

INSTITUTO FEDERAL

Mato Grosso do Sul

WEB SERVICES

Ermerson Moraes

Prof. EBTT – Informática/Desenvolvimento - Substituto

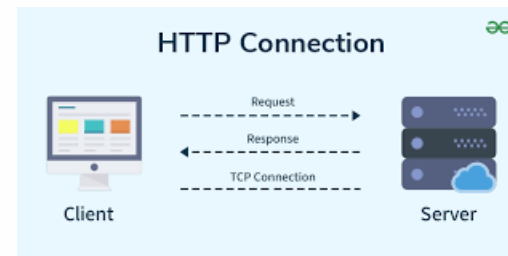
Email: ermerson.moraes@ifms.edu.br

WEB SERVICES

HTTP

O que é HTTP?

- HTTP, ou Hypertext Transfer Protocol, é um protocolo de comunicação para a transferência de informações na World Wide Web (WWW). É a base para a troca de informações entre um cliente (como um navegador) e um servidor web.
- Ele permite a requisição e a resposta de recursos, como imagens, arquivos e páginas web, por meio de mensagens padronizadas.

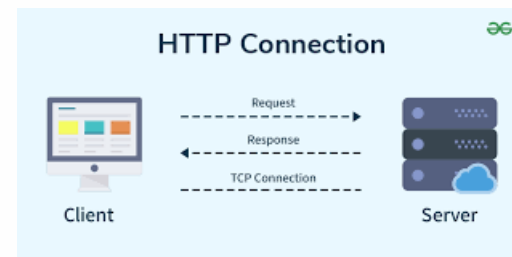


WEB SERVICES

HTTP

O que é HTTP?

- Para Web Services: O HTTP é o alicerce sobre o qual a grande maioria dos web services modernos, especialmente as APIs RESTful, são construídos. A comunicação entre uma aplicação e uma API acontece por meio de mensagens HTTP.



WEB SERVICES

HTTP

Origem e Evolução do HTTP

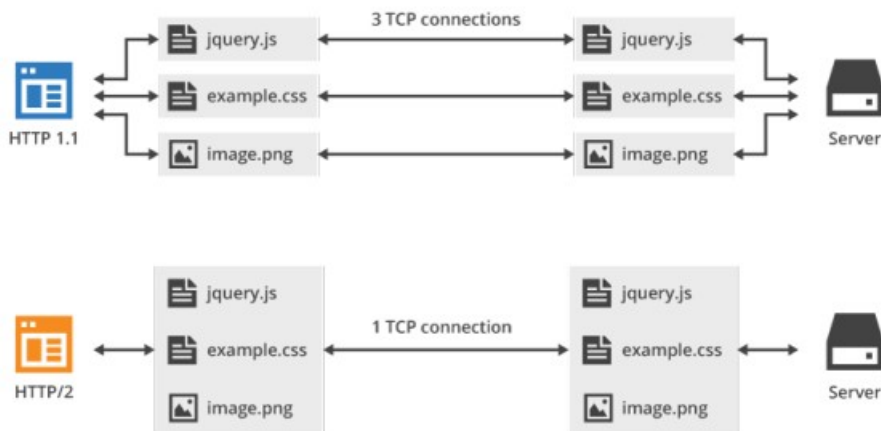
- HTTP/0.9 (Anos 90): Versão inicial criada por Tim Berners-Lee no CERN. Permitiam apenas solicitações GET e a resposta era apenas o conteúdo HTML.
- HTTP/1.1 (1997): Introduziu melhorias cruciais como a persistência de conexão, que permite múltiplas requisições na mesma conexão TCP, reduzindo a latência. Também formalizou os cabeçalhos (headers), tornando o protocolo extensível.

WEB SERVICES

HTTP

Origem e Evolução do HTTP

- HTTP/2 (2015): Focado em performance, trouxe a multiplexação (transmissão simultânea de requisições e respostas), compressão de cabeçalhos e server push, tornando a navegação mais rápida.

