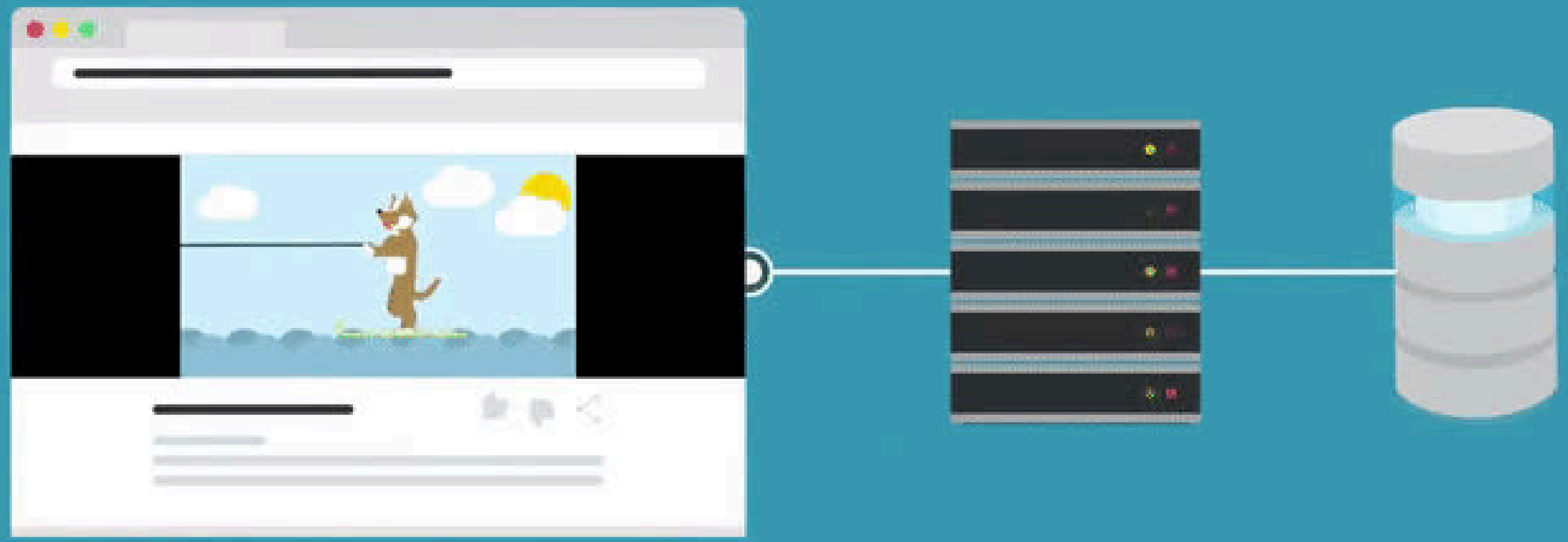
INTRODUÇÃO AO BACKEND

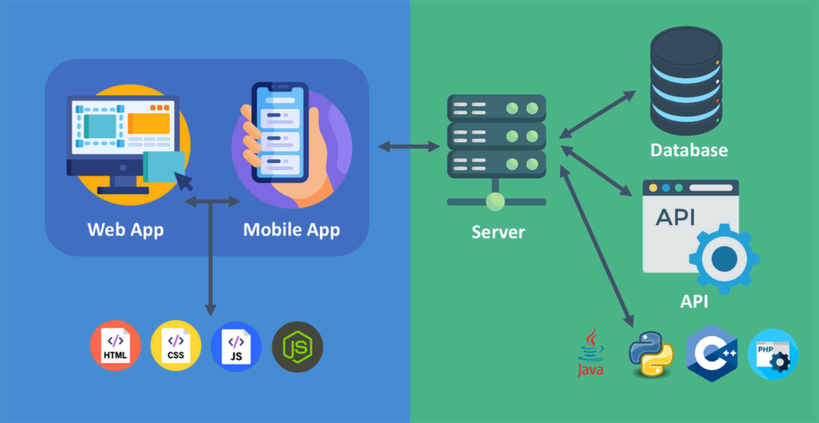
O QUE É BACKEND?

O *backend* é a parte de uma aplicação que roda no servidor e é responsável por **processar lógica**, **armazenar** e **manipular dados**, **autenticar usuários** e **fornecer respostas** para o *frontend*. Ele é invisível para o usuário final, mas é essencial para o funcionamento de sistemas dinâmicos.



