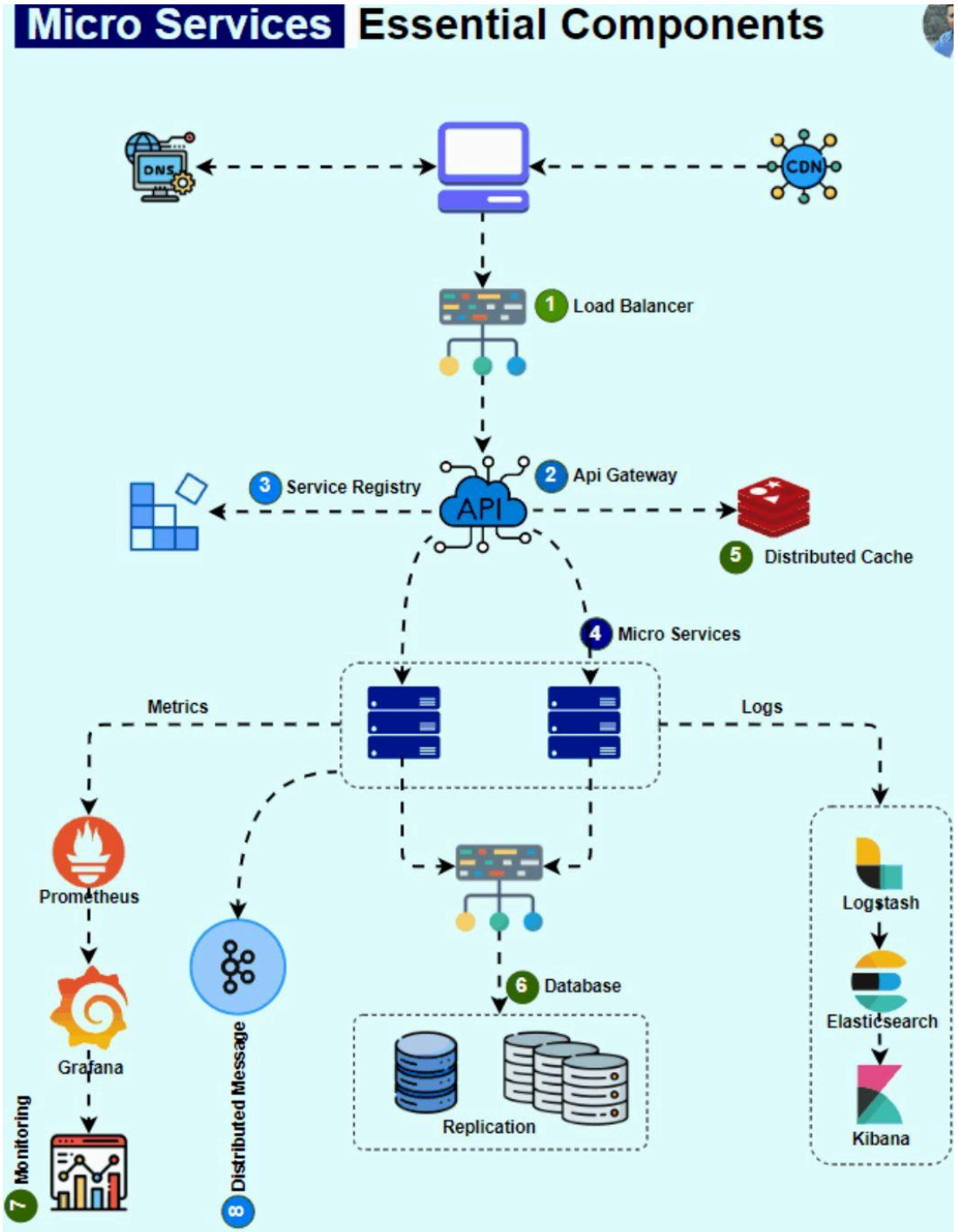
Microservices

- ## Introdução
    - Definição:

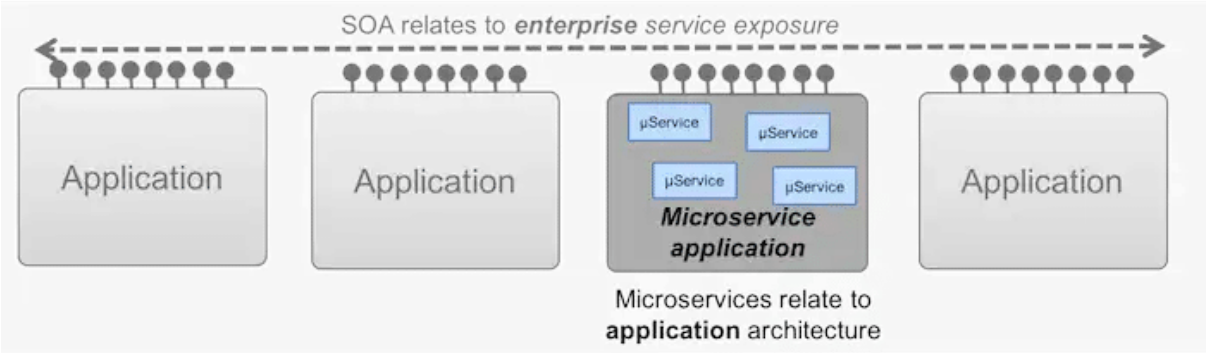
Arquitetura de microservice também chamado de microservice, diz respeito a um estilo de arquitetura para o desenvolvimento de aplicativos. É o desenho de software que permite que um sistema seja dividido em vários serviços menores, individuais e independentes.