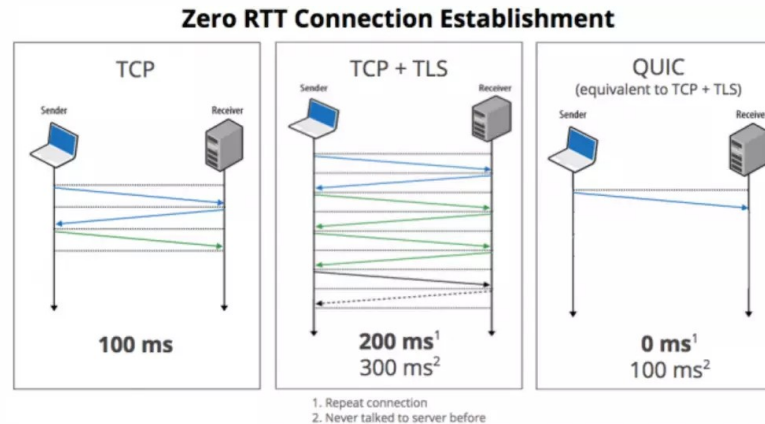


WEB SERVICES

HTTP

Origem e Evolução do HTTP

- HTTP/3 (2018): A versão atual, que utiliza o protocolo QUIC em vez do TCP. O QUIC agiliza o estabelecimento da conexão (reduzindo o "handshake"), melhorando ainda mais a performance, especialmente em redes instáveis.



WEB SERVICES

HTTP

Como o HTTP Funciona: O Modelo Cliente-Servidor

- O HTTP opera em um modelo onde um cliente (navegador, aplicativo mobile, outro serviço) faz solicitações a um servidor para obter ou manipular recursos.



WEB SERVICES

HTTP

Como o HTTP Funciona: O Modelo Cliente-Servidor

- O ciclo é:

1. O cliente abre uma conexão TCP com o servidor, encaminhando uma requisição HTTP.

2. Nessa solicitação, o cliente especifica o método pretendido (por exemplo, GET para obter informações) e o caminho do recurso desejado;

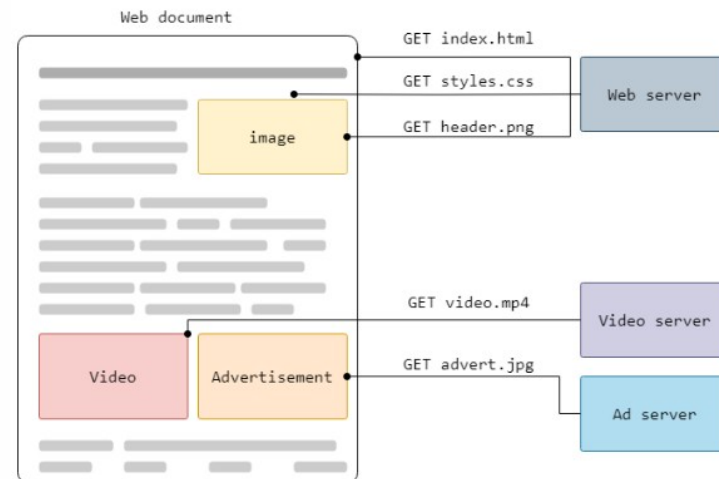
WEB SERVICES

HTTP

Como o HTTP Funciona: O Modelo Cliente-Servidor

- O ciclo é:

3. Ao receber essa requisição, o servidor a processa e responde com uma mensagem HTTP, incluindo o recurso requisitado e informações adicionais no cabeçalho da resposta.



WEB SERVICES

HTTP

Como o HTTP Funciona: O Modelo Cliente-Servidor

- O ciclo é:

E toda essa interação acontece por meio do protocolo TCP (Transmission Control Protocol), que garante uma comunicação confiável e orientada à conexão.

Sem a necessidade de gerir o estado entre as requisições, o protocolo mantém uma certa simplicidade, promovendo também uma escalabilidade maior.