DIFERENÇAS ENTRE FRONTEND E BACKEND

Frontend	Backend
Interface e interação do usuário	Processamento de dados e regras de negócio
HTML, CSS, JavaScript (React, Vue, etc.)	JavaScript (Node.js), Python, Java, PHP, etc.
Navegador do usuário	Servidor
LocalStorage, Cookies	Banco de Dados

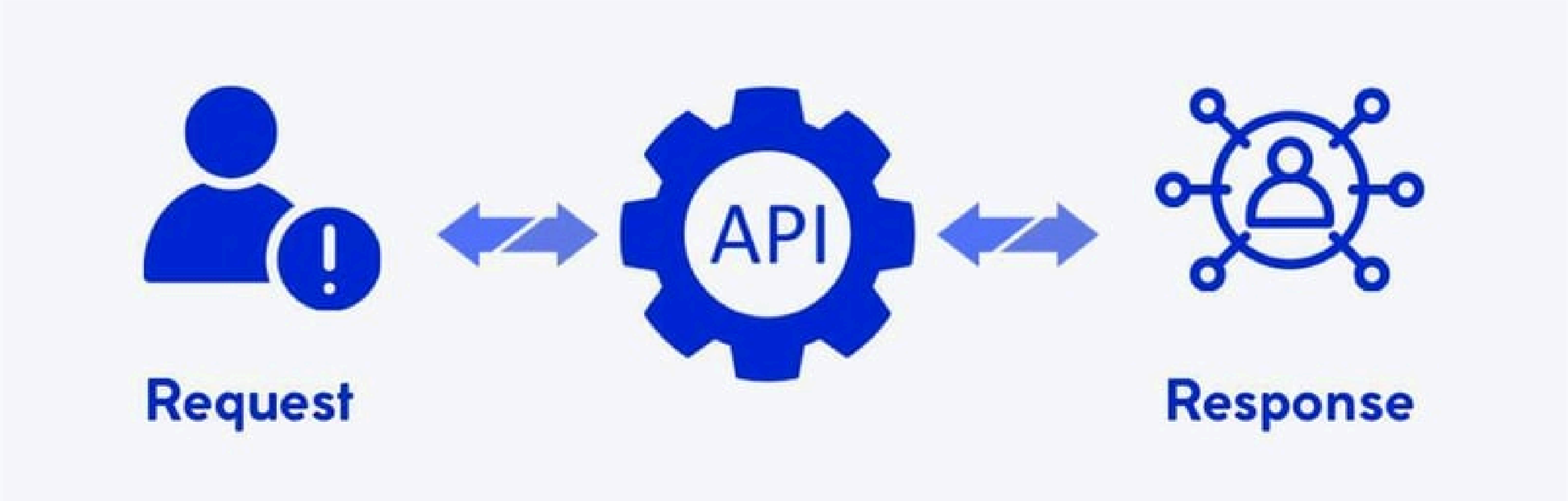


PRINCIPAIS TECNOLOGIAS DE BACKEND

- **Linguagens de Programação**: Node.js (JavaScript), Python, PHP, Java, C#.
- **Frameworks e Bibliotecas**: Express.js (Node.js), Django (Python), Spring Boot (Java), Laravel (PHP).
- **Banco de Dados**: MySQL, PostgreSQL, MongoDB.
- **APIs**: REST, GraphQL, WebSockets.

COMO O BACKEND SE COMUNICA COM O FRONTEND?

O *frontend* faz **requisições HTTP** para o *backend* por meio de **APIs**. O backend **recebe a requisição**, **processa os dados** e **devolve uma resposta**.



O PROCESSO REQUEST-RESPONSE

LINHA DE SOLICITAÇÃO

Na linha de solicitação, presente em uma solicitação HTTP, são especificados três elementos principais: o método HTTP, a URI (Uniform Resource Identifier) e a versão do protocolo HTTP.
Exemplo de linha de solicitação:
GET /exemplo/recurso HTTP/1.1
Neste exemplo, "GET" é o método HTTP, "/exemplo/recurso" é a URI, e "HTTP/1.1" é a versão do protocolo.
Uma mensagem inteira de requisição HTTP do servidor poderia se parecer com isto:

```
GET / HTTP/1.1
Host: www.meusite.com.br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geckc
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

LINHA DE STATUS

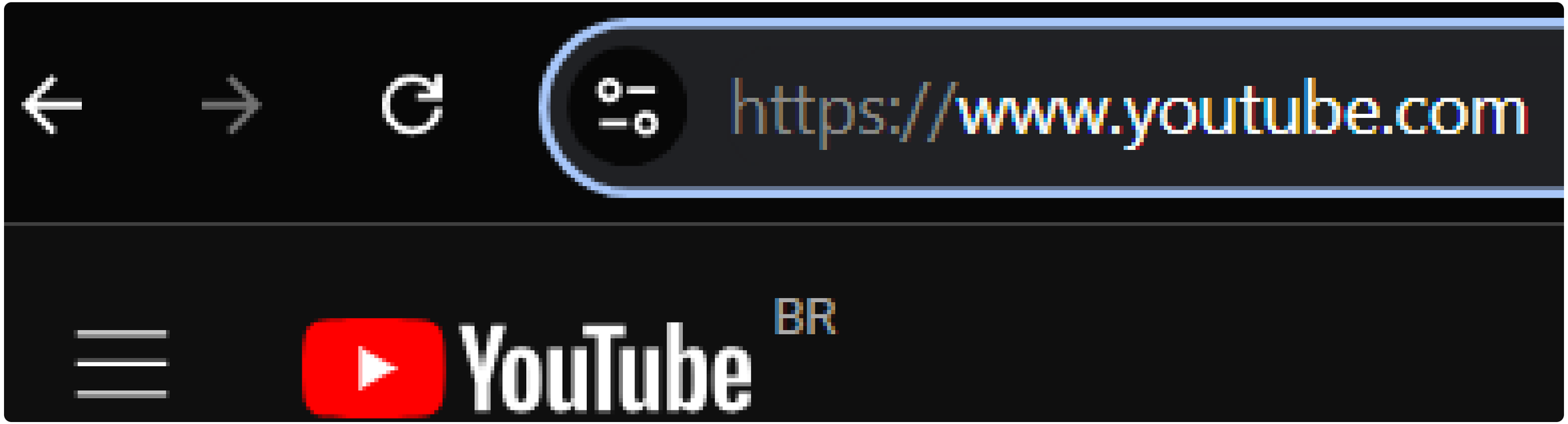
Na linha de status, presente em uma resposta HTTP, são fornecidos três elementos principais: a versão do protocolo HTTP, um código de status e uma mensagem de status.
Exemplo de linha de status:
HTTP/1.1 200 OK
Neste exemplo, "HTTP/1.1" é a versão do protocolo, "200" é o código de status indicando sucesso, e "OK" é a mensagem de status associada.
Uma mensagem inteira de resposta HTTP do servidor poderia se parecer com isto:

