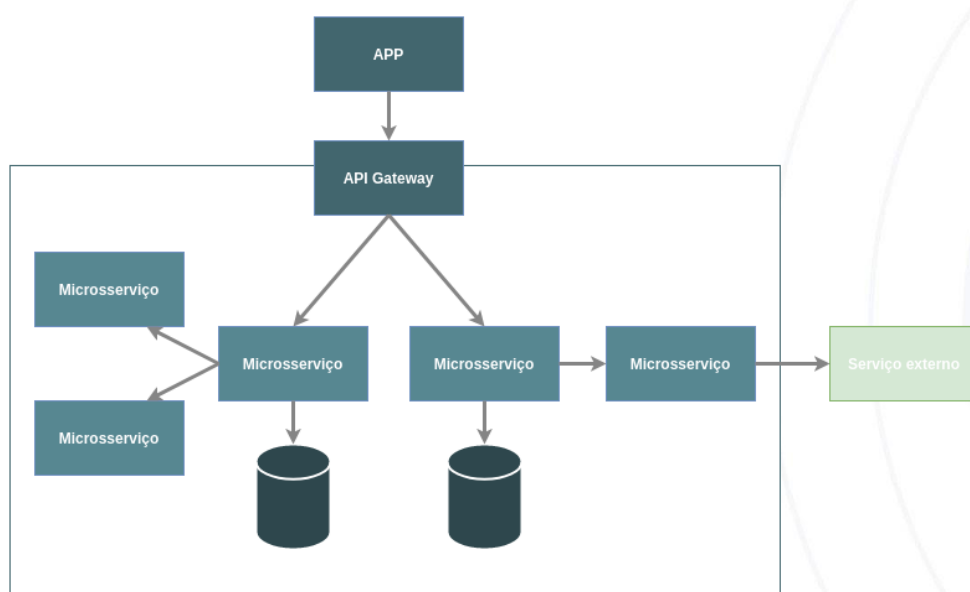
## 2.5. Arquitetura de microsserviços

Os microsserviços surgiram como uma alternativa mais popular a SOA devido aos seus benefícios. Os microsserviços são mais compostos, permitindo que as equipes reutilizem a funcionalidade oferecida pelos pequenos pontos centrais de serviço. Os microsserviços são mais robustos e permitem um escalonamento vertical e horizontal mais dinâmico.

Além disso, a arquitetura de micro serviços é utilizada para desenvolver uma aplicação como um conjunto de pequenos serviços, que funcionam com seu próprio processo. Cada serviço é desenvolvido em torno de um conjunto de regras de negócio específicas, e é implementado de forma independente.



### Arquitetura de Microserviços



### 3. ARQUITETURA DE SISTEMAS DISTRIBUÍDOS - PARTE 02

#### Objetivo

O objetivo deste capítulo é apresentar em detalhe a principal arquitetura de sistemas distribuídos, a arquitetura de microsserviços.

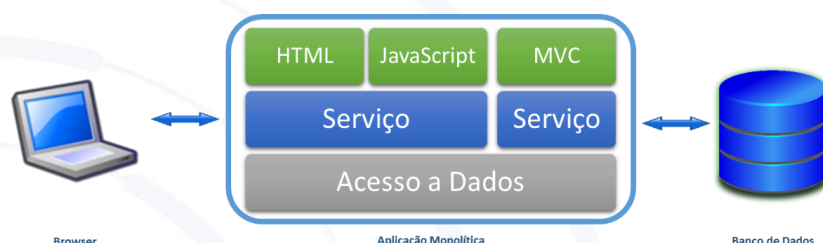
#### Introdução

O termo micro serviços foi criado em maio de 2011 durante uma conferência de arquitetos de software para representar um estilo de arquitetura de sistemas e, não exatamente, o tamanho dos serviços que a compõe, como o nome pode sugerir. A proposta da arquitetura orientada a micro serviços é desenvolver sistemas que sejam mais flexíveis, escaláveis e com manutenção mais simples do que as arquiteturas de sistemas monolíticos, que normalmente são utilizadas. Portanto, a principal filosofia desta arquitetura é “fazer uma coisa e fazê-la bem”, por isso os serviços são focados em realizar uma única função.

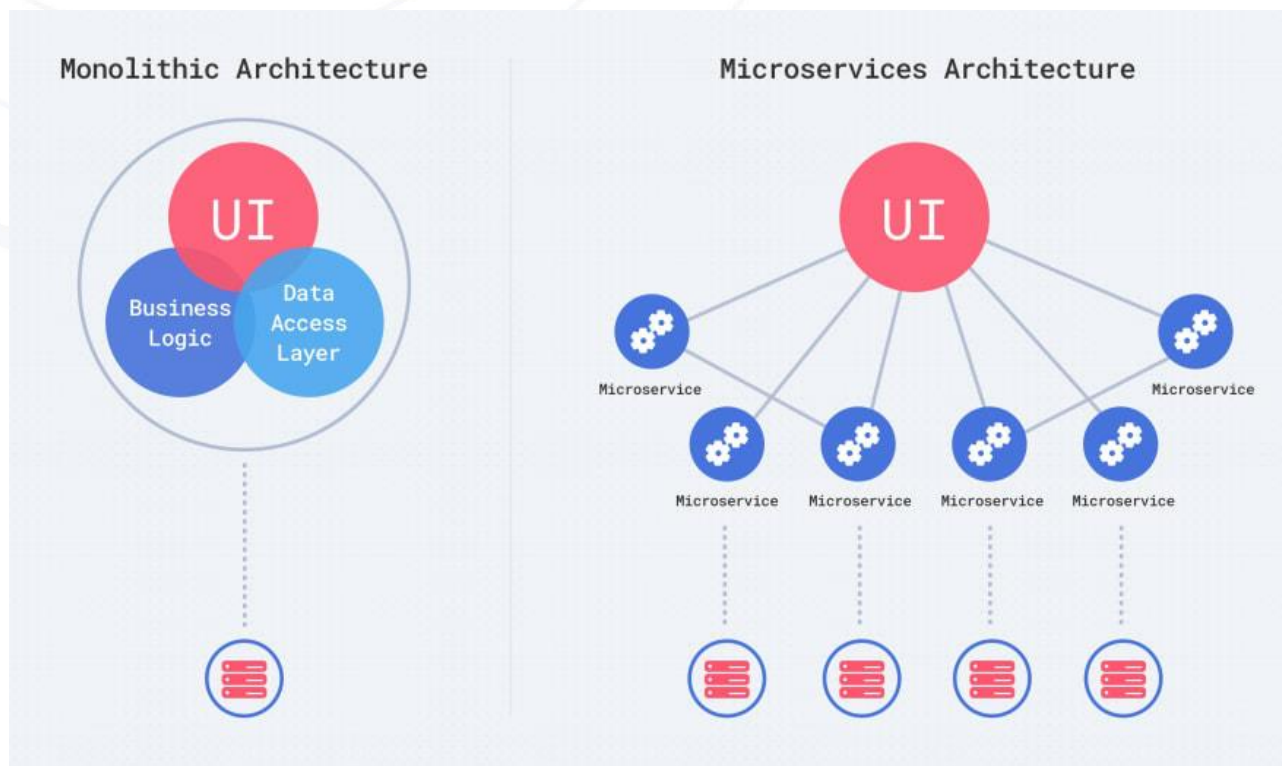
#### 3.1. Arquitetura monolítica

As principais linguagens de desenvolvimento de aplicações oferecem abstrações para quebrar a complexidade dos sistemas em módulos. Entretanto, são projetadas para a criação de um único executável/solução monolítica, no qual toda a modularização utilizada é executada em uma mesma máquina. Assim, os módulos compartilham recursos de processamento, memória, bancos de dados e arquivos.

Uma arquitetura monolítica típica de um sistema complexo pode ser representada pela figura abaixo, na qual todas as funções do negócio são implementadas em um único processo.



Podemos ver claramente a diferença comparada com uma arquitetura de microsserviço:



### 3.1.1. Desafios da arquitetura monolítica

Ao longo do tempo o sistema vai crescendo, se tornando mais complexo e consumindo cada vez mais recursos, o que acaba gerando também alguns desafios substanciais para a manutenção desse tipo de arquitetura. São eles:

#### **Aumento da complexidade e tamanho ao longo do tempo**

O sistema se torna tão complexo que a manutenção fica cada vez mais cara e lenta, porque os desenvolvedores têm que navegar em uma infinidade de códigos.

#### **Alta dependência de componentes de código**

Muitas funções são interdependentes e entrelaçadas, de forma que a inclusão ou manutenção de componentes do sistema pode causar inconsistências ou comportamentos inesperados.

#### **Escalabilidade do sistema é limitada**

Mesmo que apenas parte da funcionalidade seja necessária na nova instância, uma arquitetura monolítica exige que todo o sistema seja replicado, o que gera custos maiores do que o esperado.

### Falta de flexibilidade

Exige que os desenvolvedores fiquem amarrados à tecnologia originalmente escolhida para o sistema, mesmo que em algumas situações ela não seja a melhor escolha.

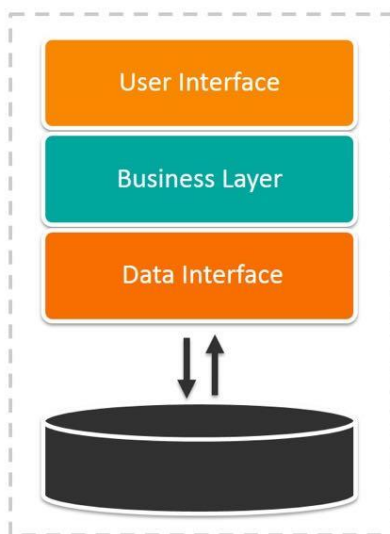
### Dificuldade para colocar alterações em produção

Qualquer mudança, por menor que seja, requer a reinicialização do sistema. Como isso pode causar algum risco operacional, é necessário que as equipes de desenvolvimento, testes e manutenção desse sistema acompanhem essas alterações.

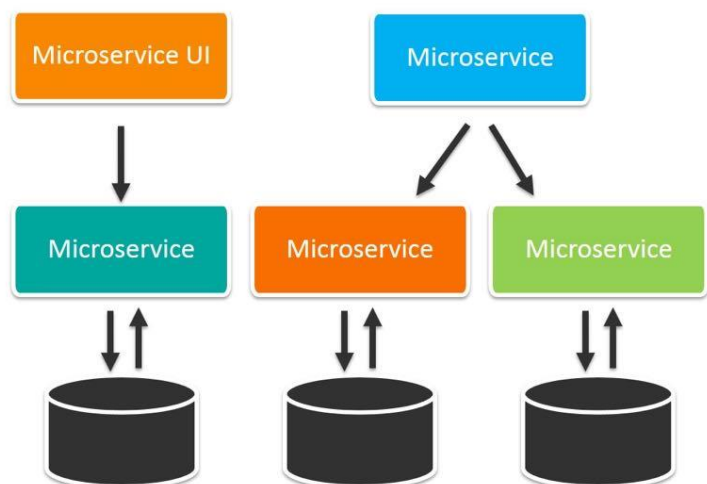
## 3.2. Arquitetura de microsserviços

A arquitetura de micro serviços é utilizada para desenvolver uma aplicação como um conjunto de pequenos serviços, que funcionam com seu próprio processo. Cada serviço é desenvolvido em torno de um conjunto de regras de negócio específicas, e é implementado de forma independente.

### Monolithic Architecture



### Microservices Architecture



Com isso, é possível quebrar algumas barreiras existentes no modelo de arquitetura monolítica.

### 3.2.1. Benefícios dos microsserviços

Manutenção e evolução dos serviços mais estáveis

Como os desenvolvedores vão trabalhar com códigos que executam uma única função, os serviços individuais não vão acompanhar o potencial crescimento do sistema, evitando que você precise carregar uma parte desnecessária da aplicação.

### **Serviços com baixo nível de acoplamento e interdependência**

A manutenção em um serviço específico não interfere diretamente nas outras funcionalidades do sistema.

### **Escalabilidade do sistema**

Os deploys e replicações de micro serviços são feitos por meio de infraestruturas de servidores, máquinas virtuais e containers que se organizam de forma independente. Isso torna o crescimento e a possibilidade de adaptação do sistema muito mais flexível.

### **Redução de custos**

Cada aplicação só utiliza os serviços que necessita. Por isso, você não vai ter gastos extras carregando funcionalidades não utilizadas, afinal os custos estão diretamente associados à funcionalidade e à carga de uso do sistema.

### **Flexibilidade de tecnologia**

Por conta do baixo acoplamento entre os serviços, não é necessário amarrar os desenvolvedores a uma tecnologia específica, o que te permite escolher a melhor opção para atender cada caso.

Isso diminui os riscos de obsolescência tecnológica e possibilita a evolução constante do sistema.

### **Facilidade de colocar alterações em produção**

As mudanças no sistema são executadas por meio de alterações e evoluções feitas nos serviços. Portanto, o sistema como um todo não precisa ser reinicializado para continuar funcionando.

Assim, o time de desenvolvimento que vai precisar acompanhar a mudança será apenas o de responsáveis pelos serviços que estão sendo alterados.



### 3.3. A importância de microserviços para os negócios

Naturalmente, faz parte dos objetivos das empresas acompanhar os avanços do mercado para evoluir constantemente, conquistar cada vez mais excelência, aumentar a agilidade na criação de novos produtos e serviços, criar diferenciais competitivos e, claro, reduzir custos.

Para que esses objetivos possam ser alcançados, a arquitetura das aplicações também precisa evoluir. Mas o que não pode faltar nessas aplicações?

#### **Foco na experiência do usuário**

Dinamismo e interatividade são conceitos que não podem ficar de fora quando o assunto é entregar valor para o usuário, independentemente da plataforma (o que inclui também os dispositivos móveis).

#### **Escalabilidade**

As aplicações desenvolvidas precisam estar altamente disponíveis. Além disso, deve ser possível executá-las em ambientes de nuvem.

#### **Atualizações suaves**

O ideal é que as atualizações não ocasionem paradas ou instabilidades que prejudiquem os usuários.

A arquitetura orientada a micro serviços viabiliza a construção de sistemas que possuem todas as características acima, diferentemente das aplicações monolíticas.

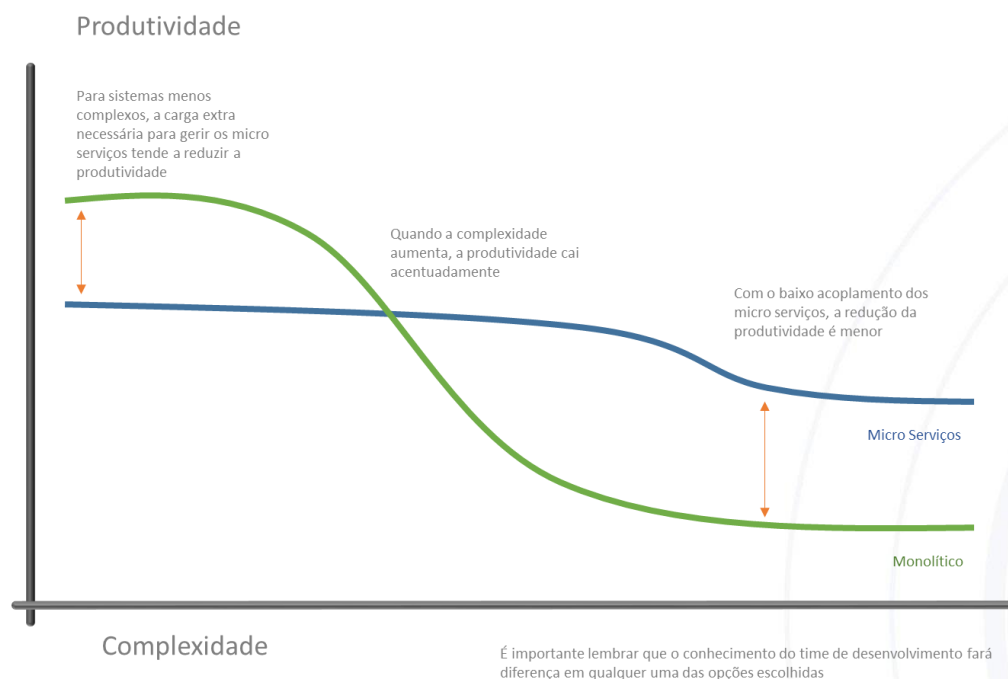
#### **Os micros serviços permitem:**

- Que novos processos ou serviços sejam disponibilizados sem impactar os processos e serviços existentes;
- Alterações em processos e serviços sem a necessidade de parada de todo o sistema;
- Otimização da utilização da infraestrutura de nuvem;
- Redução da complexidade de manutenção.

Dessa maneira, a implementação de novos serviços pelo time de desenvolvimento ao longo do ciclo de vida da aplicação se torna muito mais simples e rápida.

### 3.4. Fatores relevantes para adotar a arquitetura de micro serviços

Na análise de Martin Fowler, autor reconhecido na área de arquitetura de software, um sistema complexo que utiliza micro serviços tem um custo de manutenção menor do que o de aplicações com arquitetura monolítica, como exemplificado no gráfico a seguir:



Entretanto, existe uma barreira que pode impactar a adoção de uma arquitetura baseada em micro serviços: o custo na entrada, quando comparamos esse valor com o de um sistema monolítico. Além disso, eventualmente, pode existir a necessidade de um esforço extra para a gestão dos micros serviços que deve ser considerado também.

Os grandes players do mercado podem ver isso como uma oportunidade para fornecer ferramentas que auxiliem os times na gestão dos micros serviços, com o objetivo de diminuir a barreira de entrada para a adoção dessa arquitetura, tendo em vista os benefícios a médio e longo prazo que ela oferece.

Não podemos deixar de mencionar também os possíveis riscos da adoção da arquitetura de micro serviços traz. Afinal, como estamos falando de sistemas complexos sempre há pontos que demandam mais atenção, independente da arquitetura escolhida.

No caso dos micro serviços, é necessário prestar atenção no(a):

- Aumento da complexidade da coordenação;
- Comunicação entre os micro serviços;
- Governança;
- Portanto, para obter sucesso na adoção da arquitetura de micro serviços, esses fatores devem ser bem planejados e adequados para que a execução possa fazer parte da agenda da área de tecnologia da organização.

## 4. DEFINIÇÃO E OBJETIVOS DAS LINGUAGENS DE PROGRAMAÇÃO NO CONTEXTO DOS SISTEMAS DISTRIBUÍDOS

### Objetivo

O objetivo deste capítulo é apresentar as principais linguagens de programação e seus frameworks no contexto de desenvolvimento de sistemas distribuídos

### Introdução

Em termos gerais, a linguagem de programação é um conjunto de comandos e instruções digitais que usam sintaxes específicas para criar aplicativos de computador.

