
Newton
Quem se prepara, não para.

Arquitetura de Aplicações Web

5º período

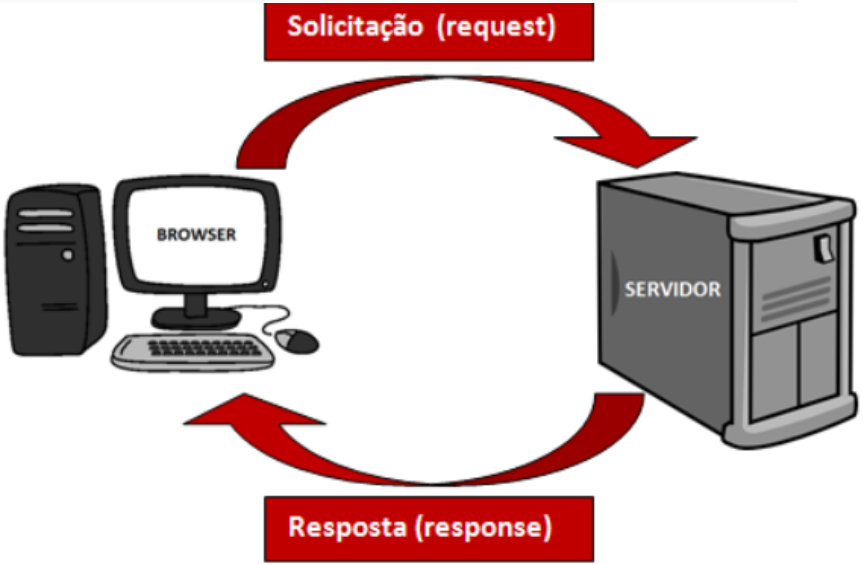
Professora: Michelle Hanne

Como Funciona as Aplicações Web

Newton
Quem se prepara, não para.

Requisição

```
GET /index.html HTTP/1.1
Host: www.twitter.com
```



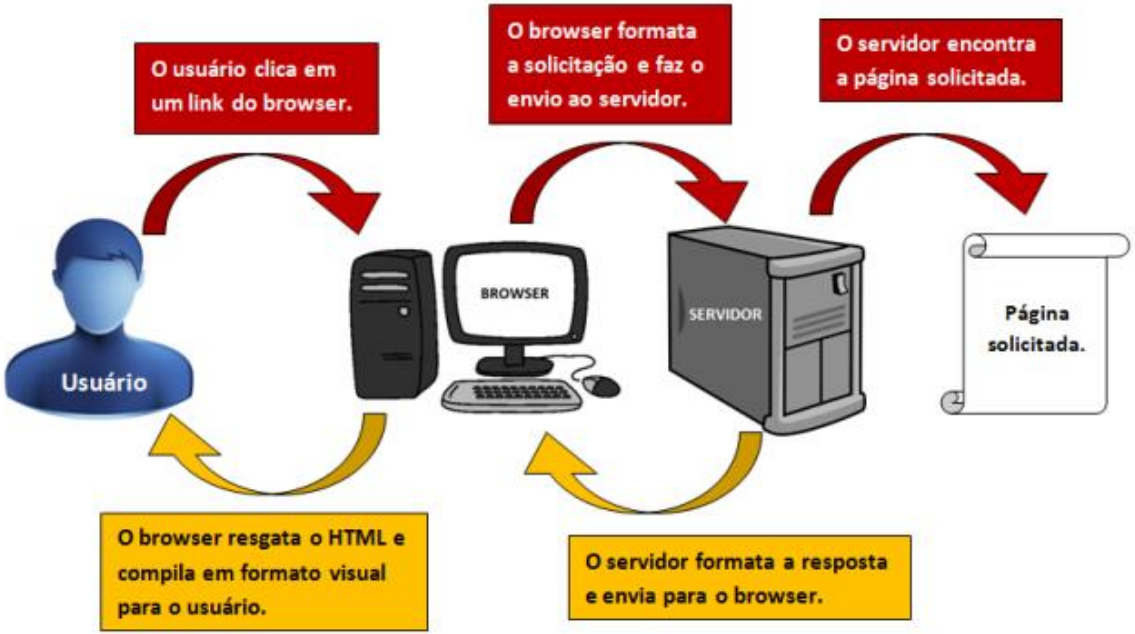
A **função do servidor web** é receber uma solicitação (requisição) e devolver (resposta) algo para o cliente.

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-
Hat/Linux)
Content-Type: text/html; charset=UTF-8
Content-Length: 131
<!DOCTYPE html>
<html>
<head>
<title>Twitter</title>
</head>
<body>
            Olá mundo, este é um tweet.