Exemplo de arquitetura de microservice



    - História

Em meados dos anos 90 surgiu a arquitetura SOA Service Oriented Architecture facilitando o envio e recebimento de informações de outros sistemas, pois este padrão tornou esta comunicação mais performática. Os microservices surgiram na sequência do SOA e apesar de apresentarem grande semelhanças suas diferenças ficam nítidas quando se olha para seu escopo de negócio. Diferente do SOA, os microservices tem escopo limitado a uma única aplicação, independente de todo restante do sistema e com função única.



- **Principais Características**

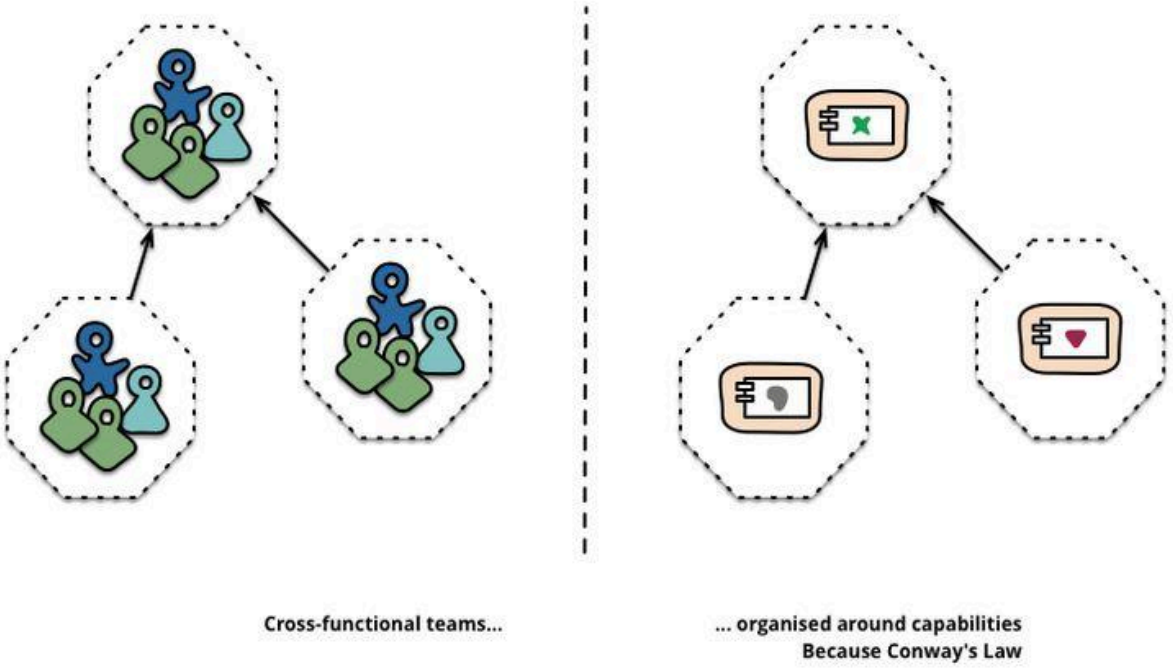
Apesar de não existir uma definição formal do que seria um microservice, é possível descrevê-lo do ponto de vista de suas características comuns em arquiteturas que se encaixem no padrão. Nem todas as arquiteturas de microservice seguem todas as características.

  - Componentização

Arquiteturas de microservice usarão bibliotecas, mas buscam organizar seu próprio software dividindo em serviços. Bibliotecas são definidas como componentes que são usadas em um programa através de chamadas de função diretamente em memória, enquanto serviços são componentes em processos diferentes que se comunicam através de requisições via web services, uma das principais razões para usar serviços como componentes é que podemos publicar serviços de maneira independente, ter uma interface mais explícita.

  - Organização

A abordagem proposta por microservices para esta divisão é organizar os times ao redor de áreas do negócio, assim os serviços possuirão uma determinada área do negócio, incluindo interface do usuário, armazenamento de dados persistente e qualquer outra necessidade, por consequência os times são funcionais.