Por outro lado, frameworks são estruturas compostas por um conjunto de códigos genéricos que permite o desenvolvimento de sistemas e aplicações. Um framework funciona como uma espécie de template ou modelo que, quando utilizado, oferece certos artifícios e elementos estruturais básicos para a criação de alguma aplicação ou software.

### 4.1. Linguagens de programação

#### 4.1.1. *JavaScript*

Por quase uma década, o JavaScript emergiu como a linguagem de programação mais popular na Pesquisa Anual StackOverflow. Na recente Pesquisa para Desenvolvedores de 2020, 69,7% dos entrevistados escolheram JavaScript como a linguagem de programação mais amplamente usada.



JavaScript não é mais apenas uma linguagem de script do lado do cliente, porque tecnologias como Node.js permitem que você execute operações do lado do servidor. O

Node.js depende das estruturas do lado do servidor chamadas Express.js para criar uma plataforma que permite aos desenvolvedores escrever códigos que são executados no servidor.

Assim que o Express.js estiver instalado e funcionando com o Node.js, os desenvolvedores podem usar JavaScript como linguagem de desenvolvimento de front-end e back-end. Ele também oferece uma interface de programação de aplicativos (API) para a criação de vários aplicativos, incluindo aplicativos móveis, híbridos, da web, de uma e várias páginas.

### Características

- **Desenvolvimento rápido:** como a mesma linguagem é usada para o desenvolvimento de front-end e back-end, os programadores podem criar aplicativos da web e móveis mais rapidamente com JavaScript.
- **Menos sobrecarga de script:** Alguns recursos integrados do JavaScript, como DOM e hoops, aumentam a eficiência da codificação e melhoram o desempenho e menos sobrecarga de script.
- **Tecnologia Lean Backend:** JavaScript usa plataformas de backend como Express.js, que serve como um middleware para resolver vários desafios de desenvolvimento. Eles fornecem várias funções, como dados POST, cabeçalhos de sutura, tratamento de erros, registrador de solicitações HTTP e muito mais. Além disso, as regras de programação do JavaScript são menos rígidas, dando aos desenvolvedores mais liberdade para trabalhar com seu middleware preferido.
- **Corte de custos:** a eficiência do JavaScript reduz os esforços de programação e os custos necessários para desenvolver aplicativos. Uma vez que a mesma linguagem de programação é usada para criar códigos de front-end e back-end.
- **Vantagem de código aberto:** Ter uma comunidade vibrante que oferece suporte à tecnologia de programação é essencial para seu aprimoramento contínuo. Felizmente, JavaScript e suas contrapartes de back-end são

plataformas de código aberto com comunidades ativas de desenvolvedores de back-end e front-end que impulsionam inovações contínuas.

- **Recursos de I / O:** a natureza robusta do Express.js e do Node.js permite que ele lide com toneladas de solicitações de I / O e notificações de aplicativos conectados.

### Limitações

- As funções orientadas a eventos dos back-ends JavaScript são um tanto complicadas, fazendo com que os desenvolvedores com experiência em outras plataformas interpretem mal os retornos de chamada. Este desafio foi resolvido principalmente nas versões mais recentes.
- Muitos programadores que usam o desenvolvimento do lado do servidor JavaScript não entendem como o middleware funciona.
- Hospedar estruturas de back-end de JavaScript com banco de dados MySQL é complicado.
- Alguns desenvolvedores acham que a abordagem de design padrão para o desenvolvimento é preferível à liberdade que as estruturas de back-end JavaScript não definidas oferecem.

#### 4.1.2. Python

Desde que o Python foi criado em 1991 por Guido van Rossum, ele cresceu e se tornou uma das linguagens de programação multifuncionais líderes no mundo hoje. Os desenvolvedores de back-end utilizam seus códigos organizados e altamente legíveis para criar scripts funcionais para lidar com atribuições de back-end.



Uma pesquisa de 2020 feita por Stack Overflow revelou que é a principal linguagem de programação de back-end que os desenvolvedores esperam aprender. Python liderou



essa categoria por quatro anos consecutivos. Ele também foi classificado em terceiro lugar entre as linguagens de programação mais amadas existentes.

### Características

- **Relativamente fácil de aprender:** um dos benefícios do Python é o estilo de codificação semelhante ao inglês, que o torna altamente legível. Portanto, codificar e ler códigos Python é relativamente fácil para programadores novos e experientes.
- **Bibliotecas enormes:** Python conta com o suporte de bibliotecas enormes que reduzem a necessidade de escrever códigos manualmente. Algumas bibliotecas contêm códigos que aprimoram tarefas como e-mail, navegação, atribuições de banco de dados, teste de unidade e muito mais.
- **Eficiente em termos de custos:** além de Python ser uma plataforma de código aberto para download gratuito, ele também oferece várias ferramentas e recursos gratuitos que aprimoram os projetos de desenvolvimento de aplicativos.
- **Recursos de IoT:** os desenvolvedores podem aproveitar os recursos modernos do Python para criar objetos físicos Raspberry Pi.
- **Códigos incorporáveis:** com a regra Write Once Run Anywhere (WORA), o código python pode ser incorporado no código-fonte de outras linguagens, como C ++.

### Limitações

- A execução do código Python fica lenta quando é interrompida. Essa deficiência prejudica todo o projeto de desenvolvimento de aplicativos.
- A camada de acesso ao banco de dados Python é menos desenvolvida em comparação com outras linguagens de programação de back-end.
- Testes extensivos são necessários para detectar erros e bugs nos códigos Python.
- É altamente dependente de bibliotecas e estruturas de terceiros.

### 4.1.3. PHP

PHP foi desenvolvido por Rasmus Lerdorf em 1994; desde então, evoluiu para uma das principais linguagens de programação do lado do servidor atualmente. PHP é uma linguagem de programação multifuncional fácil de usar. Funciona perfeitamente com uma ampla variedade de bancos de dados e sistemas operacionais. Estruturas modernas, uma base de código enorme e a comunidade ativa são fatores que impulsionam a evolução contínua do PHP.



#### Características

- **Código aberto e versátil:** há muitas bibliotecas PHP online gratuitas que os desenvolvedores podem aproveitar para desenvolver códigos de back-end rapidamente. Quase todos os sistemas operacionais, como Windows e Linux, oferecem suporte a PHP. Além disso, os aplicativos PHP também podem ser iniciados em qualquer servidor web.
- **Econômico:** o PHP está disponível gratuitamente e conta com o suporte de uma vibrante comunidade de desenvolvedores. Contratar um desenvolvedor PHP não é caro. O 2020 Stack Overflow Developer Survey sobre as tecnologias mais bem pagas classificou o PHP da penúltima posição.
- **Fácil de usar:** surgiram muitos frameworks PHP que simplificam a programação, eliminando a necessidade de escrever códigos SQL tediosos. Alguns deles usam o sistema Object Relational Mapping (ORM) que funciona como o model-view-controller (MVC) para escrever funções do lado do servidor rapidamente.

- **Excelente para iniciantes:** a simplicidade do PHP o torna uma linguagem ideal para novos desenvolvedores. Eles podem começar a funcionar rapidamente devido à curta curva de aprendizado.
- **Funções de automação:** o recurso de script do PHP o torna útil para a criação de automação, como autenticação, mapeamento de URL, gerenciamento de sessão e muito mais.
- **Segurança embutida:** Embora muitas pessoas pensem que o PHP não é seguro, ele tem muitos recursos de segurança embutidos que permitem a você mitigar várias ameaças.

### Limitações

- A influência do PHP como tecnologia de desenvolvimento está diminuindo. Hoje em dia, as pessoas dificilmente incluem PHP em suas habilidades de desenvolvimento.
- PHP não pode competir de forma eficiente com tecnologias de desenvolvimento modernas como Ruby e Python devido a bibliotecas limitadas.
- Como uma plataforma de código aberto, o PHP é suscetível ao uso indevido e à criação de códigos com erros.

#### 4.1.4. Java

O índice 2021 TIOBE classifica Java em segundo lugar entre as tecnologias de desenvolvimento de backend hoje. Essa classificação indica que Java é uma das linguagens de programação mais influentes hoje. James Gosling inventou a linguagem de programação em 1991, mas não foi até 1995 que a Sun Microsystems a publicou.



Ao longo dos anos, o Java surgiu como a plataforma de escolha para desenvolvedores que preferem criar aplicativos da web com recursos personalizados e

inovadores. Java também é útil para desenvolvimento móvel e aplicativos em dispositivos incorporados.

### Características

- **Escalável e direto:** Java Enterprise Edition é útil para criar aplicativos escalonáveis, permitindo que o servidor execute várias instâncias. Os componentes Java estão prontamente disponíveis e a sintaxe é fácil de entender. Todos esses recursos o tornam excelente como uma tecnologia de desenvolvimento de backend.
- **Multi-Threading:** Java é capaz de lidar com solicitações em threads independentes em um servidor web multi-thread. É por isso que o Java tem um desempenho excelente com aplicativos que exigem grande capacidade de CPU.
- **Bibliotecas de código aberto massivas:** os desenvolvedores podem aproveitar bibliotecas de código aberto massivas para agilizar as tarefas de desenvolvimento do lado do servidor. Algumas dessas bibliotecas incluem testes de unidade, excel, mensagens, análise JSON e muito mais.
- **Segurança aprimorada:** Java é conhecido por sua segurança rígida; ele fornece vários recursos que reduzem os riscos de segurança. Por exemplo, a Java Virtual Machine verifica os bytecodes java para manter os vírus afastados. Outros recursos que aumentam a segurança são o modelo de segurança Java e o teste de código reutilizável.

### Limitações

- A programação Java é cara e demorada
- A programação de baixo nível e o comando para coleta de lixo estão visivelmente ausentes
- O alto custo dos requisitos de hardware torna cara a implementação de back-ends baseados em java
- O kit de ferramentas Swing, o aplicativo de desenvolvimento Java GUI, carece da interface intrigante de outras ferramentas GUI da moda

#### 4.1.5. C#

Uma das linguagens de programação mais populares para o desenvolvimento de back-end é C #, comumente chamada de C-Sharp. Ele faz um excelente trabalho na automação de codificação em servidores Windows e plataformas web (usando o framework ASP.Net.) C-Sharp, que é mais ou menos uma extensão do C ++, já existe há muito tempo.



C # encontra ampla aplicação no desenvolvimento de aplicativos de desktop e também em sistemas incorporados. Ele executa o código mais rápido do que a maioria das linguagens de programação. C # foi usado para criar plataformas semelhantes ao Unity para desenvolvimento de jogos e para criar aplicativos CLI.

#### Características

- **Desenvolvimento de plataforma cruzada:** os aplicativos com back-ends C-Sharp podem ser executados em vários sistemas operacionais, como macOS, Windows, Linux, etc. Este artigo sobre estruturas de plataforma cruzada discute o desenvolvimento de plataforma cruzada mais adiante.
- **Vantagem orientada a objetos:** Como uma linguagem orientada a objetos, o código C # faz uso de classes e relacionamentos. Essa abordagem permite a reutilização de trechos de código e fácil solução de problemas de código.
- **Compatibilidade extensiva:** aplicativos C # apresentam compatibilidade com sistemas legados. Por exemplo, organizações que ainda usam versões mais antigas de estruturas de programação considerariam C # inestimável.
- **Recurso de coleta de lixo:** C-Sharp tem um recurso que elimina todos os resíduos do sistema. Este é um ótimo recurso que acelera a execução do programa. A melhor parte é que o sistema continua a funcionar perfeitamente durante o procedimento de coleta de lixo.

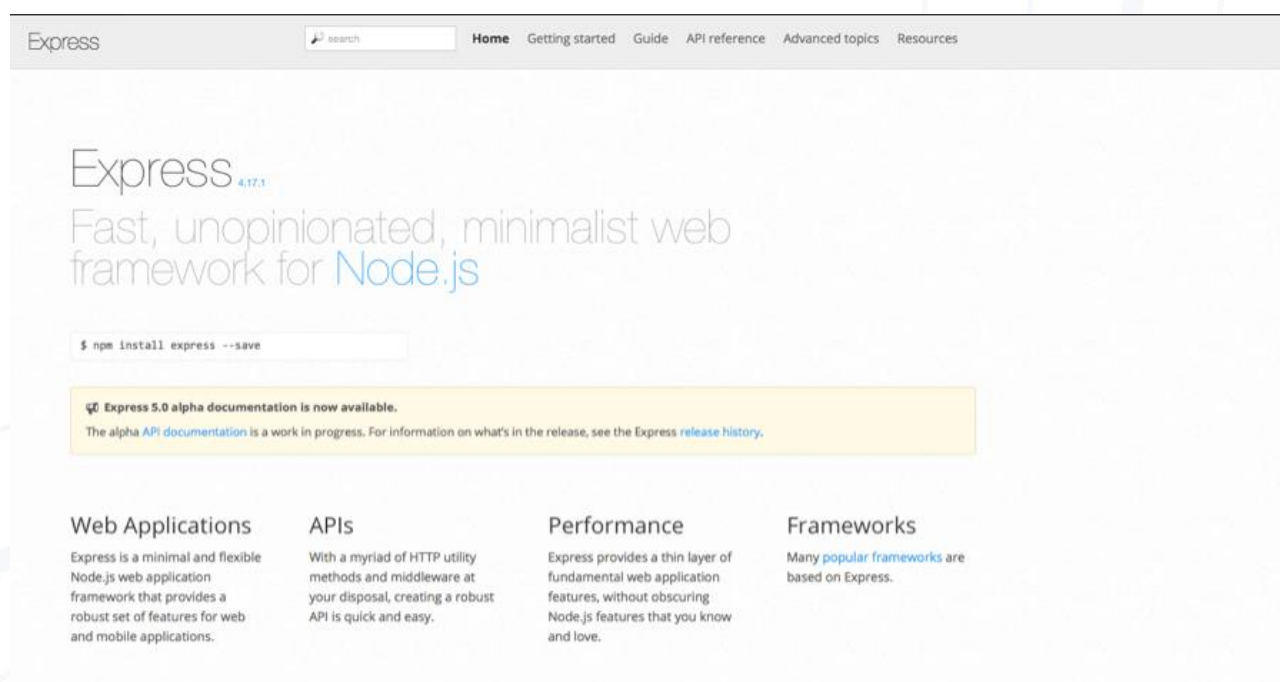
#### Limitações

- Como uma linguagem de programação de alto nível, o código C # não pode se comunicar diretamente com o hardware.
- C # é altamente rígido em comparação com outras tecnologias de back-end porque ele só funciona no .Net Framework e só pode ser instalado e executado em um computador Windows.

## 4.2. Frameworks

### 4.2.1. Express (JavaScript)

Express.js, também conhecido como Express, é uma estrutura de aplicativo da web Node.js e software de código aberto disponível sob a licença do MIT. Ele é usado para construir APIs e aplicativos da web e é considerado uma estrutura de servidor Node.js padrão. Express é um componente de back-end da pilha MEAN junto com a estrutura de front-end AngularJS e bancos de dados.



### Vantagens do Express JS

- **Curva de aprendizado fácil:** JavaScript está entre as linguagens de programação mais usadas, e a maioria dos desenvolvedores front-end confia nele. Os desenvolvedores podem facilmente começar a utilizar o back-end do Node.js, pois aprendê-lo não requer muito esforço ou tempo.



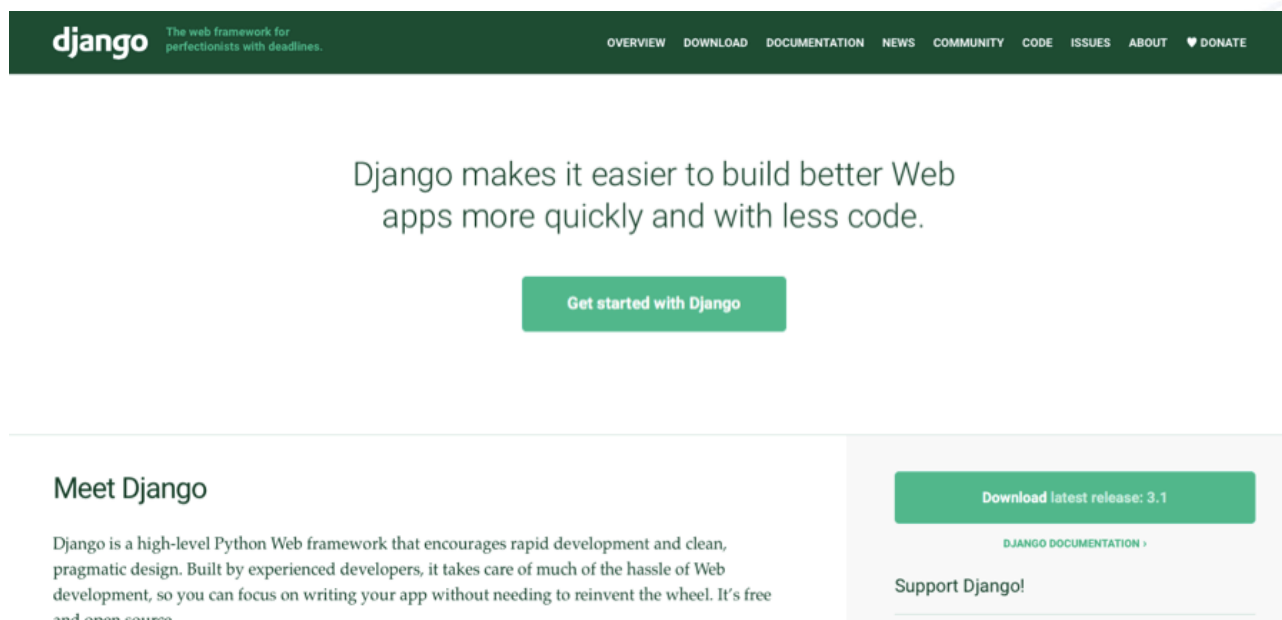
- **É uma linguagem de programação única:** Node.js permite que os desenvolvedores usem JavaScript para escrever aplicativos do lado do servidor. Ele permite que os desenvolvedores Node.js criem os aplicativos da web de back-end e front-end, utilizando um ambiente de tempo de execução JavaScript. Os desenvolvedores não são obrigados a utilizar nenhuma outra linguagem do lado do servidor. A implantação de aplicativos da Web torna-se muito mais simples, pois os navegadores mais amplamente usados suportam JavaScript.
- **Utiliza recursos JS de stack completa:** Node.js como backend é um JavaScript de stack completa para aplicativos do lado do servidor e do lado do cliente. Como resultado, não há necessidade de depender de desenvolvedores separados para o desenvolvimento de front-end e back-end. Usá-lo é uma ótima maneira de economizar tempo e dinheiro.
- **Oferece alto desempenho:** Node.js usa o mecanismo V8 JavaScript do Google para interpretar Node.js. O mecanismo facilita o código baseado em JavaScript em código de máquina e torna mais fácil implementar o código de forma eficaz. O ambiente de tempo de execução também melhora a velocidade de execução, pois o JavaScript oferece suporte a operações de I/O sem bloqueio.