</body>
</html>
```

Fonte: https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888

Como Funciona as Aplicações Web

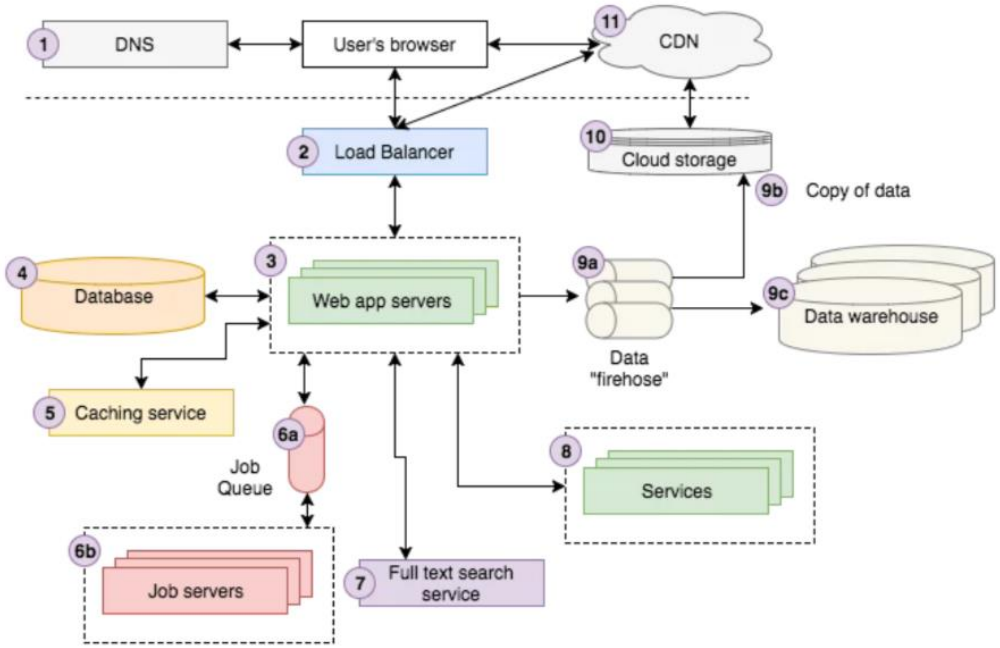


Protocolo HTTP

- Além do servidor web, podemos ter um **back-end.**
- **Back-end** é uma aplicação escrita por nós, executada pelo **servidor web.**
- Cada requisição é recebida pelo servidor web, que a delega ao back-end.
- Também podemos ter um SGBD

Fonte: https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888

Como Funciona as Aplicações Web



CDN (Content Delivery Network - Rede de Entrega de Conteúdo) é um grupo de servidores geograficamente distribuídos que aceleram a entrega do conteúdo da Web. Armazenam conteúdo em cache, como páginas da Web, imagens e vídeos em servidores proxy próximos à sua localização física.