```
HTTP/1.1 200 OK
Date: Sun, 09 Apr 2023 12:28:53 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sat, 08 Apr 2023 23:11:04 GMT
Content-Length: 612
Content-Type: text/html; charset=UTF-8
Connection: closed
```

```
<html>
<head>
<title>Exemplo de Site</title>
</head>
<body>
<h1>Olá, Mundo!</h1>
<p>Este é um exemplo de corpo de resposta HTTP.</p>
</body>
</html>
```

QUAL É A DIFERENÇA ENTRE HTTP E HTTPS?

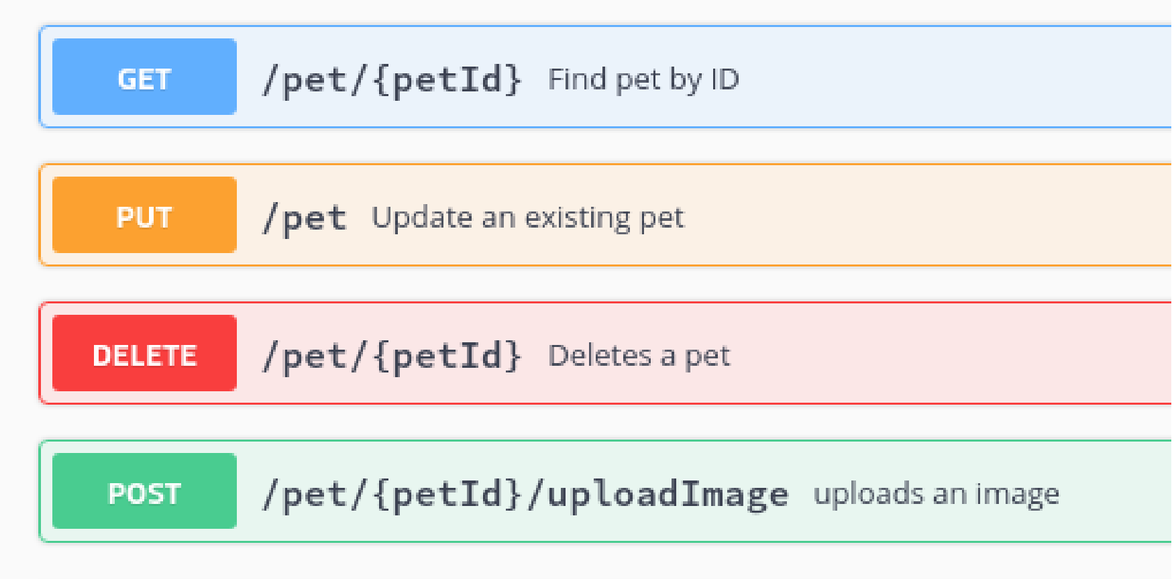
Compreendemos que o HTTP é um protocolo de comunicação entre cliente e servidor. Mas, você também já pode ter visto em algum site HTTPS, ao invés de HTTP. Afinal, qual a diferença?
Exemplo de uso do HTTPS no site do Youtube.



O HTTPS (Hypertext Transfer Protocol Secure) é uma versão segura do HTTP, protegendo a transmissão de dados com criptografia SSL/TLS. Enquanto o HTTP envia informações sem segurança, o HTTPS garante integridade e privacidade, essencial para transações online e dados sensíveis.

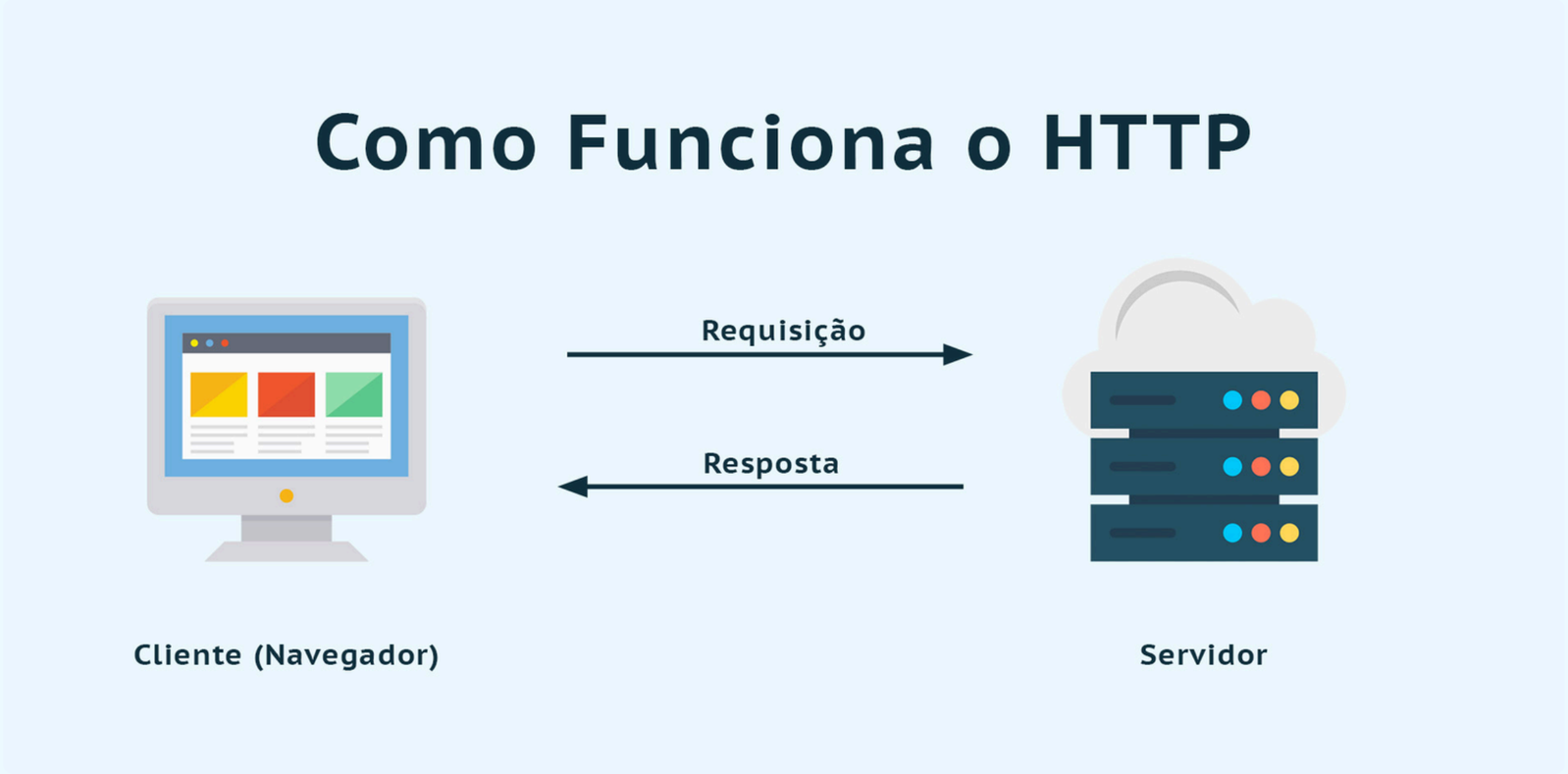
MÉTODOS

- GET: Solicita dados de um recurso específico, geralmente usado para recuperar informações de uma URL, com a opção de passar parâmetros na URL e incluir headers.
- POST: Envia dados ao servidor para processamento, frequentemente usado para formulários ou criação de recursos.
- PUT: Cria ou atualiza um recurso específico, substituindo completamente o recurso existente.
- DELETE: Solicita a remoção de um recurso específico no servidor.