### Recursos do Express JS

- **Programação rápida do lado do servidor:** Express.js é uma estrutura Node.js que oferece muitos dos recursos mais amplamente usados do Node.js. Esses recursos podem ser utilizados em diferentes pontos do programa. Os desenvolvedores Express JS podem incorporar facilmente algumas linhas de código instantaneamente em vez de escrever grandes volumes de código. O desenvolvimento de aplicativos da Web com Express é mais rápido do que apenas com Node.js.
- **Roteamento:** o roteamento é um recurso que permite que os aplicativos da web retenham estados de páginas da web por meio de URLs. Os URLs podem ser compartilhados com outras pessoas e os usuários podem visitar os URLs para acessar a página de armazenamento de estado. O Node.js oferece um mecanismo de roteamento fundamental, em comparação ao Express.js, que fornece um mecanismo de roteamento mais sofisticado, capaz de lidar com URLs dinâmicos.
- **Depuração:** Bugs são bastante comuns durante projetos de desenvolvimento e podem causar mau funcionamento em grande escala dos aplicativos. Os desenvolvedores precisam identificar as fontes dos bugs e corrigi-los sem demora. Express.js oferece um sistema de depuração conveniente para permitir que os desenvolvedores identifiquem facilmente as causas dos bugs do aplicativo.

#### 4.2.2. Django (Python)

Django é um framework backend de código aberto baseado na linguagem de programação Python. Ele segue o padrão model view controller (MVC). Django é adequado para o desenvolvimento de sites sofisticados e ricos em recursos baseados em banco de dados.



Essa estrutura de backend facilita a conectividade ideal, codificação reduzida, maior capacidade de reutilização e desenvolvimento mais rápido. Ele usa Python para todas as operações no Django e fornece uma interface de administração opcional para ajudar a criar, ler, atualizar e excluir operações. Django é utilizado por muitos sites renomados, como Disqus, Mozilla e The Washington Times.

#### Vantagens do Django

- **Velocidade:** Django é fácil de usar e uma estrutura de baixa curva de aprendizado criada para ajudar os desenvolvedores a acelerar todo o processo de desenvolvimento do início ao fim.
- **Rico em recursos:** o Django fornece uma ampla variedade de recursos para ajudar os usuários a cuidar de alguns requisitos comuns de desenvolvimento da web. Ajuda em tarefas como autenticação de usuário, sitemaps, administração de conteúdo e muito mais.
- **Segurança ideal:** Django é uma estrutura segura que ajuda seus usuários a prevenir vários problemas de segurança, incluindo cross-site scripting,

clickjacking, injeção de SQL e falsificação de solicitação. Ele fornece um sistema de autenticação de usuário para permitir que os usuários armazenem e gerenciem senhas e contas com segurança.

- **Alta escalabilidade:** Django oferece um alto nível de escalabilidade para seus usuários. É por isso que muitos dos principais sites do mundo contam com ele para atender às suas altas demandas operacionais facilmente.
- **Framework versátil:** Django é um framework que pode ser usado para o desenvolvimento de uma ampla variedade de tipos de aplicativos. Alguns deles incluem aplicativos de rede social, sistemas de gerenciamento de conteúdo e plataformas de computação.

### Recursos do Django

- **Código aberto:** Django é uma estrutura de código aberto usada para aplicativos da web baseados em Python. É simples, fácil de usar e confiável.
- **Sistema de nomenclatura:** Django apresenta seu próprio sistema de criação para todas as ferramentas e recursos. Além disso, ele também tem um painel de administração fácil de usar em comparação com Yii e Lavarel.

Alguns dos principais recursos do Django incluem sintaxe simples, arquitetura central MVC, ORM (Object Relational Mapper), suporte a Middleware e bibliotecas HTTP. Django também tem seu próprio servidor web, uma estrutura para teste de unidade Python e componentes necessários para resolver alguns casos.

#### 4.2.3. *Laravel (PHP)*

Lavarel é uma estrutura da web PHP de código aberto para o desenvolvimento de aplicativos da web baseados em Symfony que seguem a arquitetura model – view – controller (MVC).



Documentation

Vapor

Ecosystem ▾

News

Partners

Search Docs

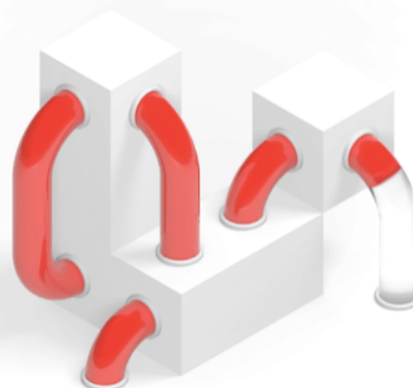
Laravel Vapor is now available! Sign up today! →

## The PHP Framework for Web Artisans

Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.

Documentation

Watch Laracasts



Ele oferece um sistema de empacotamento modular equipado com um gerenciador de dependências dedicado. O Laravel também fornece a seus usuários várias maneiras de acessar bancos de dados relacionais, juntamente com manutenção de aplicativos e utilitários de implantação. O Laravel possui uma licença MIT e um código-fonte hospedado no GitHub.

### Vantagens do Laravel

- **Autenticação:** A implementação da autenticação é bastante simples com o Laravel, pois oferece fácil configuração. O Laravel facilita a organização de lógica de autorização simples e fácil controle de recursos de acesso.
- **API simples:** o Laravel oferece uma API simples que funciona de maneira fluida com a biblioteca SwiftMailer. Laravel oferece drivers para Mandrill, SMTP, Mailgun, Amazon SES e SparkPost. Ele também possui drivers para e-mail PHP e envia e-mails. O Laravel permite o envio rápido de e-mails de aplicativos através de um serviço local ou baseado em nuvem. Ele também fornece suporte de envio de notificação em vários canais de entrega.
- **Cache:** o Laravel oferece suporte para Redis, Memcached e outros backends de cache amplamente usados. Ele usa o driver de cache de arquivo, que executa o armazenamento de objetos em cache em um sistema de arquivos. Aplicativos maiores usam um cache na memória como APC ou Memcached. O Laravel também permite que os usuários façam várias configurações de cache.

- **Logs:** Um projeto Laravel tem uma exceção pré-configurada e tratamento de erros. O Laravel também possui integração com a biblioteca de registro Monolog e oferece suporte para múltiplos manipuladores de registro.
- **Teste:** Laravel é amigável para teste e oferece suporte a PHP Unit junto com um arquivo phpunit.xml pronto para uso para aplicativos. A estrutura vem com métodos auxiliares para testes de aplicativos expressivos. O Laravel também fornece simulação fácil do comportamento do usuário para atividades como preenchimento de formulários, cliques em links e solicitações de aplicativos.

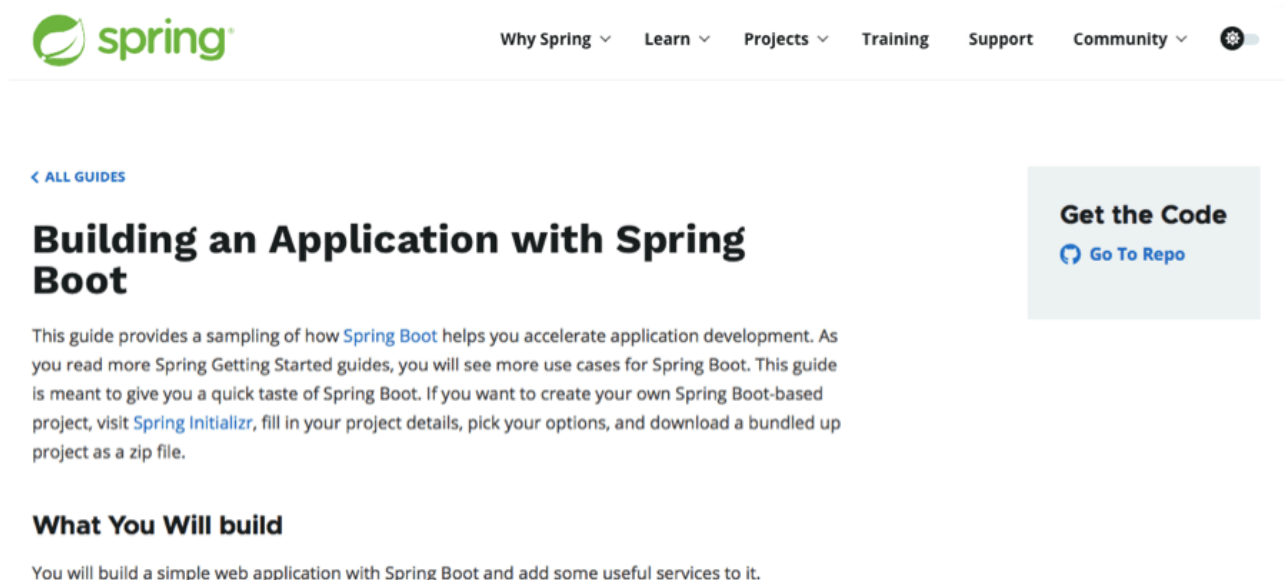
### Recursos do Laravel

- **Template Engine:** A estrutura do Laravel apresenta modelos integrados leves que podem ser utilizados para gerar layouts e propagação de conteúdo. Ele também fornece widgets com código JS e CSS. Os templates do Laravel são projetados para o desenvolvimento de layouts seccionados simples e complexos.
- **Suporte à Arquitetura MVC:** O Laravel oferece suporte ao padrão de arquitetura MVC para separar efetivamente as camadas de apresentação e lógica de negócios. O Laravel MVC oferece muitas funções, facilita um melhor desempenho e melhora a escalabilidade e segurança.
- **Eloquent Object Relational Mapping:** Os usuários do Laravel podem utilizar o Eloquent Object Relational Mapping (ORM), incluindo uma implementação simples de Active Record de PHP. ORM permite que os desenvolvedores de aplicativos construam consultas de banco de dados usando a sintaxe PHP sem escrever código SQL. Um ORM é comparativamente mais rápido do que outros frameworks PHP.
- **Segurança:** O Laravel framework fornece segurança robusta para aplicações web, com modalidades de senha com sal e hash. Como resultado, as senhas não são armazenadas em formato de texto simples nos bancos de dados. O algoritmo de hash Bcrypt também é utilizado pelo Laravel para geração de senha criptografada. Esta estrutura de desenvolvimento utiliza instruções SQL preparadas para reduzir as eventualidades de ataques de injeção.



#### 4.2.4. Spring Boot (Java)

Spring Framework é uma estrutura de aplicativo de código aberto e o recipiente de inversão de controle da plataforma Java. Os aplicativos Java podem utilizar os principais recursos desta estrutura. Os usuários também podem usar muitas extensões para criar aplicativos da web baseados na plataforma Java EE.



### Vantagens Spring Boot

- Suporte integrado para Undertow, Jetty e Tomcat
- A configuração padrão não é necessária
- O reinício automático do servidor é para atualizações de código e configuração é facilitado por DevTools
- O gerenciamento de dependências é mais fácil
- Propriedades específicas do perfil são facilmente gerenciadas
- As propriedades do aplicativo são facilmente personalizadas
- SpringBoot Starters são convenientes para desenvolvedores

### Recursos Spring Boot

- **Inicialização:** SpringBoot ajuda os desenvolvedores a realizar uma inicialização lenta. Ativar esse recurso ajuda os desenvolvedores a criar beans com base nos requisitos, em vez de durante a inicialização de um aplicativo. Portanto, a inicialização lenta pode reduzir o tempo necessário para o aplicativo iniciar.
- **Personalização de banner:** os banners de inicialização podem ser modificados pelos usuários adicionando um arquivo banner.txt ao seu classpath. Banners também podem ser modificados apontando a propriedade spring.banner.location para o local do arquivo relevante. Os usuários podem definir um spring.banner.charset para arquivos usando codificação fora de UTF-8. Os usuários



podem adicionar imagens banner.jpg, banner.gif e banner.png junto com arquivos de texto a um classpath. Eles também podem definir spring.banner.image.location.

- **Construtor de APIs:** os desenvolvedores podem usar SpringApplicationBuilder se precisarem construir uma hierarquia ApplicationContext ou utilizar sua API de construtor fluente.
- **Estado de atividade:** O estado de atividade de um aplicativo é responsável por notificar se seu estado interno permite que ele funcione ou processe a recuperação em caso de falha. Se um estado de ativação for interrompido, o aplicativo estará em um estado irreversível e sua infraestrutura reiniciará o aplicativo.

#### 4.2.5. ASP.NET Core (C#)

ASP.NET Core é uma estrutura de código aberto e gratuita que segue os passos do ASP.NET, um back-end amplamente usado criado em parceria com a .NET Foundation. ASP.NET Core é uma estrutura modular que pode ser executada em todo o .NET Framework no Windows e .NET Core.



#### Vantagens ASP.NET Core

- **Suporte de plataforma cruzada:** O desenvolvimento de aplicativos da Web requer que os desenvolvedores garantam que um aplicativo forneça suporte para todas as plataformas. O novo ASP.NET Core é uma estrutura de back-end de aplicativo da Web de plataforma cruzada que oferece suporte para várias plataformas. ASP.NET Core é uma solução de plataforma cruzada para desenvolver aplicativos da Web para as plataformas Windows, Mac e Linux. O back-end usa o mesmo código C # em todas as plataformas.
- **Codificação mínima:** O ASP.NET Core usa tecnologia que requer menos codificação. Isso significa que os desenvolvedores acham o uso do back-end

bastante conveniente, pois precisam construir menos instruções. Menos codificação se traduz em menos tempo necessário para criar um aplicativo. Como resultado, os desenvolvedores exigem menos tempo para criar um aplicativo e o processo também é econômico.

- **A manutenção é fácil:** Menos código também significa menos manutenção. O ASP.NET Core pode ser mantido automaticamente em casos com uma pequena quantidade de código. Desenvolvedores experientes podem facilmente obter uma noção sobre como reduzir o esforço de manutenção para um back-end ASP.NET Core. Eles podem otimizar o código ASP.NET com apenas algumas instruções.
- **Melhor desempenho:** A maior vantagem de usar a estrutura ASP.NET Core é a melhoria de desempenho que ela oferece. Usar atualizações e as melhorias mais recentes ajuda os desenvolvedores a melhorar o código e aprimorar o desempenho de um aplicativo. O desempenho também é alto, pois os usuários não precisam alterar o código. O compilador embutido do ASP.NET é capaz de aprimorar o código quando a estrutura do ASP.NET Core é recompilada com o código. O desempenho da estrutura é muito mais do que suas alternativas.

### Funcionalidades ASP.NET Core

- **Plataforma cruzada e suporte para contêineres:** O ASP.NET Core permite que os desenvolvedores criem aplicativos que podem ser implantados em plataformas Windows, macOS e Linux. O backend é mais adequado para a plataforma Linux.
- **Assíncrono:** ASP.NET Core oferece aos desenvolvedores a opção de usar padrões de programação assíncrona. Async é uma implementação comum em todas as classes do .NET Framework e muitas bibliotecas externas. Muitos aplicativos delegam uma quantidade significativa de tempo e ciclos de CPU para chamadas de serviço da web, consultas de banco de dados e operações de I/O.
- **Alto desempenho:** O desempenho é um dos principais recursos da estrutura de back-end do ASP.NET Core. O servidor Web Kestrel e o ASP.NET Core disponíveis para desenvolvedores agora tornam o ASP.NET uma das estruturas

de aplicativos Web mais ágeis. O desempenho é um dos maiores fatores que torna o ASP.NET Core a estrutura de aplicativo da Web preferida de vários desenvolvedores.

## 5. INTERCÂMBIO DE DADOS - PARTE 01

### Objetivo

O objetivo deste capítulo é apresentar o conceito e as características de intercâmbio de dados no contexto de redes de comunicação e protocolos.

### Introdução

Redes de computadores são formadas por uma série de conexões realizadas entre diversos dispositivos que têm a função de trocar recursos e dados, conectando-se entre si. O que nós conhecemos como internet é, justamente, um tipo de rede de computador: o único que abrange o mundo inteiro. Esse tipo de conexão permite, justamente, as trocas de dados entre diversos dispositivos. Ela possibilita, por exemplo, que consigamos acessar, enviar e receber documentos, entrar em aplicativos, abrir redes sociais, entre outros.

Para que essa conexão ocorra, é importante o uso de protocolos que sustentam esse tipo de comunicação. Um dos mais importantes, e que permitiu essas ações, é o TCP/IP, viabilizando a conexão entre diferentes dispositivos, além do HTTP também.

### 5.1. Tipos de arquitetura de redes de comunicação

#### PAN

A rede PAN é bastante restrita, normalmente, às áreas domésticas de uma residência. É a sigla para Personal Area Network. Nesse caso, conecta-se uma série de dispositivos que operam segundo o tipo de conexão estabelecido (Bluetooth, USB etc).

Contudo, a diferença está no centro do número de pessoas envolvidas. **A rede PAN, normalmente, trata-se de uma única pessoa utilizando os diversos dispositivos da rede**, enquanto na LAN doméstica, temos um maior número de usuários envolvidos.

#### LA

LAN é a sigla para Local Area Network, ou seja, é uma rede formada por dispositivos que estejam dentro da mesma área física. **Normalmente, é muito utilizada em espaços pequenos e que não tenham demandas altas de contato externo**. Assim, é a opção mais comum para escolas, escritórios pequenos, residências, entre outros.