Fonte: http://carozorodriguez.adv.br/introducao-a-arquitetura-web/

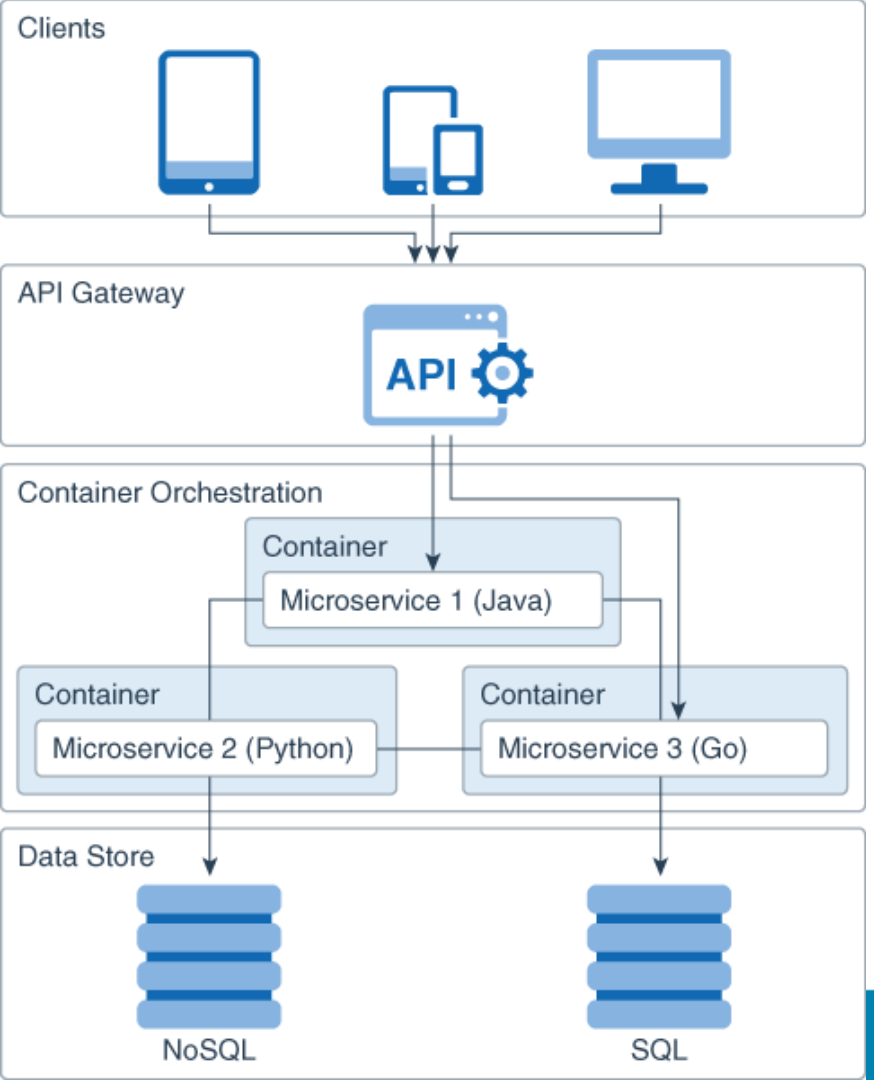
Arquitetura de Microsserviços

Newton
Quem se prepara, não para.

Microsserviços são aplicações desmembradas em serviços independentes. Para isso, os serviços se comunicam entre si usando APIs.

Em uma arquitetura de microsserviço, **cada função define um serviço e é criada e implementada de modo independente**. Essa característica implica no fato de que cada serviço, de forma individual, pode funcionar ou apresentar falha sem comprometer os demais.

A tendência se tornou popular nos últimos anos, à medida que as empresas buscam se tornar mais ágeis e avançar em direção a um DevOps e testes contínuos.



Arquitetura de Microsserviços

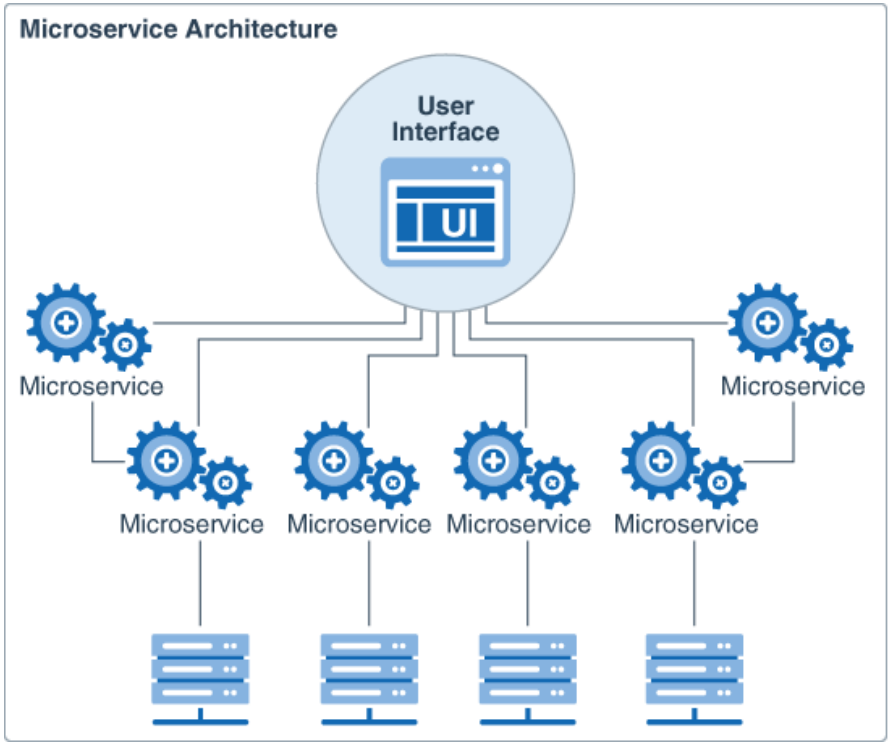
Newton
Quem se prepara, não para.