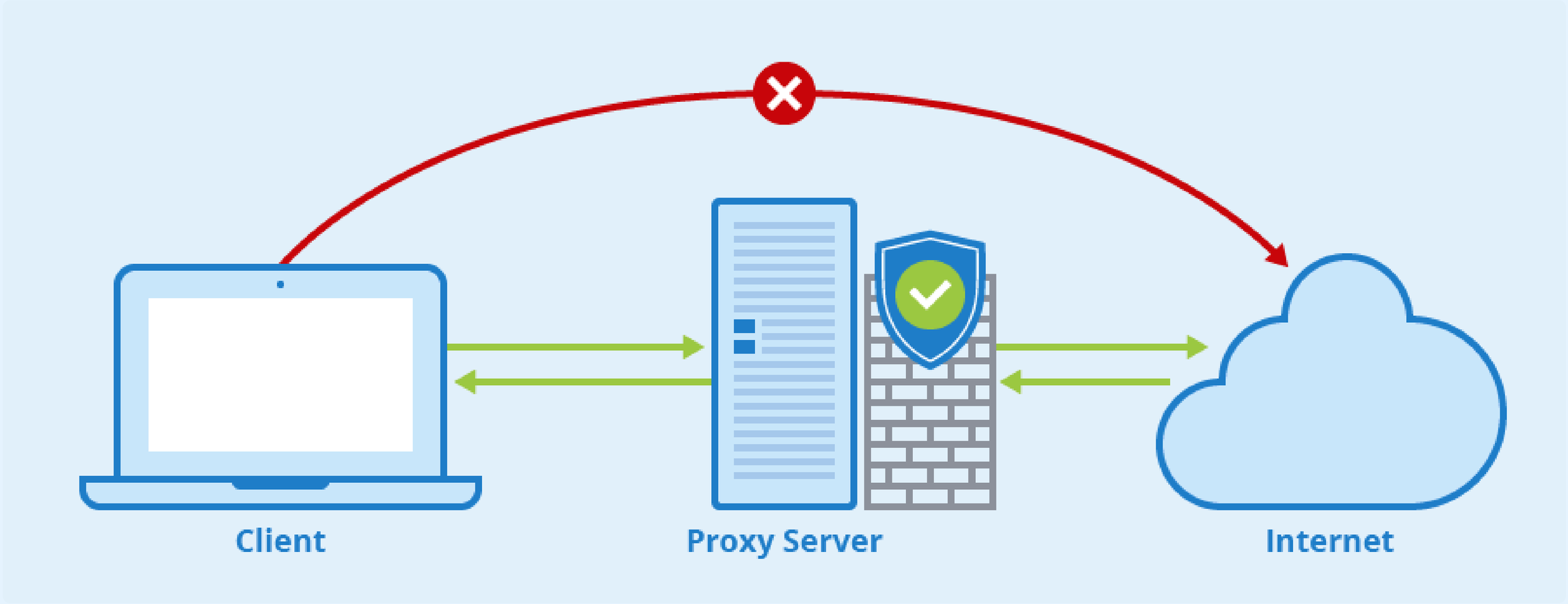


PROTOCOLO HTTP

A sigla HTTP vem de Hypertext Transfer Protocol. Traduzido para o português, HTTP significa “**Protocolo de Transferência de Hipertexto**”.
O termo “hipertexto” descreve um sistema de organização de informações em que documentos têm conexões clicáveis, permitindo aos usuários saltar de uma parte do texto para outra de maneira não linear.



INFRAESTRUTURA DO PROTOCOLO HTTP



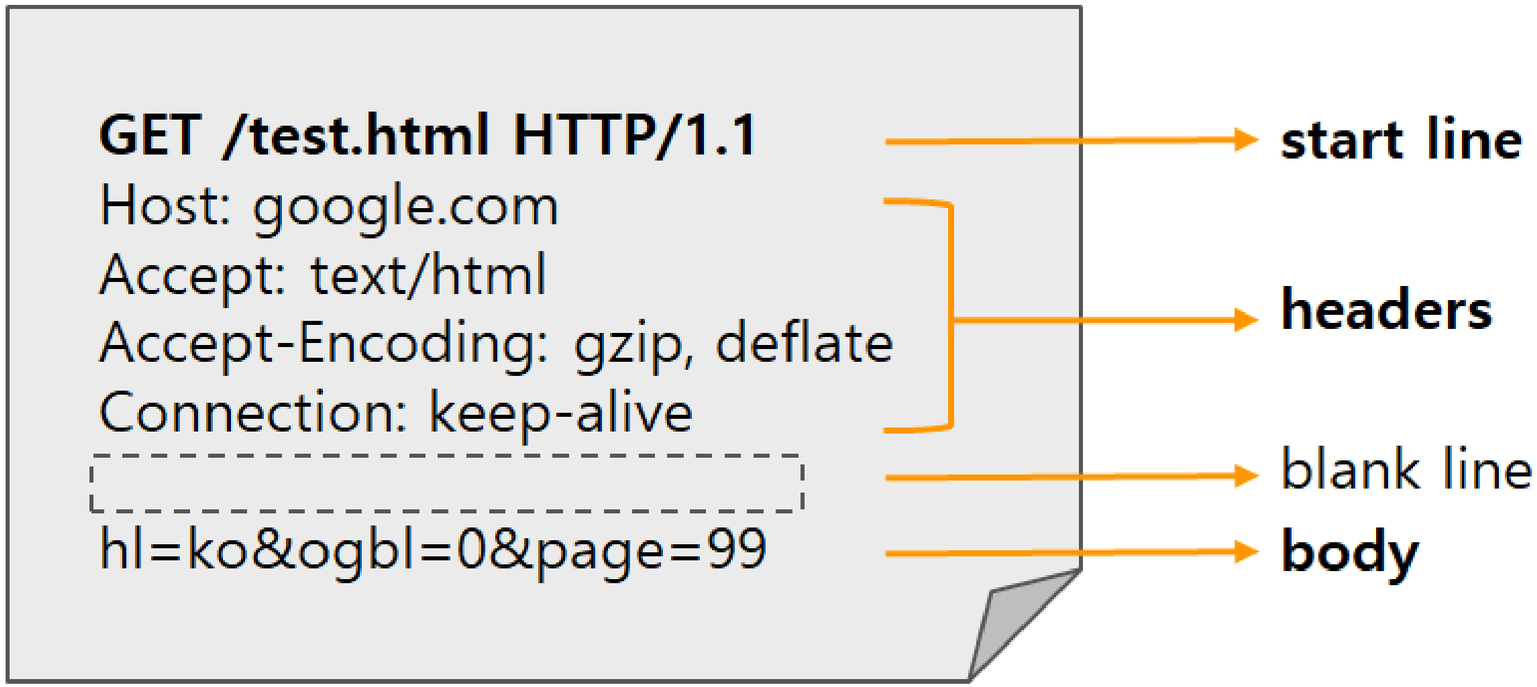
INFRAESTRUTURA DO PROTOCOLO HTTP

Cliente: Um cliente é um dispositivo ou software que interage com recursos na web em nome do usuário. Pode ser um navegador, aplicativo móvel ou software automatizado, que faz solicitações HTTP para acessar informações ou serviços em servidores.

Servidor: O servidor é o dispositivo que hospeda e fornece recursos solicitados pelos clientes. Ele processa as requisições, executa a lógica necessária e retorna as respostas, como páginas web ou dados. A inclusão do cabeçalho Host no HTTP 1.1 permite que múltiplos sites sejam hospedados em um único servidor com o mesmo endereço IP.

Proxies: Proxies são intermediários entre clientes e servidores. Existem dois tipos principais: forward proxies, que agem em nome dos clientes para buscar recursos dos servidores, e reverse proxies, que operam em nome dos servidores para receber solicitações dos clientes. Além de otimizar o tráfego e melhorar