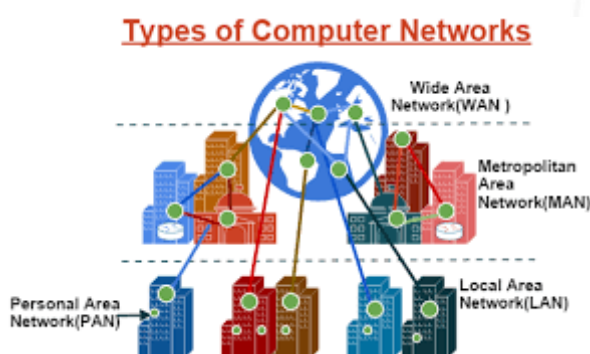
## MAN

MAN é a sigla para Metropolitan Area Network, ou seja, é uma rede formada por **dispositivos que estão na mesma área de abrangência, mas dentro de um espaço maior, ou seja, em uma região metropolitana**. É muito utilizada, principalmente, para oferecer conexões entre unidades que estão localizadas em uma mesma cidade.

Normalmente, é adotada em **escritórios que tenham mais de uma unidade na mesma cidade**, mas que não se encontram no mesmo edifício, fazendo a interligação entre eles. Além disso, também atende a escolas de uma rede da mesma localidade ou, ainda, para a conexão entre órgãos públicos na mesma rede, que estejam em edifícios diferentes.

## WAN

WAN é a sigla para a Wide Area Network e se trata de uma rede maior e, portanto, **pode abranger um país e, até mesmo, um continente inteiro ou mais**.



## WLAN

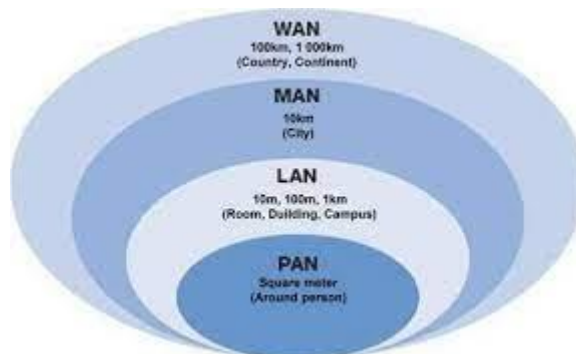
A WLAN é a sigla para Wireless-LAN, ou seja, trata-se de uma rede local na qual os dispositivos estão conectados sem a necessidade de cabos para esse fim. **É uma opção à rede LAN** e pode realizar uma divisão da conexão física em diversas conexões LAN virtuais.

## WMAN

Em paralelo, a WMAN é a sigla para Wireless-MAN, ou seja, **são redes metropolitanas que são interligadas sem a necessidade de cabos**. Utiliza torres de celulares para realizar essa conexão sem fio, unindo empresas, universidades, órgãos governamentais, entre outros.

## WWAN

Temos, em outra analogia, a WWAN, ou seja, Wireless-WAN. **São redes maiores e utilizadas, principalmente, com conexões móveis (3G, 4G e 5G),** permitindo a integração de diversos usuários no modelo Wi-Fi, ao mesmo tempo.

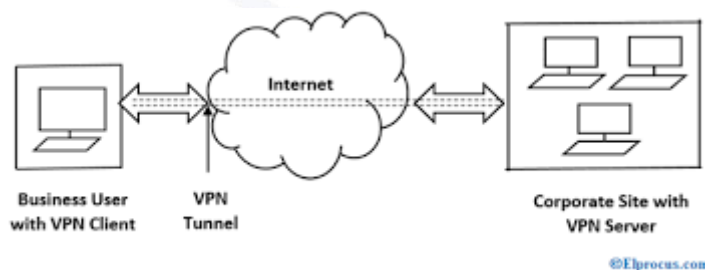


## VPN

A VPN é a sigla para Virtual Private Network, e **se trata de uma conexão entre computadores feita de forma privada.** Normalmente, é utilizada para oferecer maior privacidade e segurança de rede nas trocas de dados no dia a dia.

A VPN é bastante utilizada, por exemplo, para:

- Bloquear a navegação;
- Impedir o compartilhamento de dados internos da empresa em redes públicas;
- Realizar uma conexão criptografada;
- Esse tipo de conexão de computador é fundamental para quem deseja evitar que informações privadas possam ser obtidas por meio de cibercriminosos;



## 5.2. Protocolos de rede

Os protocolos de rede são a linguagem que permite a comunicação universal entre dispositivos diferentes. Dessa forma, é possível realizar a conexão entre as máquinas, sem



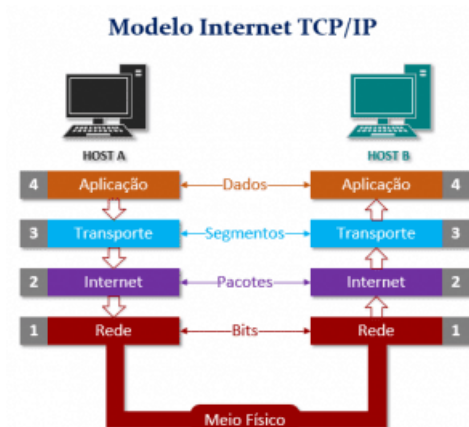
a necessidade de um software específico para realizar essa espécie de “tradução” entre eles.

Vamos estudar quais são os principais protocolos de rede:

## TCP/IP

TCP/IP é o acrônimo para Transmission Control Protocol (Protocolo de Controle de Transmissão) e Internet Protocol (Protocolo de Internet). **Eles são responsáveis pela base de envio e recebimento de dados em toda a rede, tornando a internet possível.**

Sua criação é de 1962, dentro do contexto militar da Arpanet, uma espécie de precursor da internet como conhecemos hoje. Ela foi criada para fins militares, permitindo que organizações em vários pontos do planeta pudessem trocar mensagens rapidamente, identificando o caminho mais rápido para que essa informação chegasse ao destinatário.



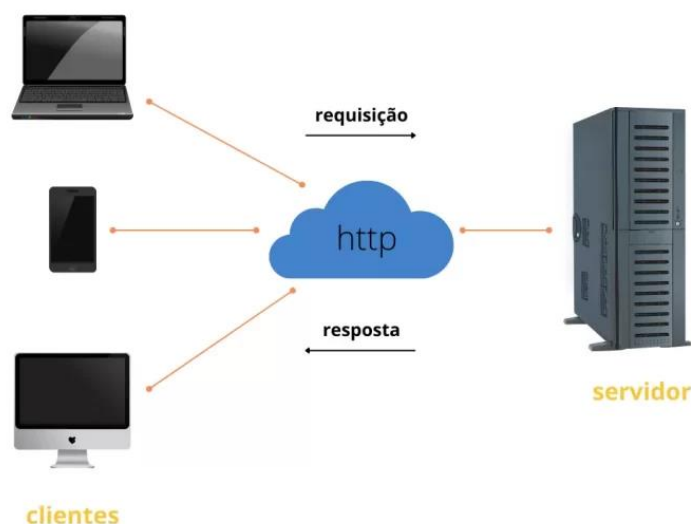
## HTTP

É o acrônimo para Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto). É um dos protocolos mais simples para a navegação em sites. Ele opera, assim, como um elo entre cliente (navegador) e servidor (aquele no qual o site ou domínio está hospedado).

Dessa forma, **o HTTP permite que as comunicações de solicitação (request) e recebimento de resposta (response) ocorram, favorecendo a navegação entre páginas por meio de links.** O mesmo protocolo possibilita, também, que ocorra a

identificação quando há um erro nessa comunicação, apontando um código: por exemplo, o Erro 500 (referente a falhas com a estrutura do site).

## Protocolo HTTP



## HTTPS

É o acrônimo para Hyper Text Transfer Secure (Protocolo de Transferência de Hipertexto Seguro). Ele age de forma semelhante ao HTTP, **contudo, oferece uma camada extra de proteção**. Assim, os sites que possuem o protocolo "S" são seguros para que o usuário os acesse.

Normalmente, é um protocolo importante para quem trabalha com operações financeiras, estabelecendo uma relação de confiança entre cliente e servidor. Por exemplo, **é uma forma de verificação de que determinado site de pagamento é seguro**.

Para que um site apareça com o protocolo HTTPS, é importante que **ele tenha um Certificado SSL, criando uma camada extra de proteção**. Ao reconhecer o certificado, o site apontará o protocolo HTTPS.

# HTTP X HTTPS

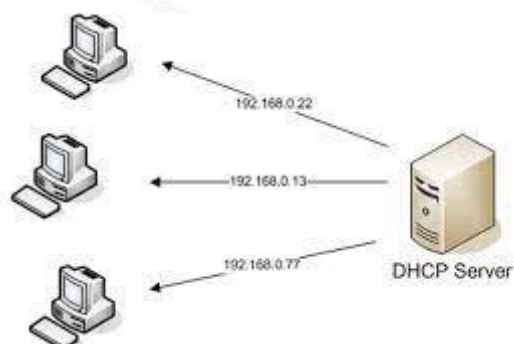
|  |  |
|--|--|
| Transfere dados em formato de hipertexto (texto estruturado) | Transfere dados em formato criptografado |
| Usa a porta 80 por padrão                                    | Usa a porta 443 por padrão               |
| Não seguro   | Protegido usando tecnologia SSL          |
| Começa com http://   | Começa com https://                      |



## DHCP

É o acrônimo para Dynamic Host Configuration Protocol (Protocolo de Configuração Dinâmica de Endereços de Rede). **Ele permite que cada dispositivo possa ter um endereço de IP automaticamente**, sem precisar realizar a configuração manual de endereços IPs para cada dispositivo.

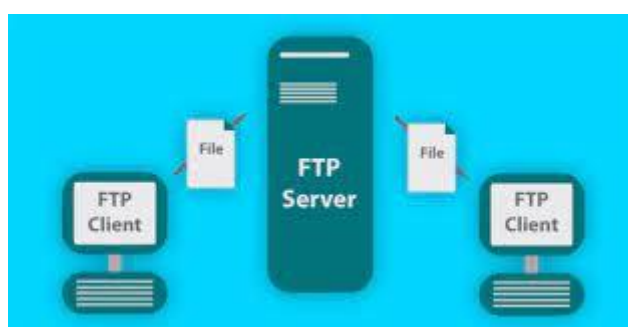
É importante lembrar que, quando o dispositivo não possui IP fixo (como é o caso nas situações de link dedicado), a identificação modifica toda vez que a máquina é desligada.



## FTP

É o acrônimo para File Transfer Protocol (Protocolo de Transferência de Arquivos). É um protocolo ainda mais antigo que o TCP/IP. Assim, é mais simples do que seu sucessor para **gerar a transmissão de dados entre duas máquinas em quaisquer tipos de rede**. Funciona por meio da conexão cliente/servidor.

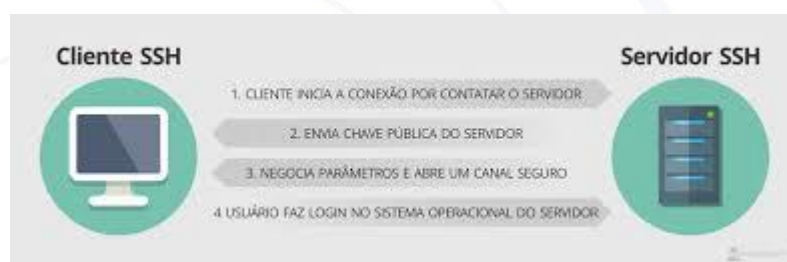
Mas afinal, se temos um sistema mais moderno, por que o FTP ainda é utilizado? Ele é útil, por exemplo, quando há perda do acesso ao painel de controle de um site. Com isso, **você pode utilizar uma ferramenta de FTP para conseguir realizar os ajustes necessários**.



## SSH

Esse protocolo trata especificamente de segurança. É a sigla para Secure Shell (Bloqueio de Segurança). Garante, assim, maior proteção nas trocas de arquivos entre cliente e servidor.

**Ele utiliza uma chave pública que permite a verificação de autenticidade do servidor**, por meio de login e senha, tornando a conexão entre computadores mais protegida. Isso permite uma maior proteção sem comprometer o desempenho da conexão. Com isso, é possível garantir a transferência com estabilidade e eficiência.



## 6. INTERCÂMBIO DE DADOS - PARTE 02

### Objetivo

O objetivo deste capítulo é apresentar o conceito e as características das APIs.

### Introdução

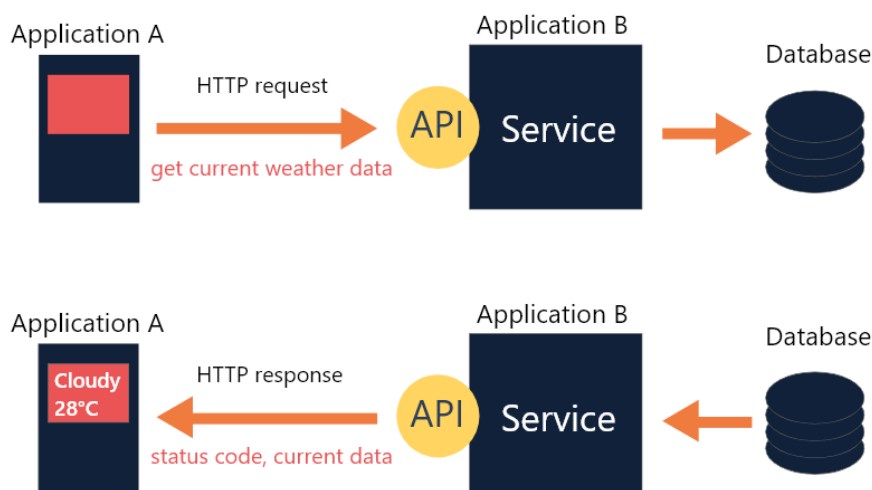
APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos. Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. O aplicativo meteorológico em seu telefone “fala” com este sistema por meio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

#### 6.1. 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.

#### 6.2. 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. Então, no exemplo do clima, o banco de dados meteorológico do instituto é o servidor e o aplicativo móvel é o cliente.



### 6.3. Tipos de dado mais usados nas APIs

A troca de informações entre APIs ocorre com a utilização de dados nos seguintes formatos:

#### XML

Extensible Markup Language, ou simplesmente XML, é uma linguagem de marcação, ou seja, um conjunto de códigos para determinar a estrutura de dados para facilitar a troca de informação entre sistemas computacionais, lançado na década de 90 pela W3C (World Wide Web Consortium - órgão responsável pela definição da linguagem XML e pela padronização de outras iniciativas ligadas à Web).

Além de ser facilmente lido sem o auxílio de qualquer software e ser responsável por prover uma língua “universal” para troca de informação entre aplicações, o XML possui como vantagens a fácil distribuição na Web, integração de dados de fontes diferentes, buscas mais eficientes, desenvolvimento de aplicações web flexíveis, entre outros.

```
<?xml version="1.0"?>
- <birds>
  - <owl id="1201">
    <species>Bubo bubo</species>
    <name>Eagle Owl</name>
    <region>Eurasia</region>
  </owl>
  - <owl id="1202">
    <species>Strix occidentalis</species>
    <name>Spotted Owl</name>
    <region>North America</region>
  </owl>
</birds>
```



## JSON

JSON é um acrônimo de “Javascript Object Notation” ou simplesmente “Notação de objeto JavaScript”. É um modelo para a transmissão de informações no formato de texto entre diferentes linguagens, ou seja, é um formato de serialização de dados muito utilizado em APIs.

Dentre suas diversas características, a mais atrativa sem dúvidas é a sua legibilidade, podendo facilmente ser lido por humanos, sem a necessidade de uma aplicação auxiliar.

```
{
  "First_Name": "eric",
  "Last_Name": "stuart",
  "ID_Number": 113547,
  "Age": 48,
  "Start_Date": "11/01/2000",
  "Job_Title": "programmer"
},
{
  "First_Name": "dan",
  "Last_Name": "baker",
  "ID_Number": 567821,
  "Age": 40,
  "Start_Date": "04/01/2004",
  "Job_Title": "programmer"
},
{
  "First_Name": "ella",
  "Last_Name": "walsh",
  "ID_Number": 983753,
  "Age": 26,
  "Start_Date": "05/01/2010",
  "Job_Title": "sales"
}
]
```

### 6.4. Tipo de APIs mais utilizadas

#### APIs SOAP

Essas APIs usam o Simple Object Access Protocol (Protocolo de Acesso a Objetos Simples). Cliente e servidor trocam mensagens usando o formato XML. Esta é uma API menos flexível que era mais popular no passado.

#### APIs REST

REST significa Transferência Representacional de Estado. 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. Vejamos as APIs REST em mais detalhes abaixo. Cliente e servidor trocam mensagens usando o formato JSON.

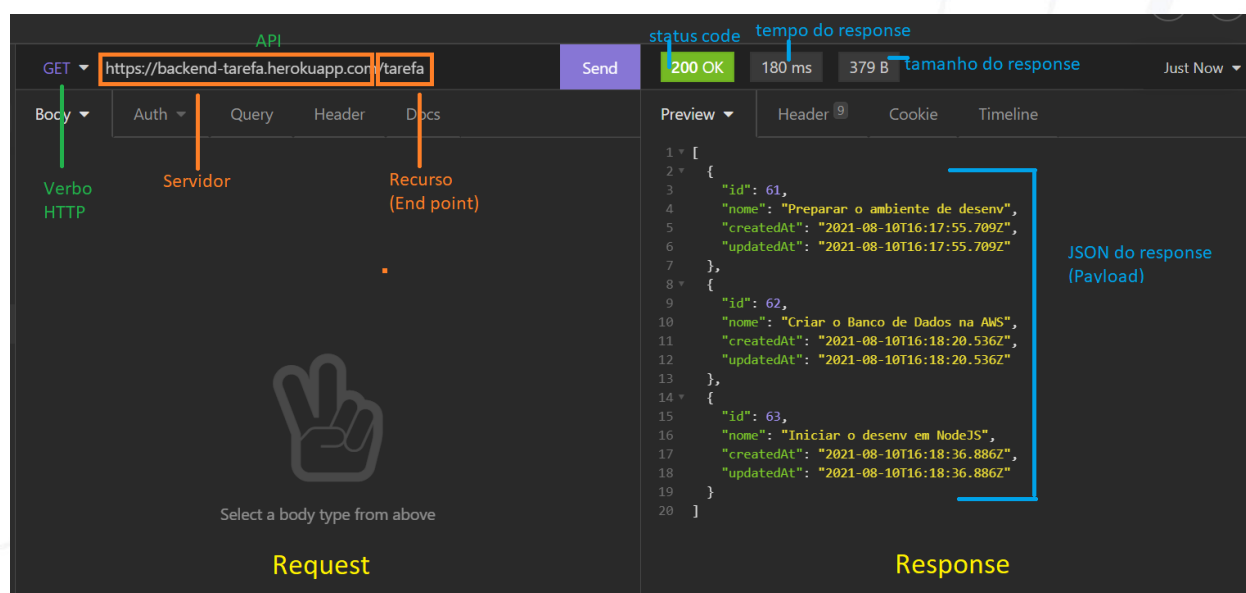
A principal característica da API REST é a ausência de estado. A ausência de estado significa que os servidores não salvam dados do cliente entre as solicitações. As

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.