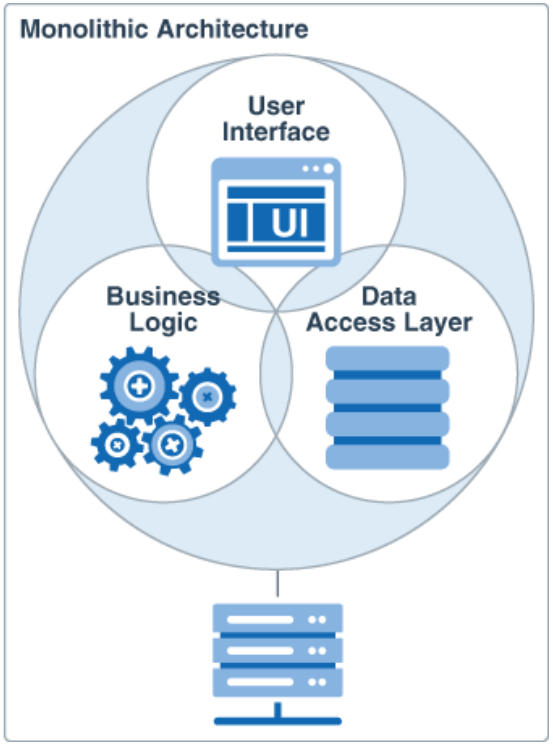
Os microsserviços seguem o modelo share-nothing e são executados como **processos sem estado**. Essa abordagem facilita a escala e a manutenção do aplicativo.

- **A camada API** é o ponto de entrada de todas as solicitações do cliente para um microsserviço. A comunicação ocorre através de protocolos: HTTP, gRPC e TCP/UDP.
- **A camada lógica** se concentra em uma única tarefa de negócios, minimizando as dependências dos outros microsserviços.
- **A camada de armazenamento** de dados fornece um mecanismo de persistência, como um mecanismo de armazenamento de banco de dados
- ***Normalmente, cada microsserviço é executado em um contêiner que fornece um ambiente de tempo de execução leve.***

Arquitetura Monolítica vs Microsserviços

Newton
Quem se prepara, não para.





Fonte: https://docs.oracle.com/pt-br/solutions/learn-architect-microservice/index.html#GUID-1A9ECC2B-F7E6-430F-8EDA-911712467953

Arquitetura Monolítica vs Microsserviços

Newton
Quem se prepara, não para.

Característica	Arquitetura de microsserviços	Arquitetura Monolítica
Desenho da unidade	O aplicativo consiste em serviços acoplados vagamente. Cada serviço suporta uma única tarefa de negócios.	Todo o aplicativo é projetado, desenvolvido e implantado como uma única unidade.
Reutilização da funcionalidade	Microsserviços definem APIs que expõem sua funcionalidade a qualquer cliente. Os clientes podem até ser outras aplicações.	A oportunidade de reutilizar a funcionalidade entre aplicativos é limitada.
Comunicação no âmbito do pedido	Para se comunicar entre si, os microsserviços de um aplicativo usam o modelo de comunicação solicitação-resposta. A implementação típica usa chamadas de API REST com base no protocolo HTTP.	Os procedimentos internos (chamadas de função) facilitam a comunicação entre os componentes do aplicativo. Não é necessário limitar o número de chamadas de procedimento interno.
Flexibilidade tecnológica	Cada microservice pode ser desenvolvido usando uma linguagem de programação e estrutura que melhor se adapte ao problema que o microservice é projetado para resolver.	Geralmente, todo o aplicativo é escrito em uma única linguagem de programação.
Gerenciamento de dados	Descentralizado: Cada microsserviço pode usar seu próprio banco de dados.	Centralizado: todo o aplicativo usa um ou mais bancos de dados.
Implantação	Cada microservice é implantado de forma independente, sem afetar os outros microservices no aplicativo.	Qualquer alteração, por menor que seja, requer a reimplantação e reinicialização de todo o aplicativo.
Capacidade de Manutenção	Os microsserviços são simples, focados e independentes. Portanto, o aplicativo é mais fácil de manter.	À medida que o escopo do aplicativo aumenta, a manutenção do código se torna mais complexa.
Resiliência	A funcionalidade do aplicativo é distribuída em vários serviços. Se um microservice falhar, a funcionalidade oferecida pelos outros microservices continuará disponível.	Uma falha em qualquer componente pode afetar a disponibilidade de todo o aplicativo.