a eficiência da rede, proxies oferecem segurança, anonimato, filtragem de conteúdo malicioso, controle de acesso e balanceamento de carga. Eles também podem acelerar o carregamento de páginas e fornecer logging e monitoramento.

ASPECTOS FUNDAMENTAIS DO HTTP NA COMUNICAÇÃO



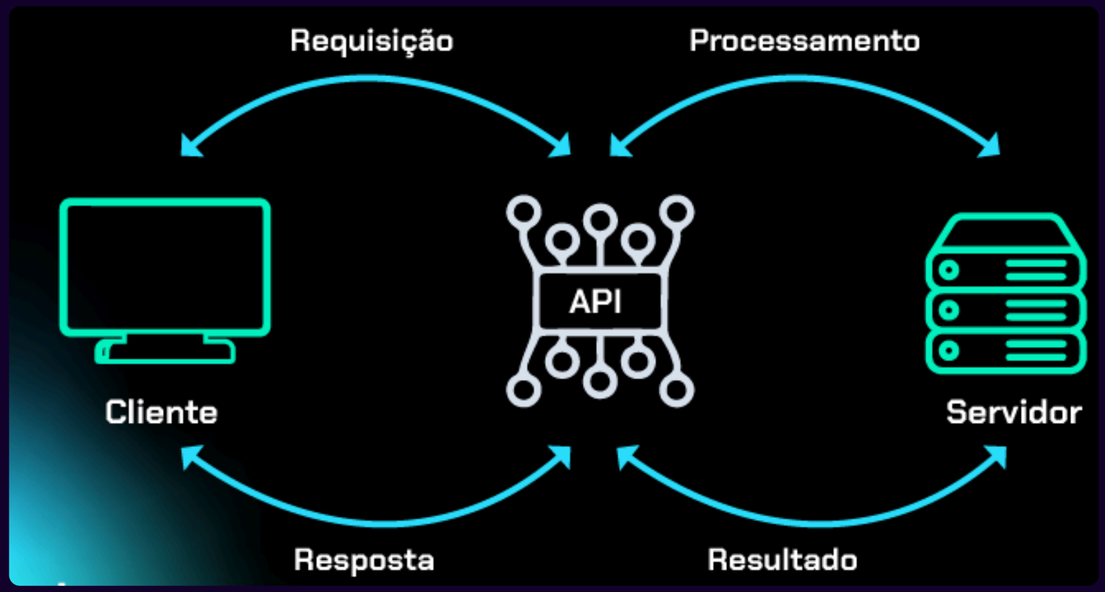
ASPECTOS FUNDAMENTAIS DO HTTP NA COMUNICAÇÃO

- Métodos HTTP O HTTP utiliza métodos, como GET, POST, PUT e DELETE, para indicar a ação desejada na solicitação. Esses métodos definem operações comuns, como obter dados, enviar dados para processamento, atualizar ou excluir recursos.
- Headers (Cabeçalhos) Os cabeçalhos HTTP contêm informações adicionais sobre a requisição ou a resposta. Eles incluem dados como o tipo de conteúdo, a data da requisição, cookies, e muitos outros.
- URI (Uniform Resource Identifier) Os recursos na web são identificados por URLs (Uniform Resource Locators) ou URIs. Uma URI é uma sequência de caracteres que identifica um nome ou um recurso na web.
- Cache O HTTP possui mecanismos de cache para melhorar o desempenho. Os cabeçalhos de controle de cache indicam se o navegador do cliente ou intermediários podem armazenar em cache uma resposta HTTP e, caso possível, por quanto tempo e em quais condições.
- Tipo de hipermídia
- O tipo de hipermídia comum no contexto do HTTP é o HTML, utilizado para criar e apresentar documentos na web. Porém, o HTTP suporta uma variedade de tipos de mídia, como XML, JSON, imagens e vídeos, permitindo a transmissão de diversos tipos de dados online.

API REST

COMO FUNCIONA UMA API?

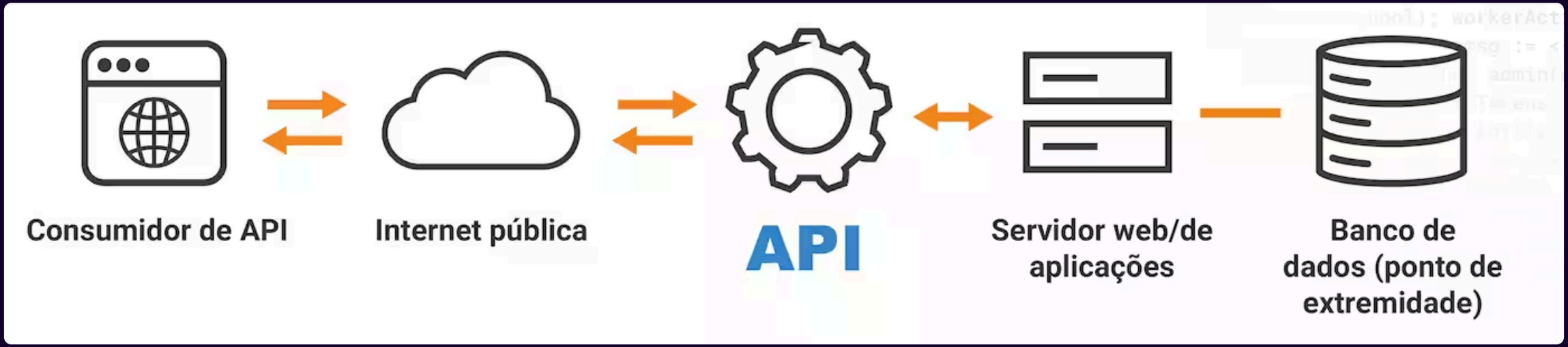
As APIs desempenham um papel de conectividade e interação entre diferentes sistemas e aplicações.



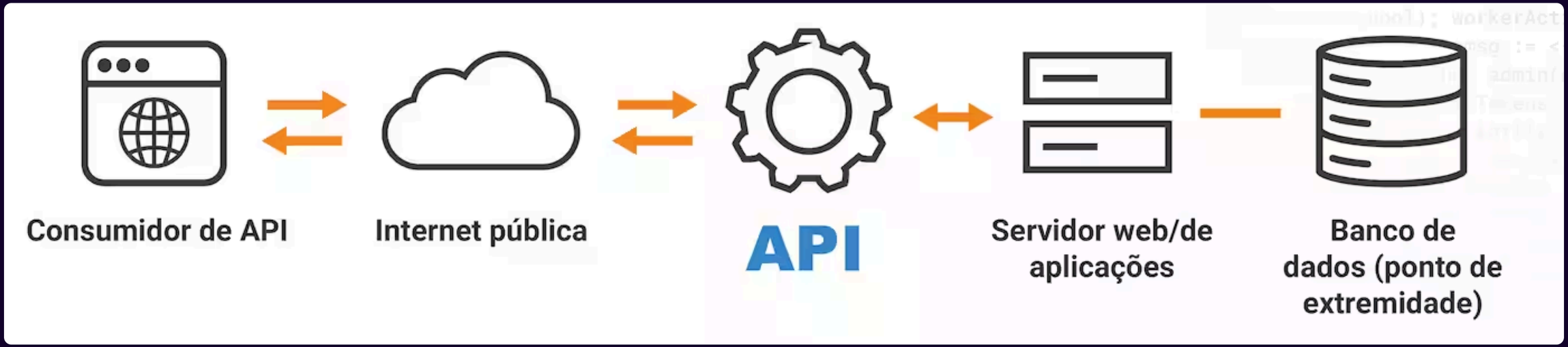
API é um acrônimo em inglês para **Application Programming Interface.**
De forma geral, é um conjunto de padrões, ferramentas e protocolos que permite a criação mais simplificada e segura de plataformas, pois permite a integração e a comunicação de softwares e seus componentes.

APIS BASEADAS EM WEB

A API baseadas em Web Serve como uma ponte eficaz entre o cliente e o servidor. Para isso, utiliza protocolos da web, como **HTTP**.
Assim, permite a comunicação entre sistemas pela internet e, consequentemente, que aplicações diferentes interajam e compartilhem dados de forma padronizada, independentemente da tecnologia utilizada em cada extremidade.



APIS BASEADAS EM WEB



Seu funcionamento acontece da seguinte forma:

- O servidor aguarda por requisições.
- O cliente envia uma requisição HTTP para o endpoint adequado no servidor (por exemplo, GET /tasks ou POST /tasks).
- O servidor processa a requisição, realiza as operações necessárias (obter ou adicionar tarefas) e retorna uma resposta ao cliente, geralmente em formato JSON.
- O cliente recebe a resposta e pode então processar os dados conforme necessário.