## 6.5. Requisições e comunicações em APIs REST

O REST precisa que um cliente faça uma requisição para o servidor para enviar ou modificar dados. Uma requisição consiste em:

- Um verbo ou método HTTP, que define que tipo de operação o servidor vai realizar;
- Um header, com o cabeçalho da requisição que passa informações sobre a requisição;
- Um path (caminho ou rota) para o servidor, ou seja, uma URL;
- Informação no corpo da requisição, sendo esta informação opcional.

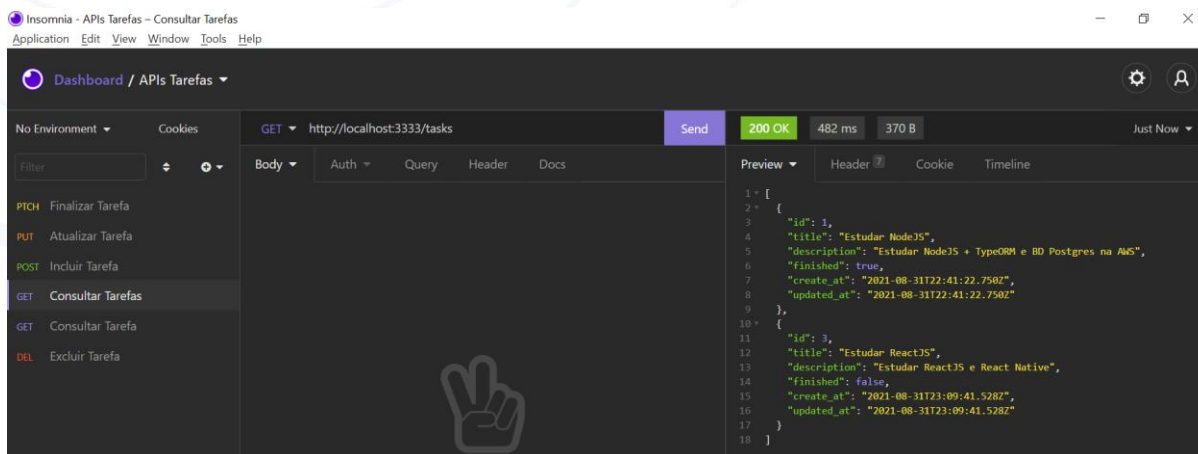


## 6.6. Verbos/Métodos HTTP

Em APIs REST, os métodos/verbos mais utilizados são:

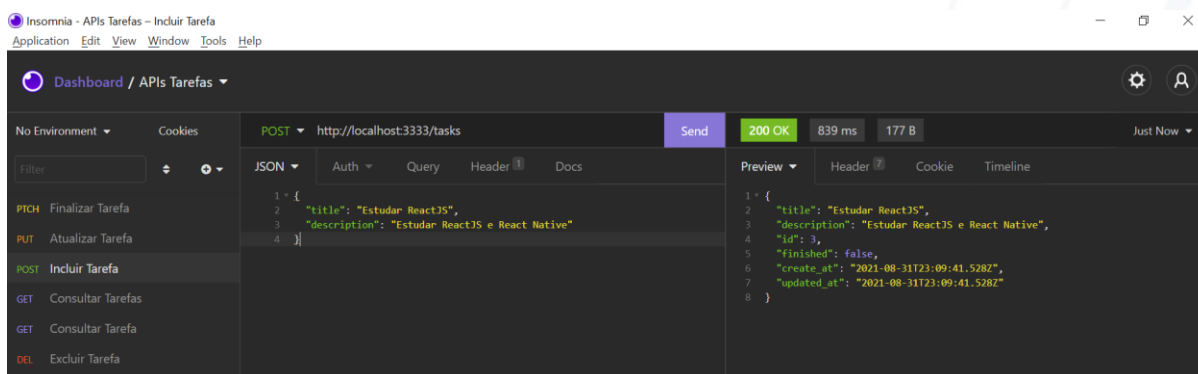
### GET

O método GET é o método mais comum, geralmente é usado para solicitar que um servidor envie um recurso



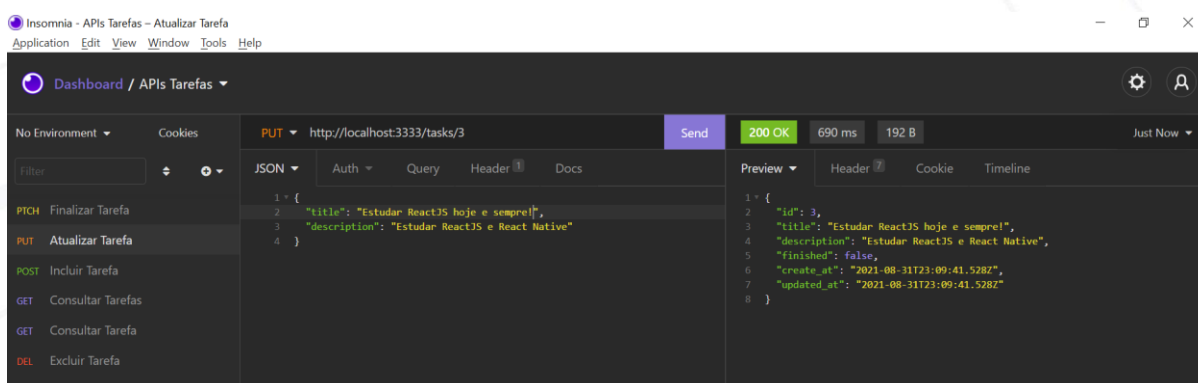
## POST

O método POST foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para suportar formulários HTML



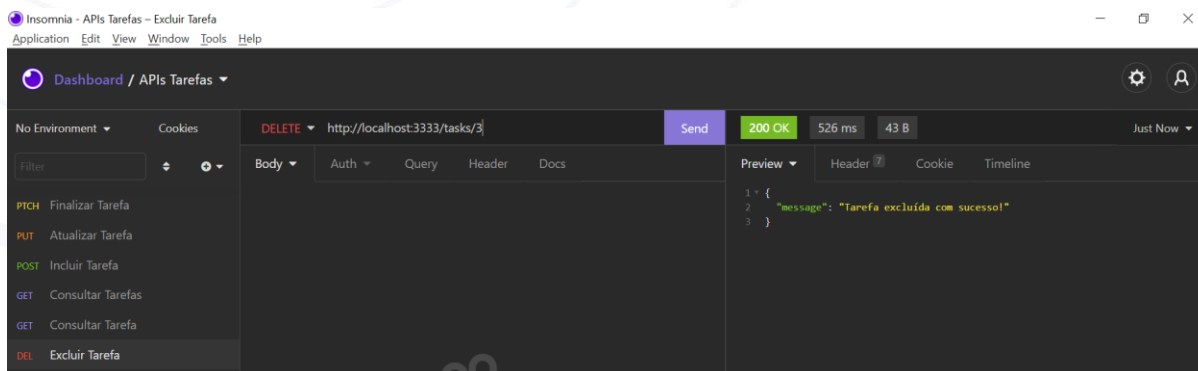
## PUT

O método PUT edita e atualiza documentos em um servidor



## DELETE

Método DELETE que como o próprio nome já diz, deleta certo dado ou coleção do servidor



## 6.7. Códigos de Respostas

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição. São eles:

- 200 (OK), requisição atendida com sucesso;
- 201 (CREATED), objeto ou recurso criado com sucesso;
- 204 (NO CONTENT), objeto ou recurso deletado com sucesso;
- 400 (BAD REQUEST), ocorreu algum erro na requisição (podem existir inúmeras causas);
- 404 (NOT FOUND), rota ou coleção não encontrada;
- 500 (INTERNAL SERVER ERROR), ocorreu algum erro no servidor.

## 6.8. Exemplo de APIs Públicas

### ViaCep

<https://viacep.com.br/>



Procurando um [webservice](#) gratuito e de alto desempenho para consultar Códigos de Endereçamento Postal (CEP) do Brasil? Utilize nosso serviço, melhore a qualidade de suas aplicações web e colabore para manter esta base de dados atualizada.

#### Acessando o webservice de CEP

Para acessar o webservice, um CEP no formato de {8} dígitos deve ser fornecido, por exemplo: "01001000". Após o CEP, deve ser fornecido o tipo de retorno desejado, que deve ser "json", "xml", "piped" ou "query".

Exemplo de pesquisa por CEP:

[viacep.com.br/ws/01001000/json/](https://viacep.com.br/ws/01001000/json/)

Exemplo de API SOAP: <https://viacep.com.br/ws/01001000/xml/>

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlcep>
  <cep>01001-000</cep>
  <logradouro>Praça da Sé</logradouro>
  <complemento>lado ímpar</complemento>
  <bairro>Sé</bairro>
  <localidade>São Paulo</localidade>
  <uf>SP</uf>
  <ibge>3550308</ibge>
  <gia>1004</gia>
  <ddd>11</ddd>
  <siafi>7107</siafi>
</xmlcep>
```

**Exemplo de API REST:** <https://viacep.com.br/ws/01001000/json/>

```
{
  "cep": "01001-000",
  "logradouro": "Praça da Sé",
  "complemento": "lado ímpar",
  "bairro": "Sé",
  "localidade": "São Paulo",
  "uf": "SP",
  "ibge": "3550308",
  "gia": "1004",
  "ddd": "11",
  "siafi": "7107"
}
```

## GitHub

<https://api.github.com/users/<<seu login>>>

```
{
  "login": "Joseffe10",
  "id": 53655211,
  "node_id": "MDQ6VXN1c2UzNjU1MjEx",
  "avatar_url": "https://avatars.githubusercontent.com/u/53655211?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/Joseffe10",
  "html_url": "https://github.com/Joseffe10",
  "followers_url": "https://api.github.com/users/Joseffe10/followers",
  "following_url": "https://api.github.com/users/Joseffe10/following{/other_user}",
  "gists_url": "https://api.github.com/users/Joseffe10/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/Joseffe10/starred{/owner}/{/repo}",
  "subscriptions_url": "https://api.github.com/users/Joseffe10/subscriptions",
  "organizations_url": "https://api.github.com/users/Joseffe10/orgs",
  "repos_url": "https://api.github.com/users/Joseffe10/repos",
  "events_url": "https://api.github.com/users/Joseffe10/events{/privacy}",
  "received_events_url": "https://api.github.com/users/Joseffe10/received_events",
  "type": "User",
  "site_admin": false,
  "name": "Joseffe Oliveira",
  "company": null,
  "blog": "https://www.linkedin.com/in/joseffe-oliveira-1aaa383a/",
  "location": null,
  "email": null,
  "hireable": null,
  "bio": null,
  "twitter_username": null,
  "public_repos": 36,
  "public_gists": 0,
  "followers": 125,
  "following": 0,
  "created_at": "2019-08-03T14:13:13Z",
  "updated_at": "2022-08-23T23:19:53Z"
}
```

## Repositório de APIs Públicas

<https://github.com/public-apis/public-apis>

# Public APIs

*A collective list of free APIs for use in software and web development*

Status

Number of Categories **51** Number of APIs **1425**

Tests of push & pull **failing** Validate links **failing** Tests of validate package **passing**

The Project

[Contributing Guide](#) • [API for this project](#) • [Issues](#) • [Pull Requests](#) • [License](#)

Alternative sites for the project (unofficials)

[Free APIs](#) • [Dev Resources](#) • [Public APIs Site](#) • [Apihouse](#) • [Collective APIs](#)

## APIs do Star Wars

<https://swapi.dev/>



**SWAPI**  
The Star Wars API  
(what happened to swapi.co?)

All the Star Wars data you've ever wanted:  
**Planets, Spaceships, Vehicles, People, Films and Species**  
From all **SEVEN** Star Wars films  
Now with The Force Awakens data!



## 7. INFRAESTRUTURA PARA SISTEMAS DISTRIBUÍDOS - PARTE 01

### Objetivo

O objetivo deste capítulo é apresentar alguns modelos de infraestrutura e suas características para a criação de sistemas distribuídos.

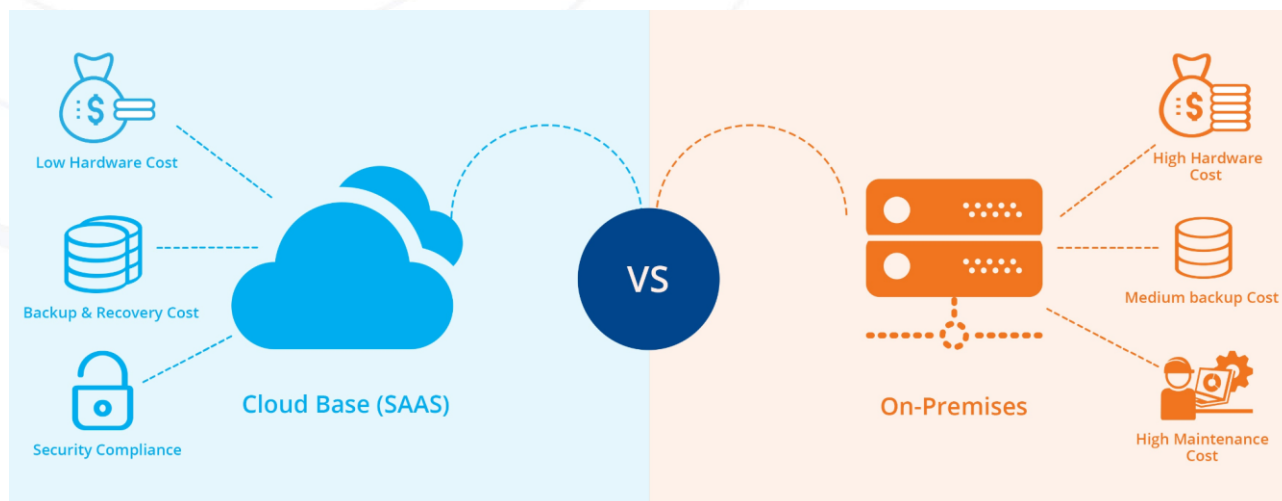
### Introdução

Com tantas diferenças entre as organizações e as novas tecnologias trazidas pela transformação digital, definitivamente não existe uma solução única para todos. A chave é, em vez disso, procurar uma solução que **ajude sua empresa a economizar custos e aumentar a eficiência**. A ampla adoção da cloud computing levou muitos fornecedores a mudar o foco dos modelos de entrega local para a nuvem, com a tecnologia cloud revolucionando os recursos de computação on-premise de trabalho. No entanto, para empresas que ainda não mudaram suas aplicações ou armazenamento de dados para a nuvem, a pergunta “Qual é a melhor solução para o meu negócio?” Pode ter vindo à mente uma ou duas vezes. Dito isso, existem empresas que ainda preferem manter uma infraestrutura de armazenamento de dados on-premise em vez de utilizar a cloud computing. Ambas as abordagens oferecem suas próprias vantagens, mas pode não ser fácil distinguir qual seria a melhor para certos tipos de organizações sem a devida consideração.

### 7.1. Infraestrutura On-premise vs Infraestrutura Cloud computing

Não é nenhuma surpresa que a cloud computing tenha crescido em popularidade, principalmente pela oferta de flexibilidade, economia de tempo e dinheiro até o aumento de agilidade e escalabilidade.

Por outro lado, o modelo on-premise - onde os servidores ficam localizados na própria empresa - foi a única opção para as organizações por um longo tempo. E até pode continuar a atender adequadamente às suas necessidades de negócios. Além disso, o armazenamento local é confiável, seguro e permite que as empresas mantenham um nível de controle que a nuvem geralmente não possibilita.



### On-premise

Quer a empresa coloque suas aplicações na nuvem ou decida mantê-los on-premise, a segurança dos dados sempre será primordial. Mas, para as empresas em setores altamente regulamentados, geralmente a decisão é manter tudo ‘dentro de casa’. Saber que seus dados estão localizados em seus servidores internos e infraestrutura de TI também pode fornecer mais tranquilidade de qualquer maneira.

**A desvantagem do on-premise é que os custos associados ao gerenciamento e manutenção** de toda a infraestrutura podem ser exponencialmente mais altos do que um ambiente de cloud computing.

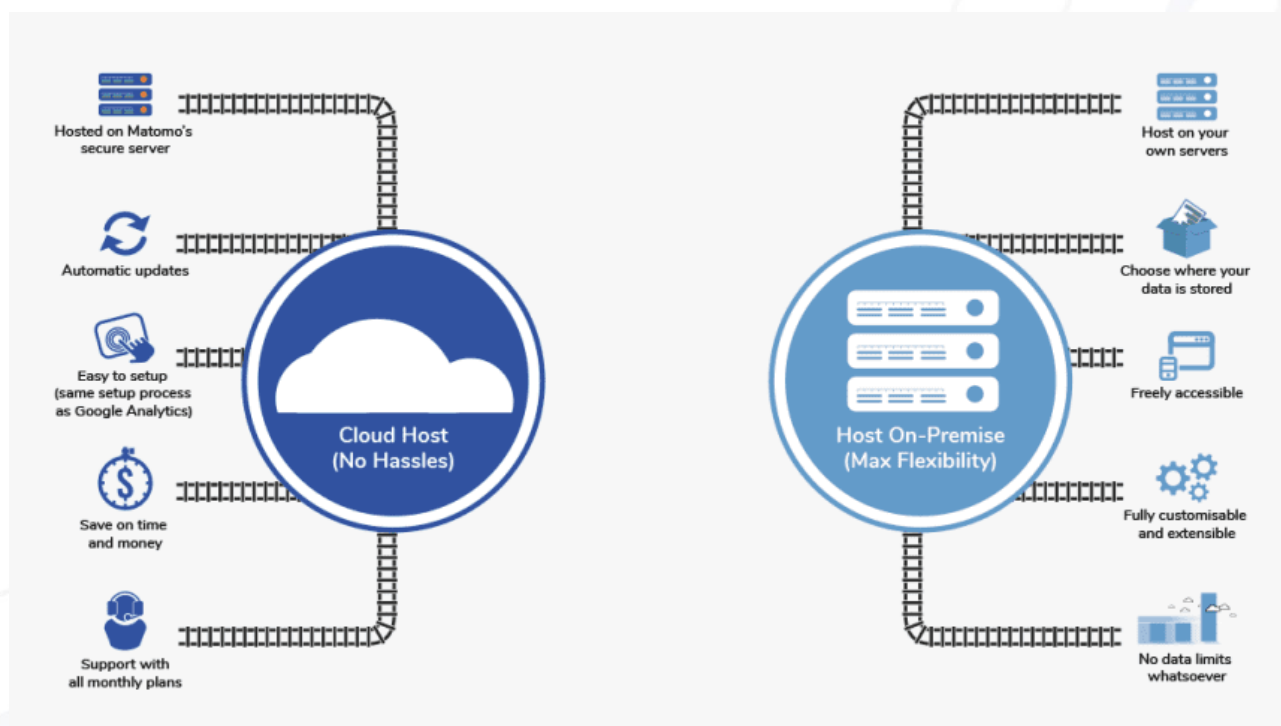
Uma configuração local requer **hardware de servidor interno, licenças de software, recursos de integração e colaboradores de TI disponíveis** para oferecer suporte e gerenciar possíveis problemas que possam surgir. Isso nem mesmo leva em consideração a quantidade de manutenção pela qual uma empresa é responsável quando algo quebra ou não funciona.

### Cloud computing

A cloud computing difere do modelo on-premise de uma maneira crítica. Uma empresa hospeda tudo internamente em um ambiente local, enquanto em um ambiente de nuvem um provedor terceirizado hospeda tudo isso para você. **Isso permite que as empresas paguem conforme a necessidade e aumentem ou diminuam o armazenamento contratado com eficácia, dependendo do uso, dos requisitos do usuário e do crescimento da empresa.**

Um servidor baseado em nuvem utiliza tecnologia virtual para hospedar as aplicações da empresa externamente. **Não há despesas com manutenção de hardware, é possível fazer backup dos dados regularmente e as empresas só precisam pagar pelos recursos que usam.** Para as organizações que planejam uma expansão agressiva, a nuvem tem um apelo ainda maior porque permite que você se conecte com clientes, parceiros e outras empresas em qualquer lugar com o mínimo de esforço.

Além disso, **a cloud computing oferece provisionamento quase instantâneo porque tudo já está configurado.** Assim, qualquer novo software integrado ao seu ambiente está pronto para ser usado imediatamente após a assinatura. Com o provisionamento instantâneo, qualquer tempo gasto na instalação e configuração é eliminado e os usuários podem acessar o aplicativo imediatamente.



## 7.2. Principais diferenças dos modelos on-premise e cloud computing

Há uma série de diferenças fundamentais entre um ambiente local e um ambiente de nuvem. Qual caminho é o correto para sua empresa depende inteiramente de suas necessidades e do que você procura em uma solução.

## Desenvolvimento

**On-premise:** Em um ambiente local, os recursos são implantados internamente e na infraestrutura de TI de uma empresa. A empresa é responsável por manter a solução e todos os seus processos relacionados.

**Cloud computing:** Embora existam diferentes formas de cloud computing (como nuvem pública, privada ou híbrida), em um ambiente de cloud computing os recursos são hospedados nas instalações do provedor de serviços, mas as empresas podem acessar esses recursos e usar o quanto quiserem em um determinado momento.

## Custo

**On-premise:** As empresas são responsáveis pelos custos contínuos de hardware de servidor, consumo de energia e espaço.

**Cloud computing:** As empresas que optam por usar um modelo de cloud computing só precisam pagar pelos recursos que usam, sem nenhum dos custos de manutenção e conservação, e o preço se ajusta para cima ou para baixo dependendo de quanto é consumido.

## Controle

**On-premise:** Em um ambiente local, as empresas retêm todos os seus dados e têm controle total sobre o que acontece com eles, para o bem ou para o mal. As empresas em setores altamente regulamentados com preocupações extras com a privacidade são mais propensas a hesitar em saltar para a nuvem antes de outras por esse motivo.

**Cloud computing:** Em um ambiente de cloud computing, a questão da propriedade dos dados é uma questão com a qual muitas empresas - e fornecedores, nesse caso, têm dificuldade. As chaves de dados e criptografia residem em seu provedor terceirizado, portanto, se o inesperado acontecer e houver um tempo de inatividade, talvez você não consiga acessar esses dados.

## Segurança

**On-premise:** As empresas que possuem informações confidenciais extras, como os setores governamental e bancário, devem ter um determinado nível de segurança e privacidade fornecido por um ambiente local.

**Cloud computing:** As preocupações com a segurança continuam sendo a barreira número um para a implantação da cloud computing. Muitas violações na nuvem foram divulgadas e os departamentos de TI em todo o mundo estão preocupados. De informações pessoais de colaboradores, como credenciais de login, a perda de propriedade intelectual, as ameaças à segurança são reais.

### **Compliance**

**On-premise:** Muitas empresas hoje em dia operam sob alguma forma de controle regulatório, como a LGPD, independentemente do setor. Para empresas que estão sujeitas a essas regulamentações, é fundamental que permaneçam em conformidade e saibam onde estão seus dados o tempo todo.

**Cloud computing:** As empresas que escolherem um modelo de cloud computing devem garantir que seu provedor terceirizado esteja preparado e, de fato, em conformidade com todos os diferentes mandatos regulatórios de seu setor. Os dados confidenciais devem ser protegidos e clientes, parceiros e funcionários devem ter sua privacidade garantida.

## 8. INFRAESTRUTURA PARA SISTEMAS DISTRIBUÍDOS - PARTE 02

### Objetivo

O objetivo deste capítulo é apresentar as 3 principais Cloud mais utilizadas no mundo, onde grande parte dos sistemas distribuídos estão sendo hospedados.

### Introdução

Mesmo com o conceito tendo surgido na década de 60, a primeira vez que “computação em nuvem” foi usado foi em 1997. Hoje, é uma realidade sólida e faz cada vez mais parte da rotina de trabalho e armazenamento de arquivos pessoais. Os três maiores nomes do mercado, a Google Cloud, a Windows Azure e a AWS têm investido em plataformas cada vez melhor desenvolvidas para oferecer a seus usuários maior agilidade e menor custo. A migração de empresas para o cenário Cloud cresce vertiginosamente e compele os provedores de tecnologia nuvem a oferecerem vantagens como escalabilidade, segurança, economia de custos e outros fatores primordiais, o que acelera a competição entre gigantes.

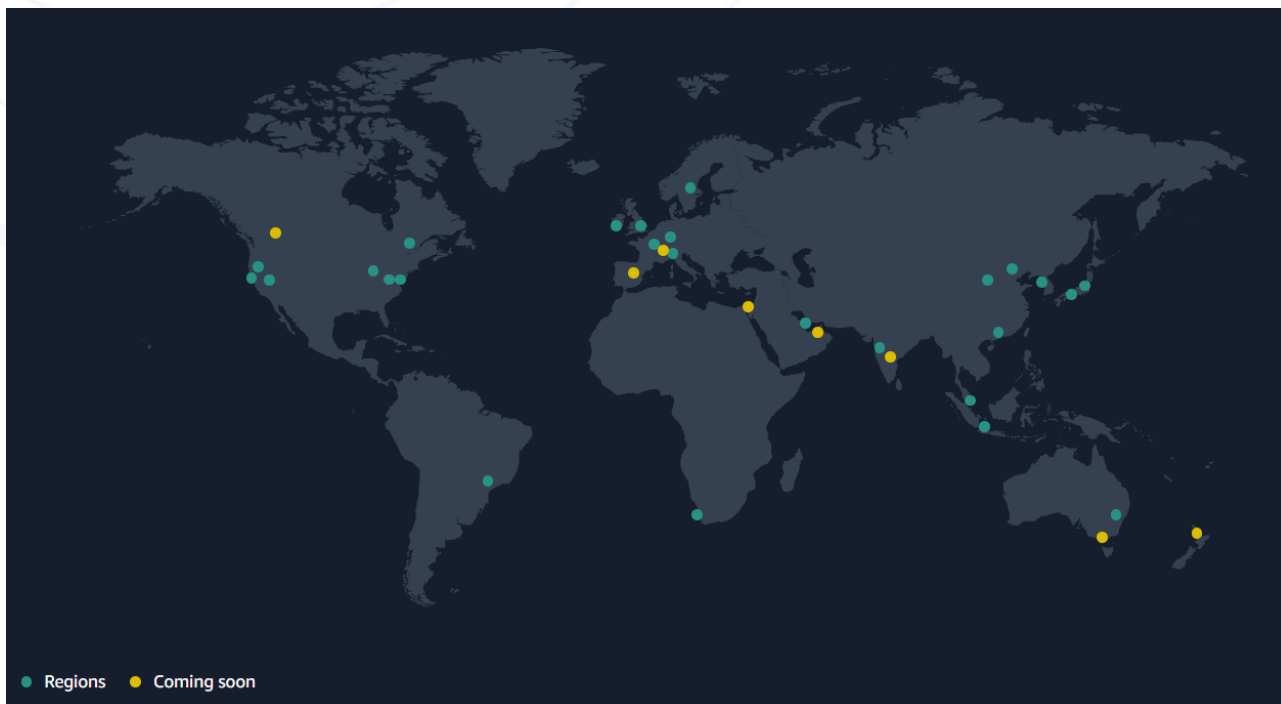
### 8.1. Regiões e Zonas de disponibilidades

#### Regiões

Uma região é um local físico em todo o mundo onde se é agrupado data centers. Chamamos a cada grupo de datacenters lógicos de zona de disponibilidade. Cada região consiste em várias Zonas de Disponibilidades (AZs) isoladas e separadas fisicamente em uma área geográfica.

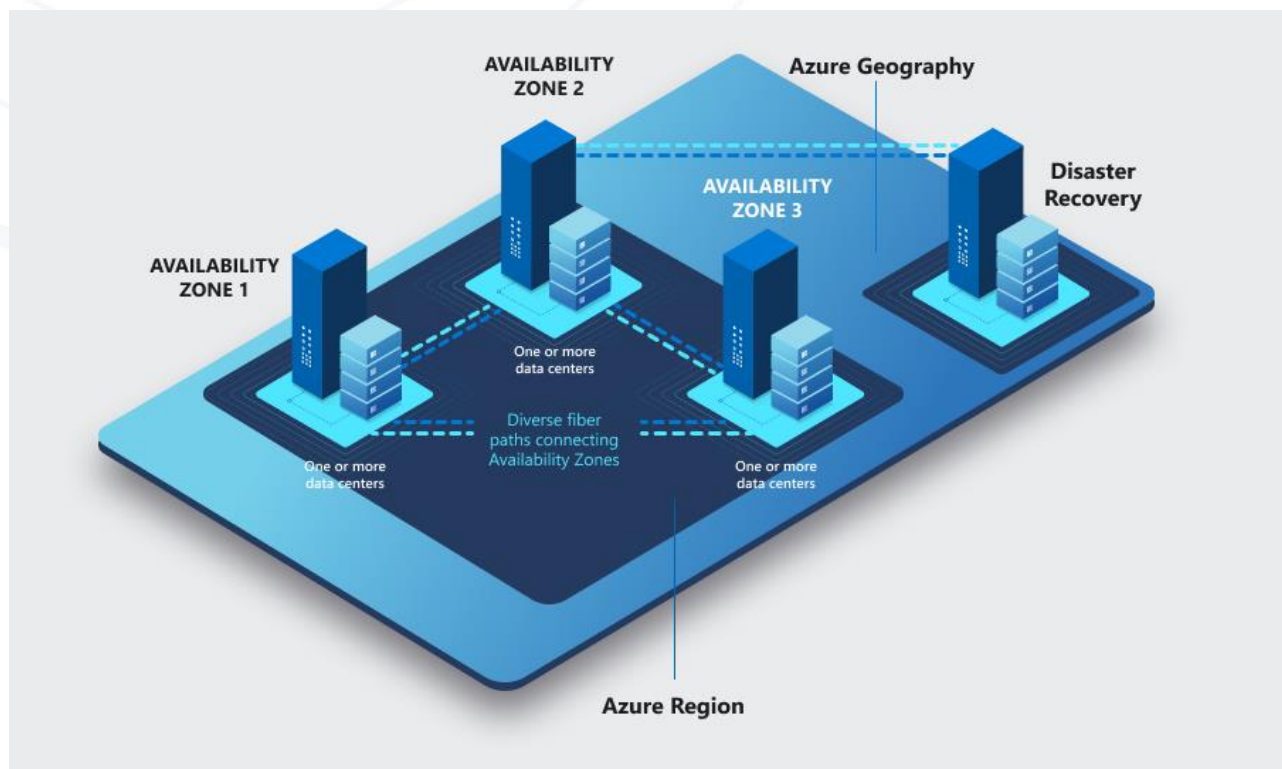
Por exemplo, a AWS fornece uma presença global mais extensa do que qualquer outro provedor de nuvem. Para oferecer suporte à sua presença global e garantir que os clientes sejam atendidos em todo o mundo, a AWS abre novas regiões rapidamente. A AWS mantém várias regiões geográficas, incluindo regiões da América do Norte, África do Sul, América do Sul, Europa, China, Ásia-Pacífico, África do Sul e Oriente Médio.





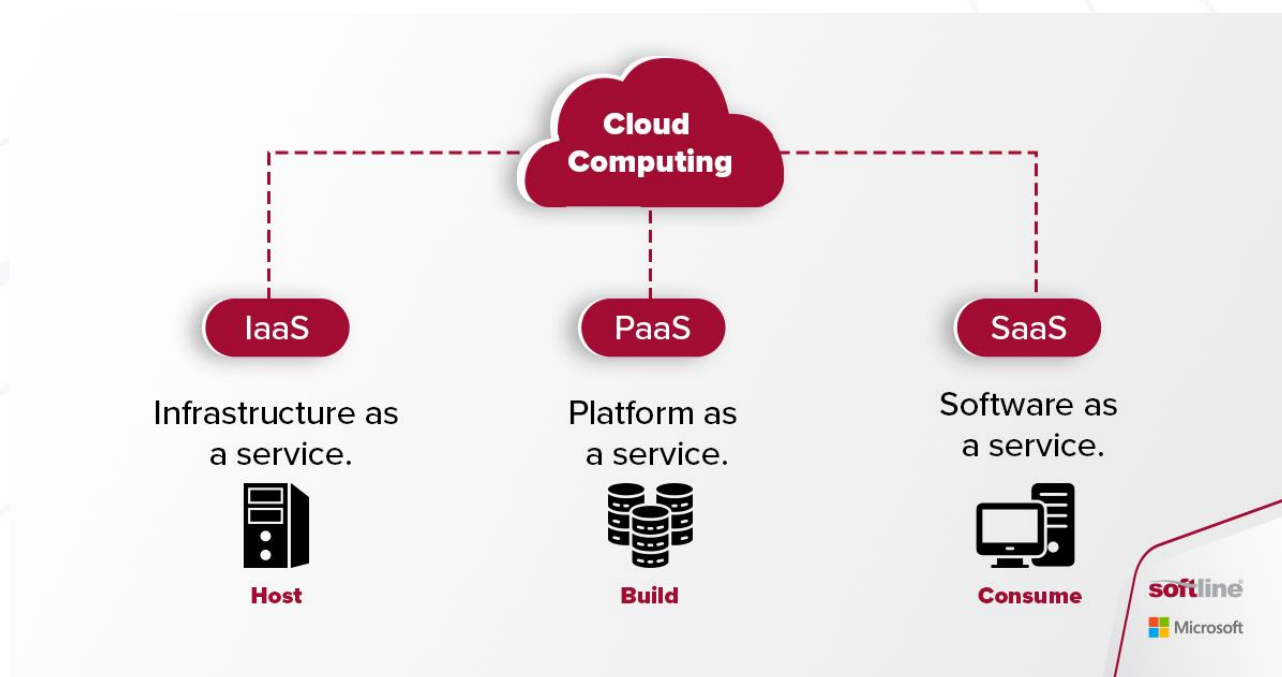
### Zonas de disponibilidade

Uma zona de disponibilidade (AZ) é um ou mais datacenters distintos com energia, rede e conectividade redundantes em uma região. As AZs proporcionam aos clientes a capacidade de operar aplicativos e bancos de dados de produção com alta disponibilidade, tolerância a falhas e escalabilidade em níveis superiores aos que um único datacenter pode oferecer. Todas as AZs em uma região estão interconectadas por redes de alta largura de banda e baixa latência, usando fibra metropolitana dedicada e totalmente redundante para proporcionar redes de alto throughput e baixa latência entre AZs. Todo o tráfego entre as AZs é criptografado. O desempenho da rede é suficiente para realizar a replicação síncrona entre as AZs. As AZs particionam aplicativos para facilitar a alta disponibilidade. Se um aplicativo for particionado em várias AZs, as empresas estarão melhor isoladas e protegidas contra problemas como quedas de energia, raios, tornados e terremotos, entre outros. As AZs são fisicamente separadas por uma distância significativa (vários quilômetros) das outras AZs, embora todas estejam em um raio de até 100 km entre si.



## 8.2. Modelos de serviço Cloud

É totalmente possível utilizar Cloud Computing para desenvolvimento e hospedagem de sistemas distribuídos, entretanto, podemos classificar o modo de uso de Cloud em alguns modelos, são eles: IaaS, PaaS e SaaS.



**IaaS: Infrastructure as a Service (Infraestrutura como Serviço)**

No modelo IaaS, a empresa contrata uma capacidade de hardware que corresponde à memória, armazenamento e processamento, por exemplo. Podem entrar nesse pacote de contratações os servidores, roteadores, racks, entre outros.

É importante dizer que, a depender do fornecedor escolhido, é possível pagar uma tarifa pelo número de servidores utilizados e pela quantidade de dados armazenados ou trafegados.

Em geral, tudo é fornecido por meio de um Data Center com servidores virtuais. O interessante é que você paga somente por aquilo que usar.

### **PaaS: Platform as a Service (Plataforma como Serviço)**

Imagine que você contratou uma ótima solução para a sua empresa — que funciona na nuvem —, mas que não possui um recurso personalizado essencial para o seu trabalho.

Nesse cenário, o PaaS é a resposta para os seus problemas! Como o próprio nome sugere, você poderá usufruir de uma plataforma capaz de criar, hospedar e gerir aplicativos na nuvem.

Nesse modelo de nuvem, o negócio contrata um ambiente completo de desenvolvimento. Nele, é possível criar, modificar e otimizar softwares e aplicações.

Tudo isso é feito utilizando a infraestrutura na nuvem. Ou seja, o time de desenvolvimento tem um ambiente completo e moderno à sua disposição. E o melhor: sem a necessidade de fazer altos investimentos.

### **SaaS: Software as a Service (Software como Serviço)**

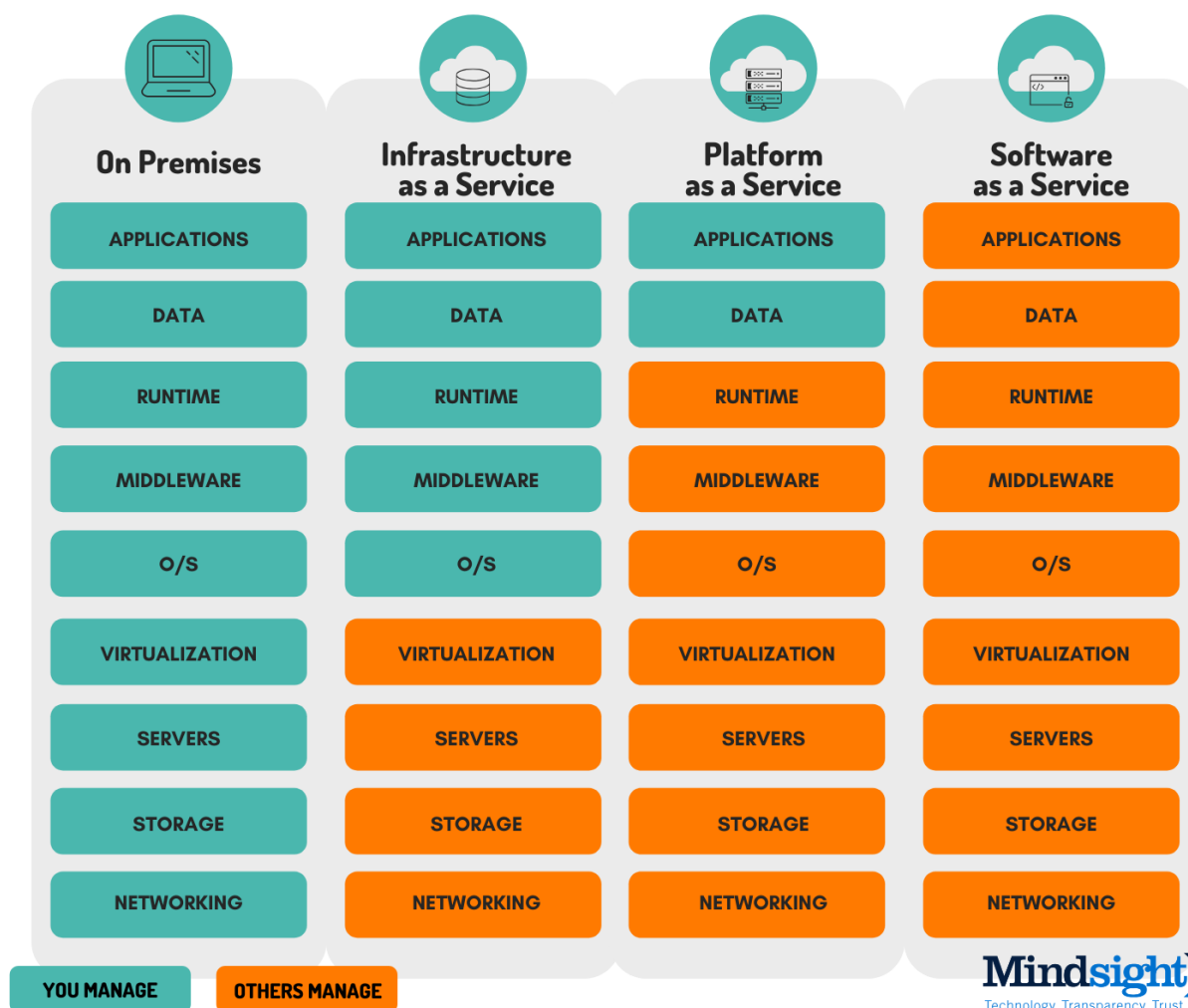
Quase todo mundo conhece o SaaS, mesmo que não saiba. Nesse modelo de nuvem, o usuário compra uma licença para ter acesso ao software, utilizando-o a partir da Cloud Computing, muitas vezes com recursos limitados.

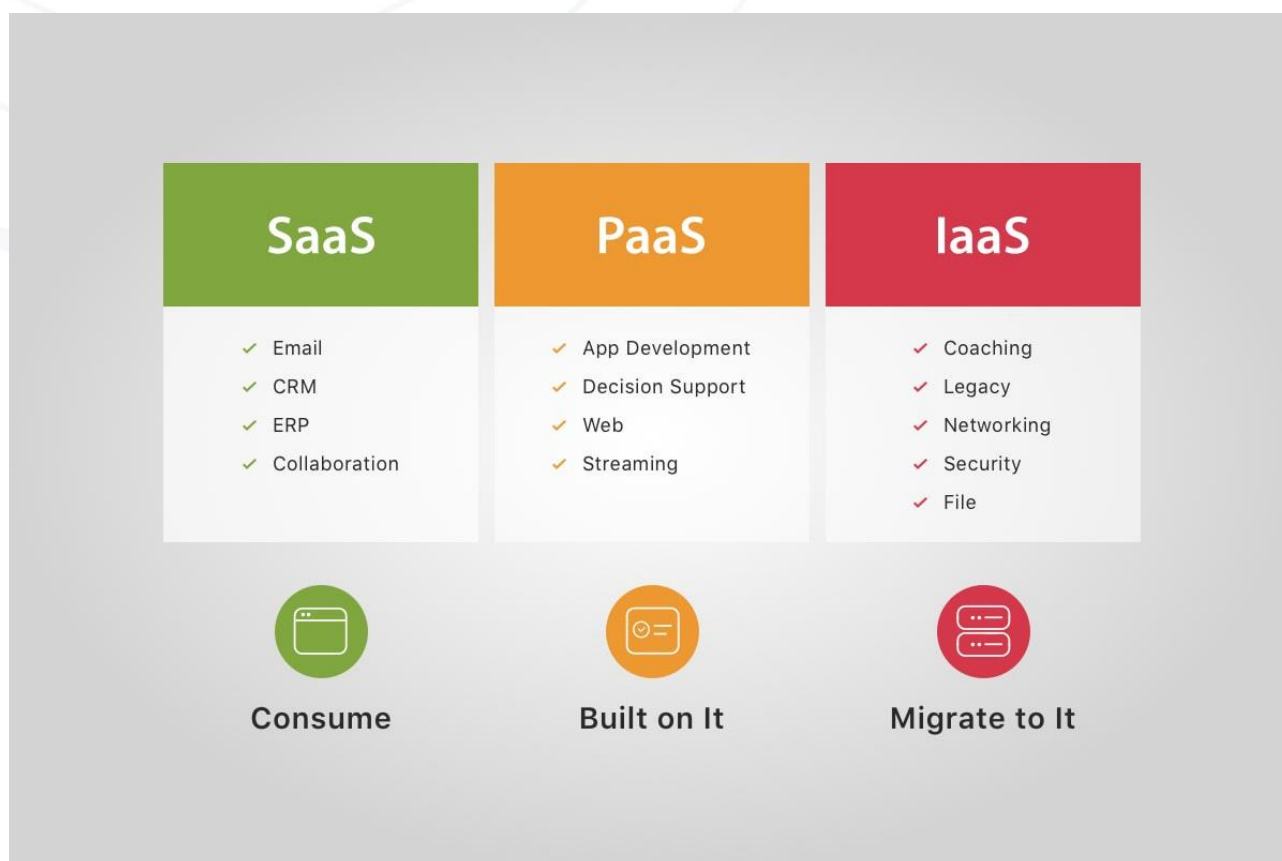
No entanto, também existem planos de pagamento nos quais é cobrada uma taxa fixa ou um valor que varia de acordo com o uso. Muitos CRMs ou ERPs trabalham no sistema SaaS.

Assim, o uso dos softwares depende da internet. Os dados, contatos e demais informações podem ser acessados de qualquer dispositivo, dando mais mobilidade à equipe.

Falamos que qualquer um conhece o SaaS porque vários sites, como o Facebook e o Twitter, e aplicativos, como o Skype, OneDrive e Office 365, funcionam dessa maneira.

Neles, tudo é disponibilizado na nuvem, para que muitos usuários consigam ter acesso ao serviço pelo browser ou por um software.

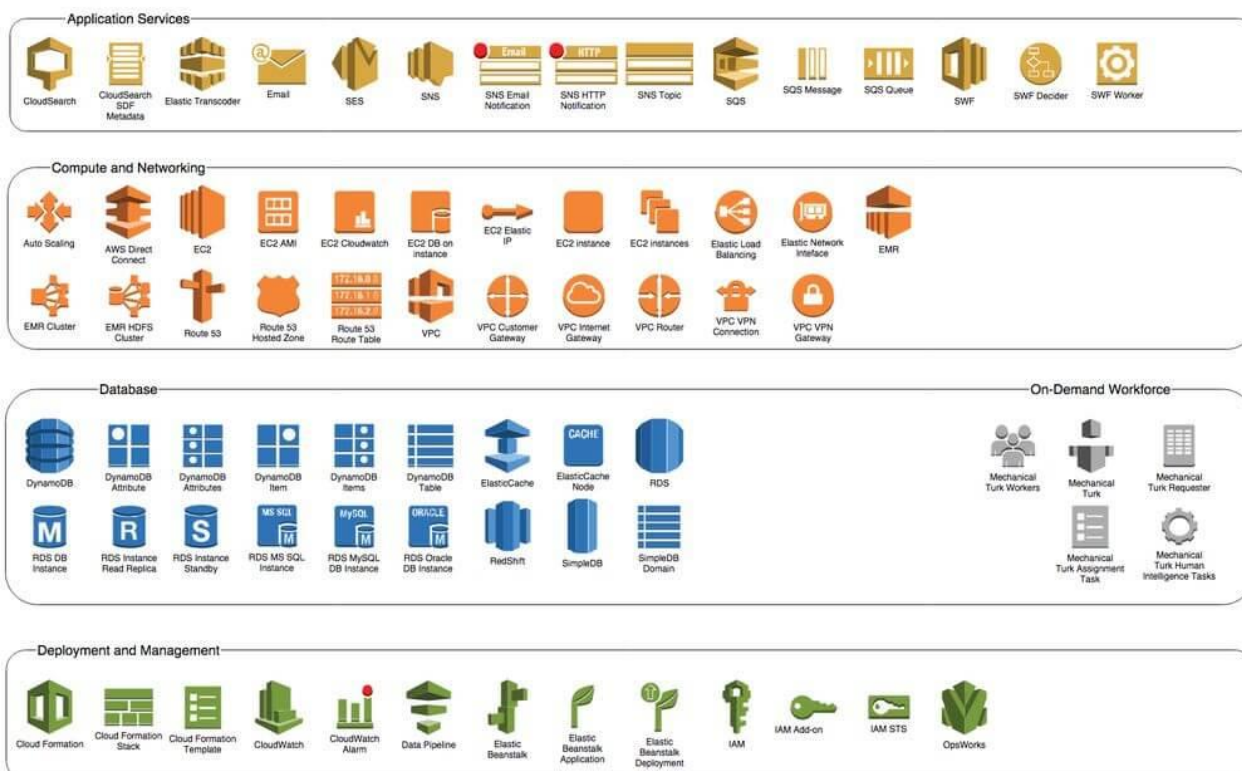




### 8.3. AWS (Amazon)

A Amazon Web Services tem seu foco voltado a ser um provedor amplo de serviços de TI, abrangendo nativos da nuvem e borda até ERP e cargas de trabalho essenciais.

A empresa possui operações geograficamente diversificadas e atende clientes de todas as demandas, desde startups em seu estágio inicial até grandes empresas consolidadas no mercado.



## Prós e Contras da AWS

A maior força da Amazon é o domínio do mercado de nuvem pública.

Em seu Quadrante Mágico de Infraestrutura de Nuvem como Serviço, em todo o mundo, o Gartner observou: “A AWS é líder em participação de mercado em IaaS na nuvem há mais de 10 anos”.

Parte da razão de sua popularidade é, sem dúvida, o enorme escopo de suas operações.

A AWS possui uma enorme e crescente variedade de serviços disponíveis, bem como a rede mais abrangente de data centers em todo o mundo.

O relatório do Gartner resumiu, dizendo: “A AWS é o provedor mais maduro e pronto para empresas, com os recursos mais profundos para administrar um grande número de usuários e recursos”.



A grande fraqueza da Amazon está relacionada ao custo. Muitas empresas acham difícil entender a estrutura de custos da empresa e gerenciar esses custos efetivamente ao executar um alto volume de cargas de trabalho no serviço.

### **Quando escolher AWS**

A AWS é uma ótima opção para cargas de trabalho analíticas e web, até migrações de data center em grande escala, a AWS fornece uma série de serviços.

Quando se trata de computação, a AWS fornece a maior variedade de tipos de VM. Atualmente, a AWS também possui as mais altas opções de computação e armazenamento disponíveis no mercado.

Sua ampla variedade de tipos de VM (136 tipos de VM e mais de 26 famílias de VM) permite que os clientes executem tudo, desde pequenas cargas de trabalho na web até as maiores cargas de trabalho.

Para aprendizado de máquina e cargas de trabalho de IA, a AWS também fornece as configurações mais altas dos tipos de VM habilitados para GPU.

Para cargas de trabalho que exigem locação única por motivos de conformidade e regulamentação, a AWS agora também fornece Bare-Metal-as-a-Service.

O armazenamento em bloco vem com uma variedade de opções, como redimensionamento dinâmico, diferentes tipos de disco (magnético e SSD).

Ao contrário de outros CSPs, a AWS não restringe IOPS por tamanho de volume. Você pode provisionar IOPS por um custo extra até para discos pequenos.

Na frente do banco de dados relacional gerenciado, a AWS oferece suporte a bancos de dados gerenciados para MySQL, PostgreSQL, MariaDB, Oracle (SE e EE) e MS SQL (edições Web e Enterprise).

Além disso, eles têm seu próprio banco de dados compatível com MySQL e PostgreSQL, que oferece desempenho semelhante ao Oracle por um investimento baixo.

Para bancos de dados NoSQL, a AWS disponibiliza seu produto DynamoDB há mais de meia década.

A AWS é um defensor e fornece uma variedade de bancos de dados NoSQL criados para esse fim. Isso inclui DynamoDB, Neptune e ElastiCache.

Para segurança de rede, a AWS lançou serviços gerenciados para proteção contra DDoS (AWS Shield) e Web Application Firewall (WAF), juntamente com o AWS Inspector, o AWS Config e o CloudTrail para gerenciamento e auditoria de inventário e políticas.

O GuardDuty fornece detecção de ameaças.

A AWS atende a cargas de trabalho do governo dos EUA em regiões separadas do GovCloud nos EUA (CIA e FBI).

#### 8.4. Azure (Microsoft)

O Microsoft Azure é forte em todos os casos de uso, que incluem a computação estendida em nuvem e de borda.

Sua capacidade excepcional garante ao cliente corporativo, principalmente, uma experiência sólida como só a Microsoft poderia – inclusive o suporte Windows.

##### Infrastructure Services

###### Compute

-  Windows
-  Linux
-  Containers

###### Storage

-  BLOB Storage
-  Azure Files
-  Premium Storage

###### Networking

-  Virtual Network
-  Load Balancer
-  DNS
-  Express Route
-  Traffic Manager
-  VPN Gateway
-  Application Gateway

##### Platform Services

###### Compute

-  Cloud Services
-  Service Fabric
-  Batch

###### Integration

-  Storage Queues
-  Biztalk Services
-  Hybrid Connections
-  Service Bus

###### Media & CDN

-  Media Services
-  Content Delivery Network (CDN)

###### App Service

-  Web Apps
-  API Apps
-  API Management
-  Mobile Apps
-  Logic Apps
-  Notification Hubs

###### Developer Services

-  Visual Studio
-  Azure SDK
-  Team Project
-  Application Insights

###### Analytics & IoT

-  HDInsight
-  Machine Learning
-  Data Factory
-  Event Hubs
-  Stream Analytics
-  Mobile Engagement

###### Data

-  SQL Database
-  Redis Cache
-  DocumentDB
-  SQL Data Warehouse
-  Search
-  Tables

##### Security & Management

-  Portal
-  Active Directory
-  Multi-Factor Authentication
-  Automation
-  Key Vault
-  Store/Marketplace
-  VM Image Gallery & VM Depot

Com foco de investimento em melhorias na plataforma, fornecendo um grande leque de serviços, suas operações são geograficamente diversificadas e seus clientes tendem a ser empresas de médio e grande porte.

A Azure é a concorrente que mais conseguiu se aproximar da líder AWS.

Em 2021, enquanto a AWS comandava 33% do nicho no mercado, a Azure conseguiu 20% de expressão, o que é uma porcentagem bastante elevada.

### **Prós e contras do Microsoft Azure**

A Microsoft chegou atrasada ao mercado de nuvem, mas deu um passo à frente, adotando essencialmente o software local – Windows Server, Office, SQL Server, Sharepoint, Dynamics Active Directory, .Net e outros – e adaptando-o novamente para a nuvem.

Um grande motivo para o sucesso do Azure é a integração com as aplicações/softwarewares da Microsoft.

Como o Azure está totalmente integrado a esses outros aplicativos, as empresas que usam muitos softwares da Microsoft geralmente acham que também faz sentido usar o Azure.

### **Quando escolher o Azure**

O Azure é uma plataforma de nuvem de grande importância no mercado com uma variedade de recursos, que pode ser uma plataforma preferida para clientes que já estão usando produtos da Microsoft.

Embora o Azure ofereça suporte a vários serviços baseados em produtos de código aberto, o portfólio da Microsoft na nuvem é o que o diferencia dos clientes.

O Azure tem mais de 151 tipos de VMs e 26 famílias que oferecem suporte a tudo, desde pequenas cargas de trabalho até as cargas de trabalho HPC, Oracle e SAP.

O Azure possui Windows e vários tipos de Linux (RHEL, CentOS, SUSE, Ubuntu). O Azure tem uma família separada de instâncias para cargas de trabalho de ML/AI.