- Governança descentralizada

Ao quebrar componentes monolíticos em serviços tem-se a escolha de como construir cada um deles de maneira diferente. Significa estabelecer e impor o modo como as pessoas e as soluções trabalham juntas para atingir os objetivos organizacionais.
Não é necessário ter uma governança em tempo de projeto centralizado
Os microservices podem tomar suas próprias decisões sobre seu projeto e sua implementação
Promovem o compartilhamento de serviços comuns/reutilizáveis
Alguns dos aspectos de governança em tempo de execução, SLAs, throttling, monitoramento, requisitos de segurança em nível de API-GW.

- Administração descentralizada de dados

De forma mais abstrata, isto significa que o modelo conceitual de domínio pode ser diferente entre os sistemas. Um exemplo disto é o DDD Domain Driven Design que divide um domínio complexo em múltiplos contextos limitados e mapeia o relacionamento entre eles, assim como os microservices descentralizam decisões sobre os modelos conceituais eles também descentralizam decisões sobre o armazenamento de dados.

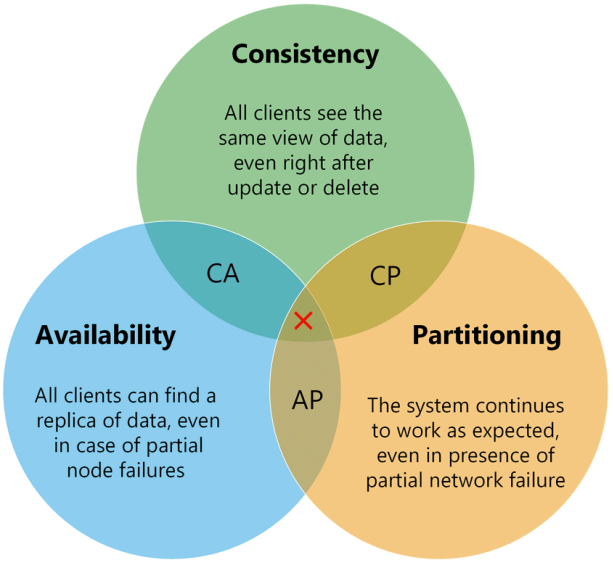
- Projetado para falhar

Uma consequência do uso de serviços como componentes é que as aplicações precisam ser desenhadas de maneira que possam tolerar a falha dos serviços. Uma vez que os serviços podem falhar a qualquer momento, é importante ser capaz de detectar falhas rapidamente e se possível restaurar o serviço automaticamente. Devem possuir bastante ênfase no monitoramento em tempo real checando elementos de arquitetura e métricas relevantes.

- Teorema CAP

O teorema CAP afirma que um sistema distribuído não pode apresentar mais de duas das seguintes características ao mesmo tempo:

a) Consistência: cada requisição recebe a versão mais atualizada do dado, isto é, recebe a última leitura bem sucedida.
b) Disponibilidade: cada requisição recebe uma resposta em tempo hábil, mesmo que não seja a versão mais atualizada dos dados.
c) Tolerância à partição: o sistema continua a funcionar mesmo que um número arbitrário de mensagens entre os nós sejam perdidas.



- ## Vantagens e desvantagens

Vantagens

Escalabilidade de pessoas/times:

- Embora não seja a mais falada, uma das principais vantagens é facilitar que grandes times de desenvolvimento trabalhem em um mesmo produto ao mesmo tempo, uma vez que promove uma segregação de componentes, cada um com suas responsabilidades.

Flexibilidade operacional:

- O fato de serem deployados separadamente traz maior confiança e flexibilidade aos operadores, sendo um incentivo para a realização de deploys com maior frequência.
- Permite o monitoramento de cada serviço individualmente.

Escalabilidade:

- Facilita a escalabilidade horizontal, uma vez que os serviços podem ser replicados de acordo com a carga recebida por cada um deles.

Desvantagens

Bancos separados:

- A separação dos bancos de dados tira a possibilidade de se utilizar chaves estrangeiras para estabelecer relacionamentos entre tabelas em bancos diferentes.

Aumento da complexidade:

- Microserviços são mais difíceis e complexos de planejar, implementar e monitorar.
- A comunicação entre os serviços precisa ser corretamente coordenada e orquestrada, o que é mais difícil do que simplesmente invocar uma função local.
- É preciso preparar o sistema para ser tolerante a falhas, uma vez que não é incomum que um serviço fique fora do ar ou apresente alta latência.

- **Apresentação do código, overview de um projeto.**

Referência bibliográfica:

- https://cloud.google.com/learn/what-is-microservices-architecture?hl=pt-BR
- https://codigo35.com/2016/01/02/microsservicos/
- https://medium.com/xp-inc/microsservi%C3%A7o-vantagens-e-desvantagens-c89f227ef1f9
- https://aws.amazon.com/pt/microservices/
- https://blog.casadodesenvolvedor.com.br/arquitetura-de-microsservicos/