WEB SERVICES

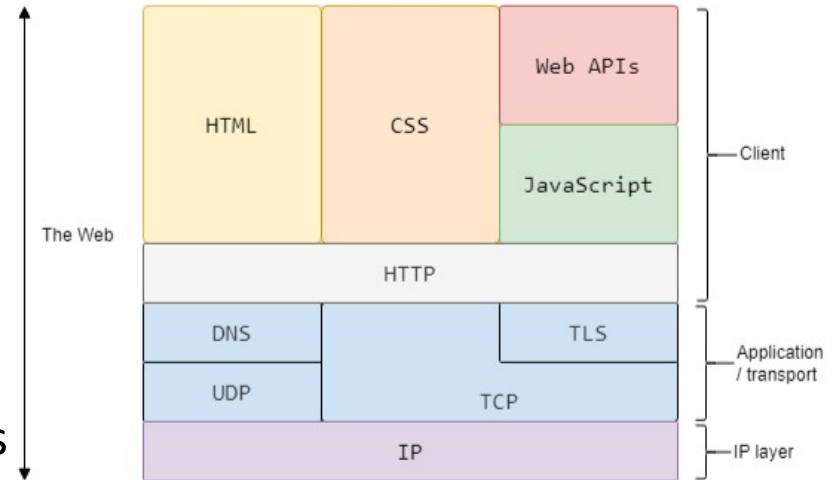
HTTP

Como o HTTP Funciona: O Modelo Cliente-Servidor

- O ciclo é:

Contudo, para abordar a demanda de manter estados em determinadas aplicações web, são comumente empregados mecanismos suplementares, tais como cookies e sessões.

4. A conexão é fechada ou reutilizada para futuras requisições

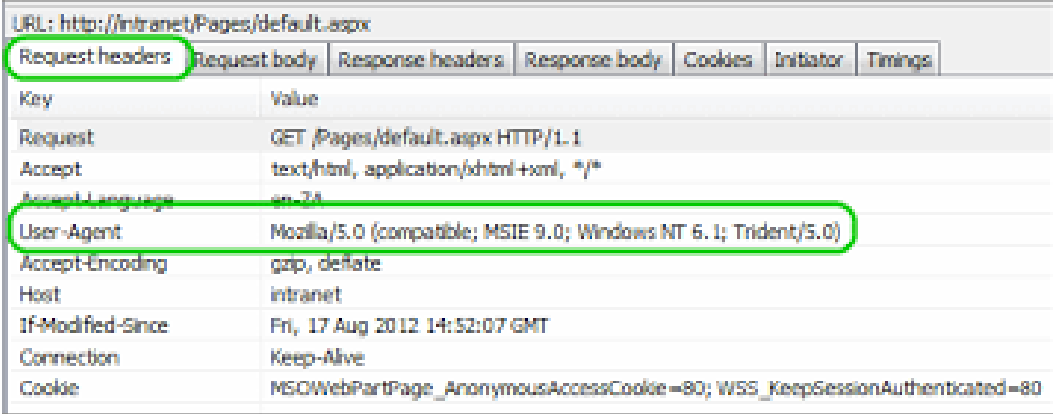


WEB SERVICES

HTTP

Componentes de Sistemas HTTP

- Cliente (Agente-Usuário): Qualquer ferramenta que age em nome do usuário. Embora o mais comum seja um navegador web, no contexto de web services, o cliente é frequentemente outra aplicação, um script de servidor ou um aplicativo móvel consumindo uma API.



URL: http://intranet/Pages/default.aspx

| Key | Value |
|-------------------|--|
| Request | GET /Pages/default.aspx HTTP/1.1 |
| Accept | text/html, application/xhtml+xml, */* |
| Accept-Language | en-74 |
| User-Agent | Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0) |
| Accept-Encoding | gzip, deflate |
| Host | intranet |
| If-Modified-Since | Fri, 17 Aug 2012 14:52:07 GMT |
| Connection | Keep-Alive |
| Cookie | MSOWebPartPage_AnonymousAccessCookie=80; WSS_KeepSessionAuthenticated=80 |

WEB SERVICES

HTTP

Componentes de Sistemas HTTP

- Servidor Web: A máquina que armazena os recursos e recebe as requisições. Pode ser um único servidor ou um conjunto deles com balanceamento de carga para distribuir as requisições.



WEB SERVICES

HTTP

Componentes de Sistemas HTTP

- Proxies: Intermediários que ficam entre o cliente e o servidor. Essas máquinas operam na camada de aplicação. Eles podem ser transparentes ou não. Suas funções são vitais para a arquitetura de sistemas distribuídos e web services:
 - Cacheamento: Armazenar respostas para entregá-las mais rápido.

WEB SERVICES

HTTP

Componentes de Sistemas HTTP

- Balanceamento de Carga: Distribuir o tráfego entre vários servidores.
- Autenticação/Autorização: Atuar como um gateway de API, controlando o acesso.
- Filtragem e Logging: Registrar requisições ou bloquear tráfego malicioso.

WEB SERVICES

HTTP

Aspectos Fundamentais do HTTP

- HTTP é simples

Mesmo com mais complexidade introduzida no HTTP/2.0 por encapsular mensagens HTTP em quadros (frames), o HTTP foi projetado para ser simples e legível às pessoas. As mensagens HTTP podem ser lidas e entendidas por qualquer um, provendo uma maior facilidade para desenvolvimento e testes, e reduzir a complexidade para os estudantes.

WEB SERVICES

HTTP

Aspectos Fundamentais do HTTP

- É extensível: Cabeçalhos (Headers) permitem que novas funcionalidades sejam adicionadas.
- Não tem estado (Stateless): Cada requisição é independente; o servidor não guarda informações sobre requisições anteriores na mesma conexão. Isso é fundamental para a escalabilidade dos web services, pois qualquer servidor pode processar qualquer requisição.

WEB SERVICES

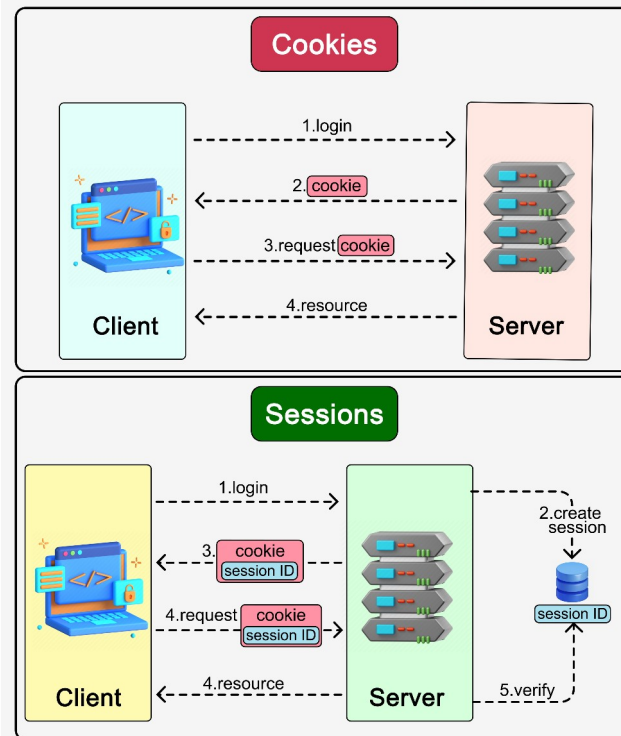
HTTP

Aspectos Fundamentais do HTTP

- Para manter o estado quando necessário (ex: um carrinho de compras), utilizam-se mecanismos como Cookies e Sessões.

Cookies vs Sessions

ByteByteGo



WEB SERVICES

HTTP

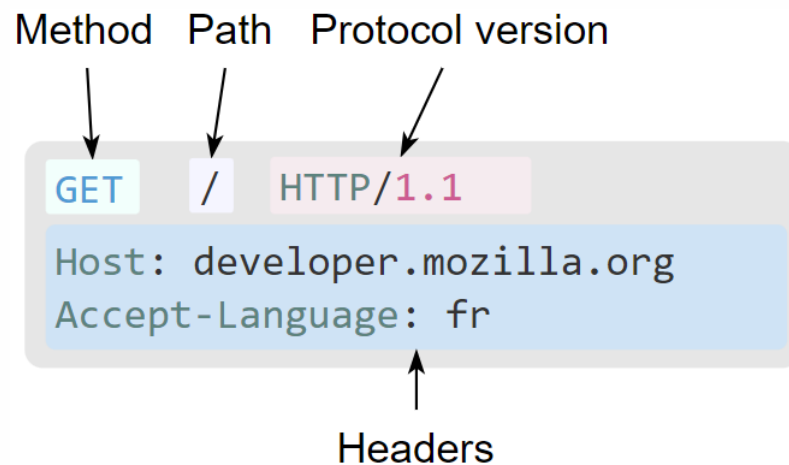
Mensagens HTTP: A Comunicação na Prática

Uma mensagem HTTP é composta por uma linha inicial, cabeçalhos e, opcionalmente, um corpo.

A. Requisições (Requests)

Uma requisição consiste em:

1. Método (Method): O verbo que indica a ação desejada.

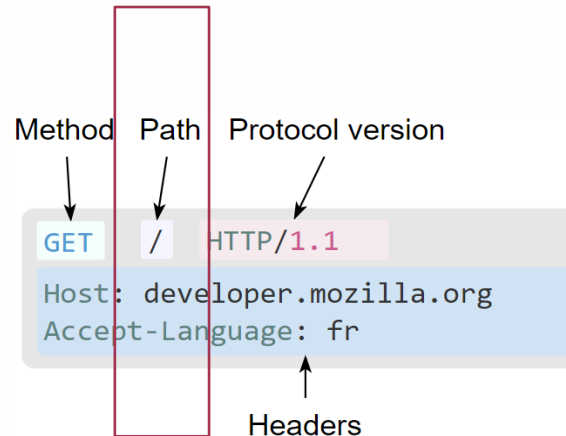


WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

2. Caminho (Path): O recurso que se deseja acessar no servidor. A URL do recurso sem os elementos que são de contexto, o domínio domain ou a porta port TCP (aqui indicada pelo 80 que é ocultado por ser o número da porta padrão)

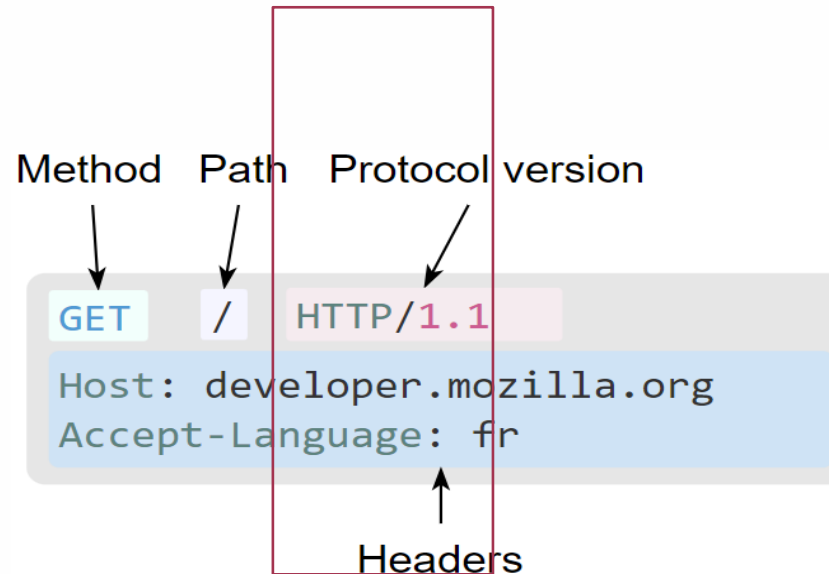


WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

3. Versão do Protocolo: Ex: HTTP/1.1.

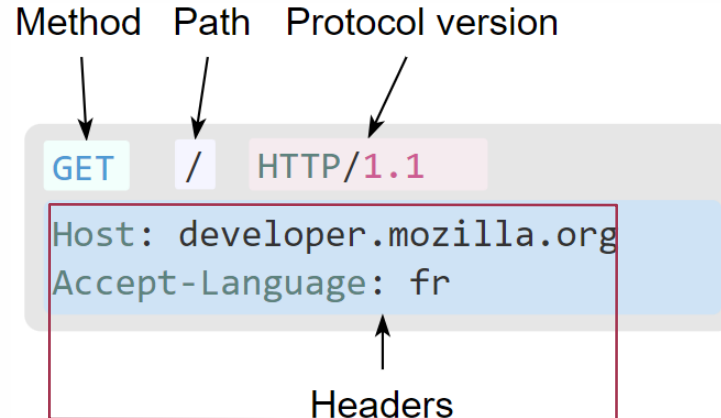


WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

4. Cabeçalhos (Headers): Informações adicionais sobre a requisição.

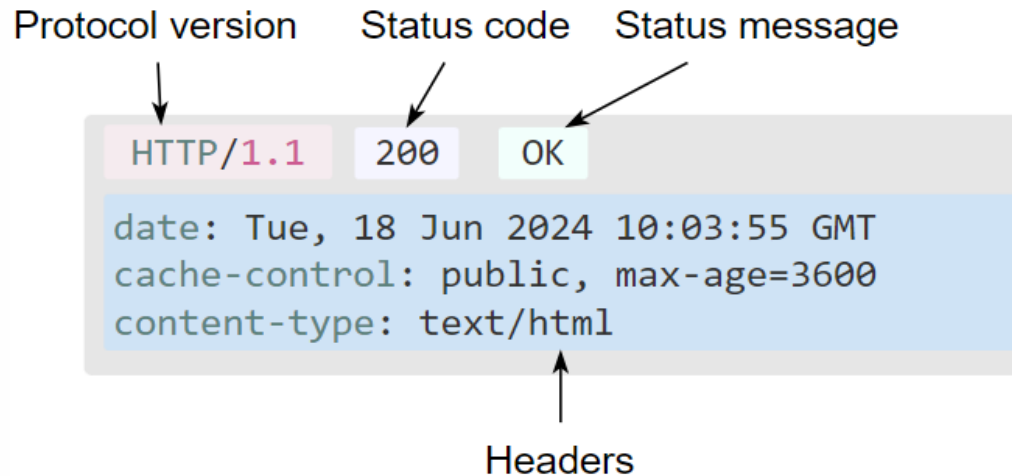


WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

5. Corpo (Body): Dados enviados ao servidor, comum em métodos como POST e PUT.



WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

Conexão com Web Services (APIs RESTful):

Os métodos HTTP são mapeados diretamente para as operações CRUD (Create, Read, Update, Delete):

| Método HTTP | Operação CRUD | Descrição na API |
|-------------|---------------|---|
| GET | Read | Usado para ler/buscar um ou mais recursos. |
| POST | Create | Usado para criar um novo recurso no servidor. |
| PUT | Update | Usado para atualizar um recurso existente por completo. |
| DELETE | Delete | Usado para remover um recurso. |

WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

Conexão com Web Services (APIs RESTful):

Exemplo de Requisição a uma API JSON:

HTTP

POST /api/usuarios **HTTP/1.1**

Host: meudominio.com

Content-Type: application/json

Accept: application/json

```
{  
  "nome": "Ana Silva",  
  "email": "ana.silva@example.com"  
}
```

WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

B. Respostas (Responses)

Uma resposta consiste em:

1. Versão do Protocolo.
2. Código de Status (Status Code):

Um número que indica o resultado da requisição.

Protocol version Status code Status message

HTTP/1.1

200

OK

```
date: Tue, 18 Jun 2024 10:03:55 GMT  
cache-control: public, max-age=3600  
content-type: text/html
```

Headers

WEB SERVICES

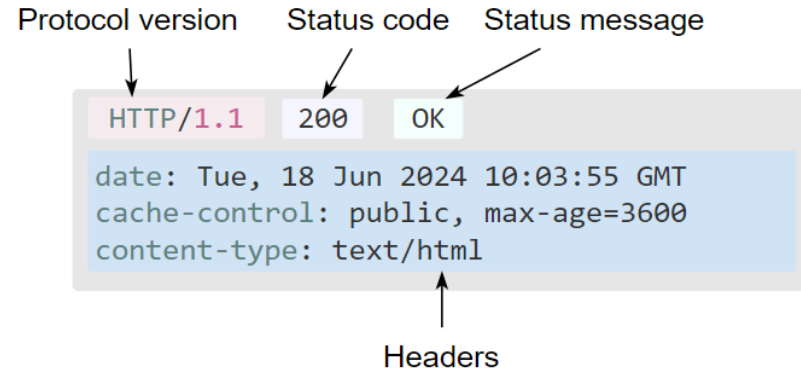
HTTP

Mensagens HTTP: A Comunicação na Prática

B. Respostas (Responses)

Uma resposta consiste em:

3. Mensagem de Status: Uma breve descrição do código (ex: OK, Not Found).
4. Cabeçalhos (Headers): Informações sobre a resposta.
5. Corpo (Body): O conteúdo do recurso solicitado (ex: HTML, JSON, uma imagem).



WEB SERVICES

HTTP

Mensagens HTTP: A Comunicação na Prática

Exemplo de Resposta de uma API JSON (para a requisição acima):

HTTP/1.1 201 Created

Date: Tue, 26 Aug 2025 10:30:00 GMT

Content-Type: application/json

```
{  
  "id": 123,  
  "nome": "Ana Silva",  
  "email": "ana.silva@example.com",  
  "mensagem": "Usuário criado com sucesso."  
}
```

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 1xx (Informativas): Requisição recebida, processo continuando.

100 Continue - Essa resposta provisória indica que o cliente deve continuar a solicitação ou ignorar a resposta se a solicitação já estiver concluída.

101 Switching Protocols - Esse código é enviado em resposta a um cabeçalho de solicitação Upgrade do cliente e indica o protocolo para o qual o servidor está mudando.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 2xx (Sucesso): A requisição foi recebida, entendida e aceita com sucesso.
 - 200 OK: Sucesso padrão para GET.
 - 201 Created: Em APIs: A resposta ideal para um POST que cria um novo recurso.
 - 202 Accepted: Em APIs: Usado para tarefas assíncronas. O servidor aceitou a requisição, mas ainda não a processou.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 3xx (Redirecionamento): Ações adicionais precisam ser tomadas para completar a requisição.
 - 301 Moved Permanently: A URL do recurso mudou permanentemente.

Em APIs: Útil para versionamento ou alteração de endpoints.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 4xx (Erro do Cliente): A requisição contém sintaxe incorreta ou não pode ser atendida.

- 400 Bad Request: Erro genérico do cliente.

Em APIs: Usado para dados inválidos, como um JSON malformatado.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 4xx (Erro do Cliente): A requisição contém sintaxe incorreta ou não pode ser atendida.

- 401 Unauthorized: O cliente precisa se autenticar.

Em APIs: Ocorre quando um token de autenticação está faltando ou é inválido.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 4xx (Erro do Cliente): A requisição contém sintaxe incorreta ou não pode ser atendida.
 - 403 Forbidden: O servidor entendeu a requisição, mas se recusa a autorizá-la.

Em APIs: O cliente está autenticado, mas não tem permissão para acessar o recurso.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 4xx (Erro do Cliente): A requisição contém sintaxe incorreta ou não pode ser atendida.
 - 404 Not Found: O recurso solicitado não foi encontrado.
 - 405 Method Not Allowed: O método HTTP usado não é permitido para aquele recurso.

WEB SERVICES

HTTP

Códigos de Status HTTP (Status Codes)

Os códigos de status são agrupados em cinco classes:

- 5xx (Erro do Servidor): O servidor falhou em atender a uma requisição aparentemente válida.
 - 500 Internal Server Error: Um erro inesperado aconteceu no servidor.
 - 503 Service Unavailable: O servidor está temporariamente indisponível (manutenção ou sobrecarga).