Se você precisar executar cargas de trabalho de última geração que exijam até 128 vCPU e memória de 3,5 TB, o Azure consegue.

Se você possui licenças existentes para Windows OS, MS-SQL e as traz para a nuvem (BYOL) por meio do Microsoft License Mobility Program, o Azure é a opção.

O Azure também foi o primeiro player de nuvem a reconhecer a tendência da nuvem híbrida. O Azure também forneceu suporte para dispositivos de armazenamento híbridos como o StorSimple, que era único no espaço da nuvem pública.

Se você possui um data center com cargas de trabalho predominantemente da Microsoft e precisa fazer uma migração em grande escala para a nuvem, aproveitando as ferramentas conhecidas, o Azure fornece ferramentas e serviços, como o Azure Site Recovery.

Quando se trata de bancos de dados SQL e NoSQL, o Azure tem um conjunto de serviços bastante completo. Ele fornece o MS SQL Server e o SQL Datawarehouse Gerenciados.

O Azure também fornece bancos de dados gerenciados para MySQL, PostgreSQL e MariaDB.

Ele fornece uma API compatível com MongoDB, Cassandra, Gremlin (Graph) e Armazenamento de Tabela do Azure.

Se você precisar executar vários modelos de dados gerenciados, incluindo modelos de dados de documentos, gráficos, valores-chave, tabelas e famílias de colunas em uma única nuvem, o Cosmos pode ser a melhor opção.

O Microsoft Azure Cosmos DB é nomeado líder no relatório Forrester Wave™: NoSQL Big Data, relatório do 1º trimestre de 2019.

Além do modelo de cobrança pagamento por uso com cartão de crédito e outros modos de faturamento, os clientes com contas corporativas existentes podem comprar pré-assinaturas do Azure como parte de suas renovações anuais.

Isso é útil para clientes que desejam orçar os gastos anuais da nuvem com antecedência. Evitando a incerteza e as aprovações adicionais de orçamento para o meio do ano.

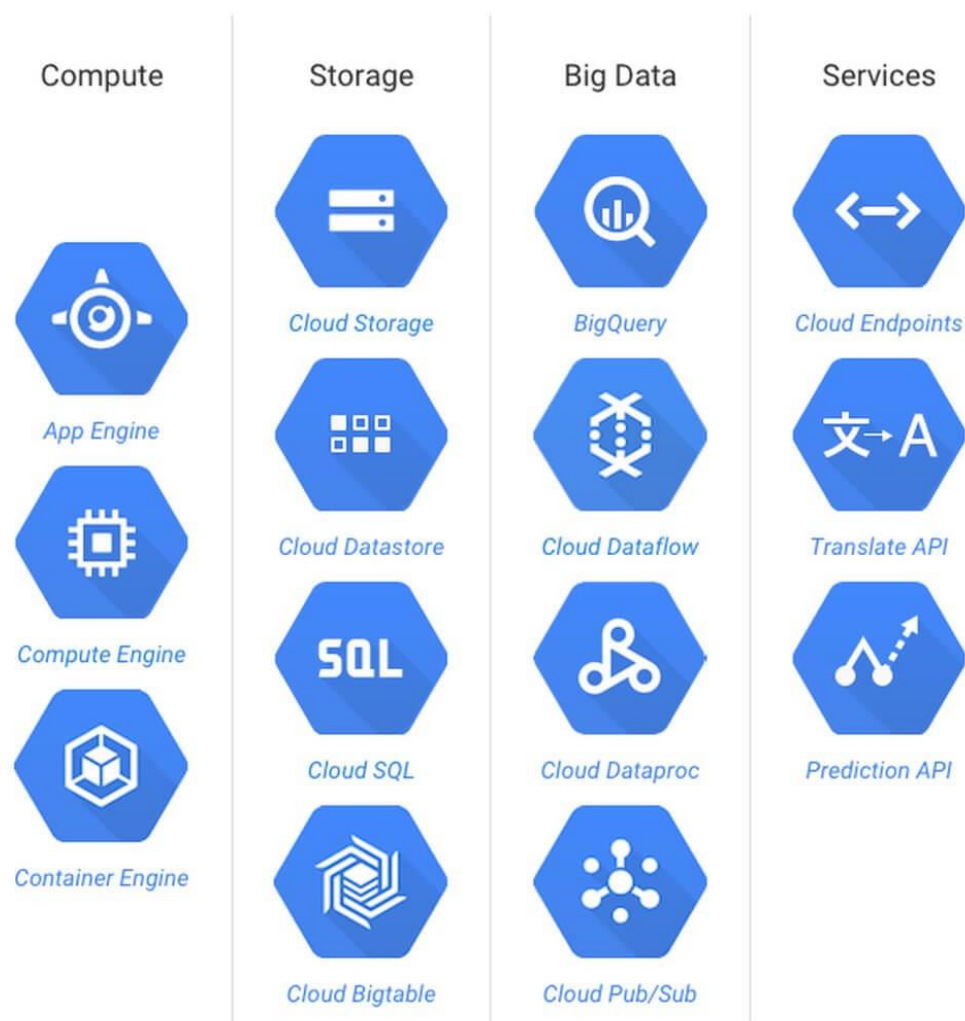
A mobilidade de licenças na nuvem para produtos da Microsoft também é relativamente fácil para clientes com vários produtos da Microsoft em execução no local.

### **8.5. GCP (Google)**

O Google Cloud Platform (GCP) está melhorando lentamente seus recursos de borda e tomou dianteira pela terceira maior fatia do mercado.

Com contínuo investimento para ser um provedor amplo de IaaS e PaaS e expandindo seus recursos, é um nome forte de mercado que vem se fortalecendo dentro do segmento.





Como as outras, tem posições geograficamente diversificadas e os clientes são variados, desde startups até grandes negócios.

No último balanço fornecido pelo Synergy Research Group, o Google Cloud ocupou 10% de todo o montante do segmento em 2021.

### Prós e contras do Google Cloud Platform (GCP)

O Google Cloud Platform (GCP), apesar de atrasado no jogo e com a menor participação de mercado dos provedores de nuvem pública, está mostrando um crescimento nos últimos anos.

Possui vários recursos que o colocam à frente de seus concorrentes em determinadas áreas, principalmente a área de Dados.

O GCP também está pegando onda, não apenas com os novos clientes que já fazem parte do ecossistema, mas também os primeiros usuários da nuvem que desejam expandir seu cenário para o Google como parte de uma estratégia para várias nuvens.



O Google também começou com serviços de PaaS, mas vem expandindo constantemente seu portfólio de produtos.

### **Quando escolher o GCP**

Do ponto de vista da computação, o Google tem o menor número de tamanhos de VM (28 tipos de instância em 4 categorias).

No entanto, ele tem uma característica que torna esses números um pouco irrelevantes.

O Google permite que os usuários criem seus próprios tamanhos personalizados (CPU, memória) para que os clientes possam combinar o tamanho das cargas de trabalho na nuvem com o tamanho no local.

O faturamento também é feito com base na CPU e memória totais usadas, em vez de VMs individuais. Isso reduz o desperdício de capacidade não utilizada.

Outro recurso exclusivo é que o GCP permite que quase todos os tipos de instância conectem GPUs. Isso pode transformar qualquer instância padrão ou personalizada em uma VM pronta para ML.

O Google também foi líder em cobrança por segundo, o que forçou outros CSPs a seguir o exemplo.

Comparado à norma usual do faturamento por hora, o faturamento por segundo reduz muito qualquer desperdício de capacidade. Isso resulta em uma economia de até 40% no geral.

O Google também vinculou ou comprou ferramentas de migração para a nuvem de terceiros.

Essas ferramentas, como Velostrata e CloudPhysics, ajudam os clientes a avaliar, planejar e migrar ao vivo suas VMs para o GCP.

Rede é o destaque do GCP. Eles têm uma rede global de baixa latência. Mesmo da perspectiva do cliente, uma rede VPC abrange todas as suas regiões.

Outros CSPs limitam as redes VPC a uma região. Isso facilita para os clientes do GCP criar aplicativos que atendem aos clientes globalmente, sem criar complexos mecanismos de design de infraestrutura entre regiões e replicação de dados.

Para Bancos NoSQL, o GCP tem um produto chamado BigTable. O BigTable é um banco de dados NoSQL gerenciado em escala de petabytes, usado pelo Google em seus próprios produtos, como o Gmail.

Do ponto de vista de cobrança, o Google oferece descontos automáticos, como descontos de uso sustentado, que reduzem o preço sob demanda se uma VM executar mais de um determinado número de horas em um mês.

Se você deseja o provedor de nuvem mais econômico, o GCP é uma ótima opção.

## 9. SOLUÇÕES EM SISTEMAS DISTRIBUÍDOS

### Objetivo

O objetivo deste capítulo é apresentar algumas soluções de mercado que foram desenvolvidas na arquitetura de sistemas distribuídos, em especial microsserviços.

### Introdução

Empresas globais como Amazon, Coca Cola, Netflix, Spotify e Uber, estão transformando suas infraestruturas de TI em uma arquitetura de microsserviços. Além disso, eles estão reconstruindo suas estruturas organizacionais internas e colocando seus negócios à frente da concorrência.

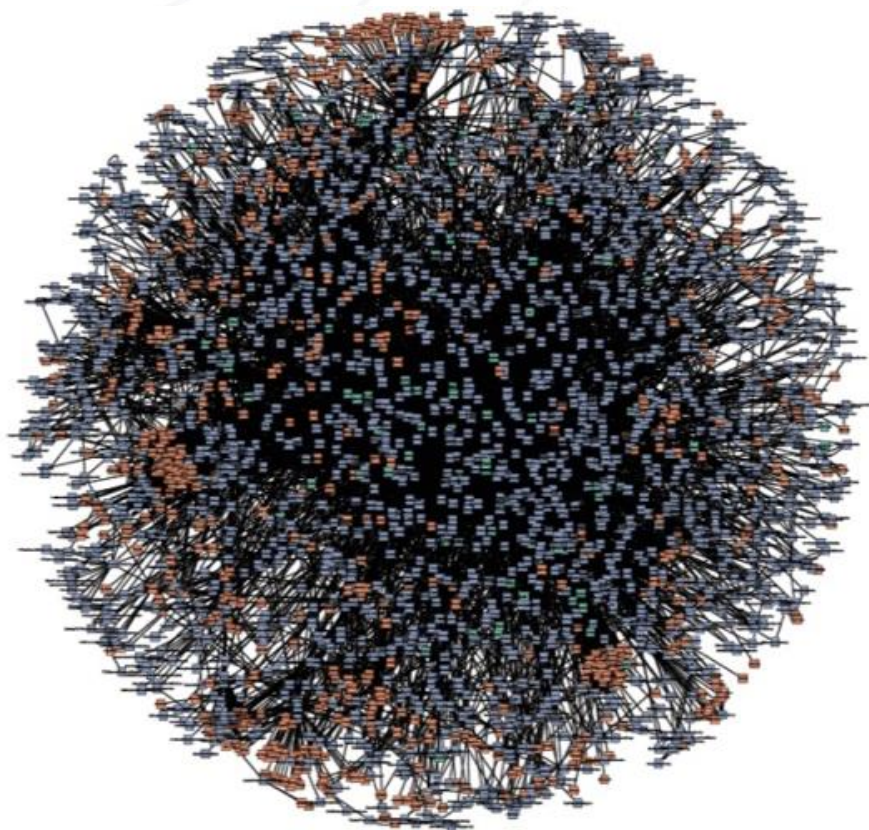
#### 9.1. Amazon

O caso da Amazon é um dos primeiros em que os microsserviços tiveram um papel importante na transformação de todo o negócio. A gigante global alcançou seu sucesso extraordinário nos tempos em que a arquitetura monolítica era “o caminho” de desenvolvimento de sistemas de TI.

Embora a arquitetura da Amazon não fosse exatamente um grande monolito, todos os seus serviços e componentes estavam fortemente acoplados uns aos outros. Com centenas de desenvolvedores espalhados por toda a organização, a Amazon não conseguia mais implantar mudanças de maneira rápida.

Todas as principais alterações de código ficaram presas no pipeline de implantação por semanas antes que os clientes pudessem usá-las. A Amazon usou microsserviços para simplificar e encurtar o pipeline. Dividir as estruturas em aplicativos únicos permitiu que os desenvolvedores percebessem onde estavam os gargalos, qual era a natureza dessas lentidão e permitiu que os desenvolvedores os reconstruíssem como uma arquitetura orientada a serviços, cada um com uma pequena equipe dedicada a um serviço específico.

Arquitetura de microsserviços na Amazon:



O que começou como uma limpeza do sistema acabou sendo uma evolução de um dos maiores players online da arquitetura moderna. Junto com essa mudança, a Amazon abriu caminho para outras empresas e lançou uma série de soluções de código aberto, como AWS (Amazon Web Services) que agora são onipresentes.

## 9.2. Coca-Cola

A Coca Cola Company – 3.800 produtos em todo o mundo e subsidiárias em todos os países do mundo – enfrentou o desafio de conectar entidades em diferentes continentes e apoiar seu crescimento. O grupo de TI global da Coca Cola decidiu alavancar microsserviços e APIs para atingir esse objetivo e substituir gradualmente seu software legado. Nesse caso, mudanças rápidas seriam impossíveis devido a múltiplas soluções implementadas globalmente (ERP, conversões, repositórios).

A empresa sabia que existem muitas maneiras de implementar microsserviços e, no final, decidiu avançar para uma nova arquitetura usando o modelo DevOps. Dentro dele, eles introduziram uma estrutura dividida em aplicativo único, com o qual eles podem resolver quaisquer problemas que surjam em relação à velocidade e agilidade. Por outro lado, foi criada uma biblioteca de módulos reutilizáveis (chamada Anypoint Platform),



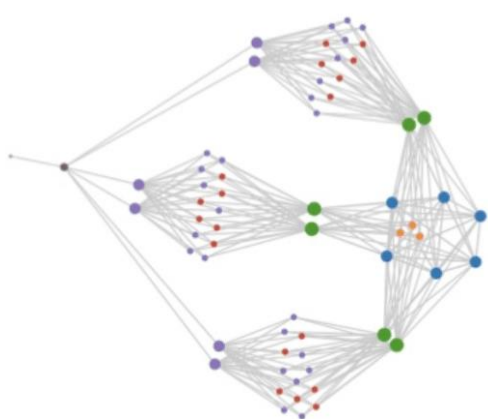
organizada em domínios que estão disponíveis para todas as entidades do grupo. Graças à base de serviços prontos para uso, projetos em toda a organização e de parceiros externos podem ser implementados em menos tempo e com menor custo.

Junto com as mudanças na tecnologia e na arquitetura, a Coca Cola se concentrou em processos e pessoas. A empresa passou do modelo COE para o modelo C4E (Center 4 Excellence) e ganhou sinergias colaborando e compartilhando em todo o sistema.

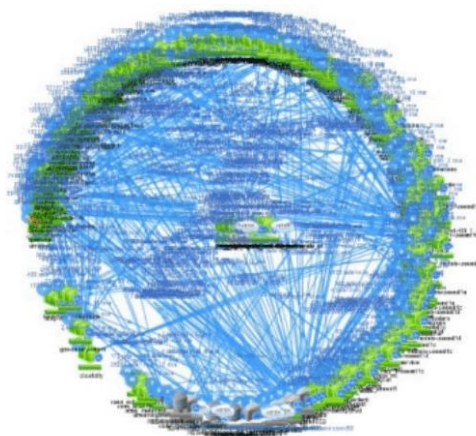
### 9.3. Netflix

A Netflix é uma das primeiras a adotar microserviços e uma das mais discutidas. A história da Netflix se voltando para microserviços começa em 2009, quando essa abordagem não era conhecida. Eles configuram sua arquitetura de microserviços na AWS. Seu processo de transição progrediu em etapas: primeiro, eles mudaram a codificação de filmes e outros aplicativos não voltados para o cliente. Em seguida, eles dissociaram os elementos voltados para o cliente, como inscrições de contas, seleção de filmes, seleção de dispositivos e configuração. A Netflix precisou de 2 anos para dividir seu monólito em microserviços e, em 2011, anunciou o fim de redesenhar sua estrutura e organizá-la usando arquitetura de microserviços.

Arquitetura de microserviços na Netflix:



**Simplified Architecture**

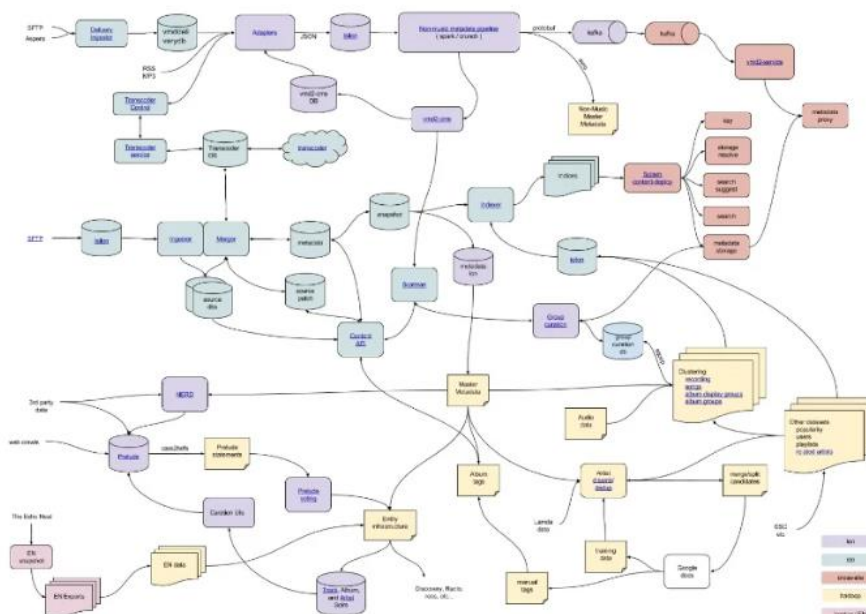


**Actual Architecture**

Hoje, a Netflix aproveita mais de 500 microserviços e API Gateways que lidam com mais de 2 bilhões de solicitações de borda de API diariamente.

## 9.4. Spotify

No momento em que o Spotify tinha mais de 75 milhões de usuários ativos mensalmente, estava procurando uma solução que pudesse ser dimensionada para milhões de usuários, suportasse várias plataformas e lidasse com regras de negócios complexas. Eles queriam ser competitivos em um mercado em rápida evolução, sendo rápidos em reagir e superando a concorrência. Suas equipes de tecnologia encontraram uma maneira de atender aos requisitos acima lançando microsserviços gerenciados por mais de 90 equipes autônomas de pilha completa organizadas em tribos.



## 9.5. Uber

Em seus primeiros dias, quando o Uber estava entrando no mercado, eles construíram sua solução para uma única oferta em uma única cidade. Mas à medida que a empresa se expandia, seu sistema, baseado em uma arquitetura monolítica, começou a causar problemas de escalabilidade e integração contínua. Foi nesse momento que a Uber decidiu transformar seu sistema global de TI em microsserviços.