QUAIS SÃO AS VANTAGENS DE USAR UMA API?

01 **Interoperabilidade**: Permitem que diferentes sistemas e aplicativos se comuniquem de forma padronizada, facilitando a integração.

02 **Acesso a funcionalidades**: Desenvolvedores podem usar funcionalidades de um software sem conhecer seus detalhes internos, criando aplicações mais complexas.

03 **Reutilização de código**: APIs permitem que o código seja reutilizado em diferentes contextos, aumentando a eficiência e reduzindo a redundância.

03 **Desenvolvimento rápido**: Ao usar APIs, desenvolvedores podem incorporar funcionalidades existentes, economizando tempo e recursos.

QUAIS SÃO AS VANTAGENS DE USAR UMA API?

05 I**ntegração de serviços**: APIs são usadas para integrar serviços de terceiros, como autenticação via API.

06 **Distribuição de dados**: Facilitam o compartilhamento de dados entre diferentes sistemas, permitindo acesso e atualização de informações.

07 **Atualização independente**: Permitem que partes do software sejam atualizadas separadamente, facilitando a manutenção.

08 **Economia de recursos**: Desenvolvedores podem aproveitar serviços existentes, economizando esforço e recursos.

ESTRUTURA DE MENSAGENS FORMATADAS EM JSON

As APIs web são um tipo específico de API remota que se comunica pela internet usando HTTP/HTTPS. Elas seguem um padrão para requisições e respostas, geralmente formatadas em JSON ou XML.

```
{
"pedidos":[
  {"pedido":"almoço","horário":"meio-dia" },
  {"pedido":"janta","horário":"tarde" },
  {"pedido":"café-da-manhã","horário":"manhã" }]
}

```

PRIMEIRA API

O MÓDULO HTTP

1. Configuração do Projeto
Inicie criando um novo diretório para o seu projeto e inicialize um projeto Node.js utilizando npm ou yarn:

```
mkdir minha-api-node
cd minha-api-node
npm init -y
```

O MÓDULO HTTP

2. No seu editor de código, crie um novo arquivo (por exemplo, server.js) e importe os módulos necessários:

```
1 // Importar módulos necessários
2 const http = require ( 'http' );
3 const url = require ( 'url' );
```

O MÓDULO HTTP

3. Para simular um armazenamento de dados simples, inicialize uma matriz de produtos:

```
1 // Inicializar uma matriz de produtos para simular um armazenamento de dados
2 let products = [
3   { id : 1 , name : 'Product 1' , price : 20.0 },
4   { id : 2 , name : 'Product 2' , price : 30.0 },
5 ];
```

O MÓDULO HTTP

4. Inicie o servidor e escute na porta especificada:

```
1 const PORT = 3000;
2 // Iniciar o servidor
3 server.listen(PORT, () ⇒ {
4   console.log(`Servidor do produto escutando em http://localhost:${PORT}`);
5 });
```

O MÓDULO HTTP

5. Manipule solicitações GET para recuperar informações do produto:

```
// Criar um servidor HTTP
const server = http.createServer((req, res) => {
  const parsedUrl = url.parse(req.url, true); // Analisar a URL

  if (req.method === "GET" && parsedUrl.pathname === "/products") {
    res.writeHead(200, { "Content-Type": "application/json" });
    res.end(JSON.stringify(products));
  } else {
    res.writeHead(404, { "Content-Type": "text/plain" });
    res.end("Rota não encontrada");
  }
});
```

THANK YOU

@wallace027dev