Escalabilidade	Cada microsserviço pode ser dimensionado independentemente dos outros serviços.	Todo o aplicativo deve ser dimensionado, mesmo quando o requisito de negócios é dimensionar apenas determinadas partes do aplicativo.

Fonte: https://docs.oracle.com/pt-br/solutions/learn-architect-microservice/index.html#GUID-1A9ECC2B-F7E6-430F-8EDA-911712467953

Exemplos de Microsserviços

Newton
Quem se prepara, não para.

- **Muitos gigantes da web – incluindo Amazon, Netflix, Twitter, PayPal, Spotify e The Guardian – adotaram com sucesso a arquitetura de microsserviço.**

- Barra de busca em um site, markenting place ou e-commerce
- Recomendações de produtos relacionados ao que está buscando em um e-commerce
- Adicionar produtos no carrinho
- Realização de checkout

Netflix é um dos melhores exemplos de implementação de arquitetura de microsserviços. Em 2009, a Netflix mudou de uma arquitetura monolítica para microsserviços devido à crescente demanda por seus serviços. Porém, como não existiam microsserviços na época, os engenheiros da Netflix criaram uma tecnologia de código aberto.

Fonte: https://ltsconsulting.com.br/arquitetura-de-microsservicos-exemplos-escalaveis/#:~:text=Embora%20j%C3%A1%20exista%20h%C3%A1%20algum,sucesso%20a%20arquitetura%20de%20microsservi%C3%A7o.

Exemplos de Microsserviços

Newton
Quem se prepara, não para.

Uber: começou com uma arquitetura monolítica. Era mais simples para os fundadores da empresa quando forneciam aos clientes apenas o serviço UberBLACK. Mas, como a inicialização cresceu rapidamente, os desenvolvedores decidiram mudar para microsserviços para usar várias linguagens e estruturas. **Agora, o Uber tem mais de 1.300 microsserviços com foco na melhoria da escalabilidade do aplicativo.**

Spotify: com mais de 172 milhões de usuários premium (em 2021), os fundadores do Spotify decidiram construir um sistema com componentes escalonáveis de forma independente para tornar a sincronização mais fácil. **Para o Spotify, o principal benefício dos microsserviços é a capacidade de prevenir falhas massivas. Mesmo que vários serviços falhem simultaneamente, os usuários não serão afetados.**

Fonte: https://ltsconsulting.com.br/arquitetura-de-microsservicos-exemplos-escalaveis/#:~:text=Embora%20j%C3%A1%20exista%20h%C3%A1%20algum,sucesso%20a%20arquitetura%20de%20microsservi%C3%A7o.

Relembrando o HTTP

Newton
Quem se prepara, não para.

- Protocolo para comunicação entre cliente e servidor
- Modelo de requisição e resposta
- Gere recursos na web, que podem ser:
  - Páginas
  - Imagens
  - Scripts
  - Folhas de estilo
  - Fontes
  - Vídeo etc.
- Operações sobre recursos feitas pelos verbos HTTP (GET, POST…)
- Recursos identificados por URLs
- Prevê possibilidade de caching de recursos
- É totalmente stateless

Fonte: https://fegemo.github.io/cefet-web/classes/ssn6/

Ideia do Rest

Newton
Quem se prepara, não para.

Dados como recursos

- Os dados são vistos como um recurso HTTP (assim como uma imagem, uma página HTML etc.)
- Cada informação exposta pelo banco de dados tem uma URL
- Operações (buscar, excluir, atualizar etc.) são realizadas na informação usando verbos HTTP (GET, DELETE, POST, PUT etc.)

Stateless (sem estado)

- Nenhum contexto é armazenado após o atendimento de uma requisição

"Cacheável"

- Clientes podem guardar as respostas, se interessante

Uniformidade de interface

- As operações e as URLs são padronizadas e fáceis de infereir

Ideia do Rest

Newton
Quem se prepara, não para.

API: conjunto de métodos públicos de um programa

API REST: conjunto de métodos públicos expostos por meio de um *web service* na arquitetura REST

- Proposto por Roy Fielding em 2000 (tese de doutorado)
- Abordagens: purista ou “inspirada”
- Como fazer?
  - Identifique os recursos de dados do banco
  - Identifique as operações sobre recursos que são permitidas
  - Implemente os métodos para cada recurso, respondendo possivelmente em vários formatos (.html, .json, .xml)

Referências

https://docs.oracle.com/pt-br/solutions/learn-architect-microservice/index.html#GUID-1A9ECC2B-F7E6-430F-8EDA-911712467953

Node.js in Action, First Edition,
Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich