Desenvolvimento de Software para Web

UFC - Universidade Federal do Ceará

André Meireles
andre@crateus.ufc.br

Desenvolvimento de Software para Web

Fundamentos da Web E Servidores HTTP

UFC - Universidade Federal do Ceará

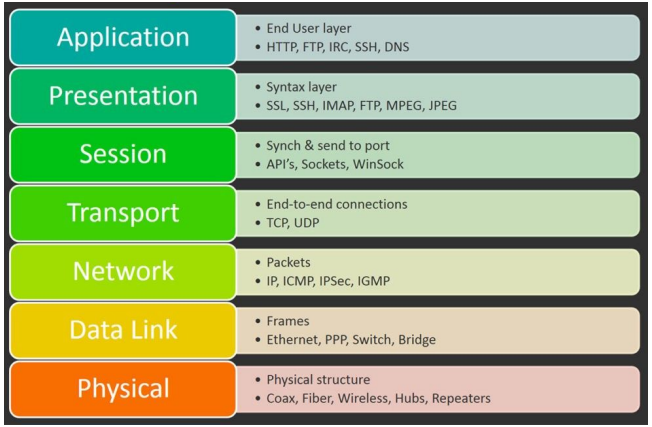
André Meireles
andre@crateus.ufc.br

Arquitetura básica da Web

A Web baseia-se em:

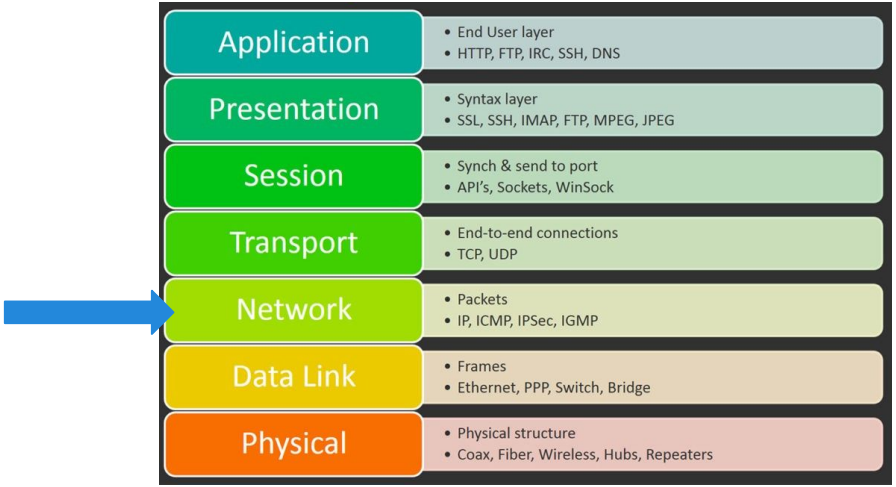
- **Protocolos**: Convenção que controla e possibilita uma conexão, comunicação, transferência de dados entre dois sistemas computacionais

  - **IP** - Internet Protocol
  - **TCP** - Transmission Control Protocol
  - **DNS** - Domain Name System
  - **HTTP** - Hypertext Transfer Protocol



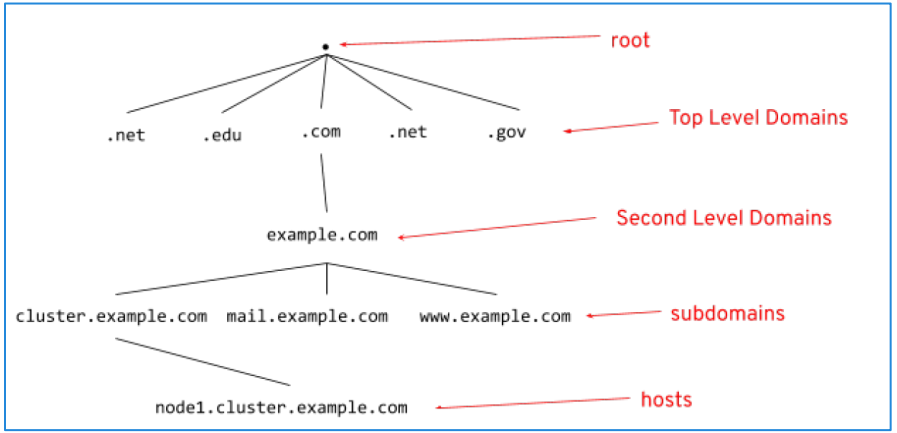
Arquitetura básica da Web

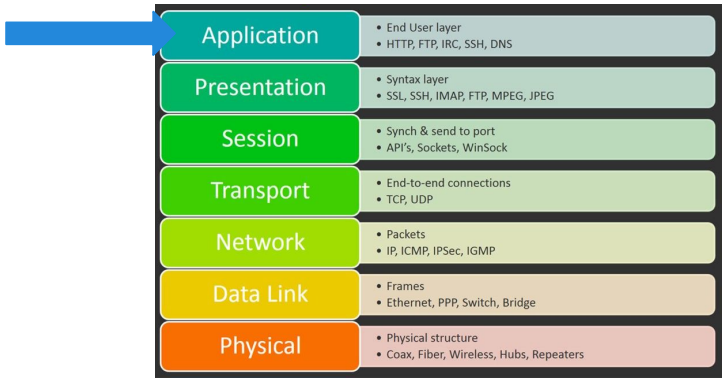
- **IP (Internet Protocol):** permite a localização de um computador na internet, através de um conjunto de dígitos chamado de endereço IP
  - **Formato** do protocolo IPv4:
    - Ex.: 192.168.1.12, 255.255.255.255



Arquitetura básica da Web

- **DNS** - **Domain Name System**: Permite resolver os nomes de domínios da internet, ou seja, identifica os IPs associados a cada nome de domínio
  - **Formato** dos nomes de domínios é organizado de forma hierárquica

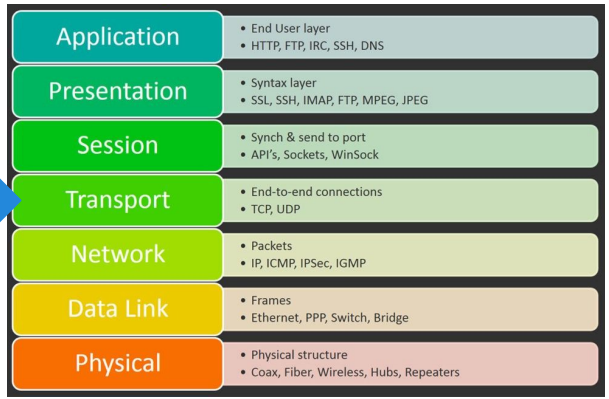




Fonte: Huawei Community > Forums > Routing & Switching > DNS Architecture

Arquitetura básica da Web

- **TCP (Transmission Control Protocol):** provê a transmissão de dados fim a fim de uma origem a um destino e suporta a recuperação de pacotes perdidos
  - **Porta**: representa o ponto de entrada de uma conexão em um sistema
    - Protocolos de camadas superiores (e alguns softwares) costumam estabelecer portas padrão
      - SSH: 22
      - FTP: 21
      - HTTP: 80
      - HTTPS: 443
      - MySQL: 3306
  - **Formato** de um endereço de conexão TCP:
    - [host]:[port]
      - 200.12.25.10:443
      - crateus.ufc.br:80



Arquitetura básica da Web

URL - Uniform Resource Locator

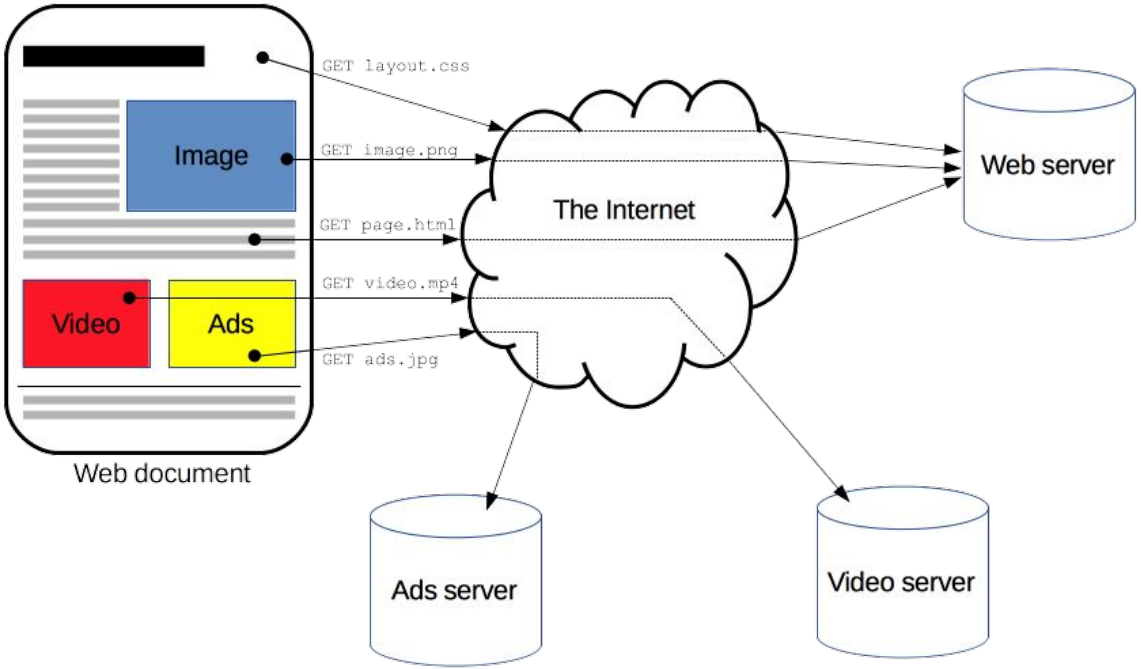
Todas as comunicações na plataforma Web utilizam a sintaxe chamada URL para localizar os recursos que são transferidos.

- Contém duas informações essenciais:
  - **COMO** transferir o objeto
  - **ONDE** encontrá-lo.
- Formato: `protocolo://endereço:porta/recurso`
  - **Exemplos**:
    - http://www.dominio.com.br/texto.html
    - ssh://crateus.ufc.br
    - ftp://192.16810.125

Arquitetura básica da Web

- **HTTP** - Hypertext Transfer Protocol
  - É um protocolo (protocol) que permite a **obtenção de recursos**, como documentos HTML.
  - É o **mecanismo mais utilizado para a troca de dados na Web**
  - Protocolo **cliente-servidor**: as requisições são iniciadas pelo cliente, geralmente um navegador da Web.
  - Um **documento** completo é reconstruído a partir dos diferentes **sub-documentos** obtidos (exemplo: texto, descrição do layout, imagens, vídeos, scripts e muito mais)

HTTP - Hypertext Transfer Protocol



Prática:

- Abra o link
- Acesse as Ferramentas do Desenvolvedor
- Clique na aba Network
- Analise as requisições

HTTP - Hypertext Transfer Protocol

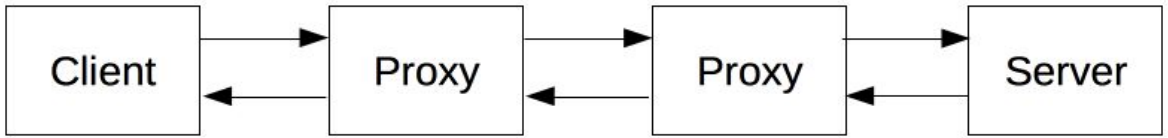
- Clientes e servidores se comunicam trocando **mensagens individuais** (ao contrário de um fluxo de dados)
- **Requests** (requisições) são as mensagens enviadas pelo cliente, geralmente um navegador da Web
- **Responses** (respostas) são as mensagens enviadas pelo servidor como resposta.

Componentes de sistemas baseados em HTTP

Requisições são enviados por uma entidade, o **user-agent** (ou um proxy em nome dele).

Exemplo: um navegador.

Cada **requisição** individual é enviada para um **servidor**, que irá processar e fornecer um resultado, chamado de **resposta**.



Entre a solicitação e a resposta existem várias entidades, designadas coletivamente como **proxies**, que executam operações diferentes e atuam como **gateways** (intermediários) ou **caches**, por exemplo

Componentes de sistemas baseados em HTTP

Cliente: o agente-usuário

- Essa função é predominantemente realizada pelo **navegador Web**
- Nas últimas 2 décadas, o HTTP tem sido utilizado com diversos **outros clientes** (mobile apps, comunicação server-to-server, dispositivos IoT, etc)
- É **sempre** o cliente que **inicia as requisições**, nunca o lado do servidor
- Quando o navegador recebe o documento HTML da página, ele realiza uma análise sintática desse arquivo, buscando requisições adicionais correspondentes a:
  - **scripts** de execução
  - informações de layout (**CSS**)
  - subrecursos contidos na página (geralmente **imagens** e **vídeos**)
- Depois, o navegador **interpreta** esses recursos **para mostrar** ao usuário a página completa.
- **Scripts** executados pelo navegador podem buscar mais recursos em seguida
- A partir de interações do usuário, o navegador pode **modificar a página dinamicamente**

Componentes de sistemas baseados em HTTP

Servidor de páginas Web e conteúdos

- Do outro lado do canal de comunicação está o **servidor** que serve o documento requisitado pelo cliente
- Um servidor se apresenta virtualmente apenas como uma máquina
- Porém, um servidor pode ser **autocontido**, ou uma **coleção de servidores dividindo a carga**, ou **um programa complexo que acessa outros servidores** (como um cache, um servidor de banco de dados, servidores de e-commerce (lojas virtuais), etc.) para gerar o documento solicitado
- Também é possível que **vários servidores** sejam **hospedados numa mesma máquina**
    - Com o HTTP/1.1 e o cabeçalho Host, eles podem até compartilhar o mesmo endereço IP

Componentes de sistemas baseados em HTTP

Proxies

- Entre o cliente e o servidor, vários computadores e máquinas **transmitem as mensagens** HTTP.
- A maioria dessas máquinas operam em alguma das camadas: de **transporte**, de **rede** ou **física**, sendo transparente na camada da aplicação HTTP
- As máquinas que operam na camada de aplicação são normalmente conhecidas como **proxies**
- Podem desempenhar várias **funções**:
    - **cache** (o cache pode ser público ou privado, como o cache dos navegadores)
    - **filtragem** (como um scanner de antivírus, controle de acesso, etc)
    - **balanceamento de carga** (para permitir que vários servidores possam responder a diferentes requisições)
    - **autenticação** (para controlar quem tem acesso aos recursos)
    - **autorização** (para controlar quem tem acesso a determinada informação)
    - **registro de informação** (permite o armazenamento de informações de histórico)

Componentes de sistemas baseados em HTTP

Sessões e Estado

- HTTP não tem estado (é *stateless*), mas tem sessões: não existe uma relação entre duas requisições
- **O problema do carrinho de compras**: como o protocolo HTTP não guarda o estado, é impossível fazer com que um site guarde as informações de um carrinho de compras somente através do HTTP
- **Cookies HTTP** podem ser usados para criar uma sessão: eles são adicionados ao fluxo do HTTP, permitindo que a criação de uma **sessão** para que diferentes requisições compartilhem o mesmo contexto. Estratégias mais comuns:
  - O programa escrito no servidor identifica o usuário e armazena as informações do seu carrinho
  - Um programa escrito na plataforma cliente (como JavaScript) gerencia essas informações através dos cookies e de bancos de dados que os próprios navegadores disponibilizam para as aplicações, para armazenamento temporário dessas informações de carrinho

HTTP - Hypertext Transfer Protocol

Gerenciamento de Conexões

- Uma **conexão é controlada na camada de transporte**, e portanto fundamentalmente fora do controle do HTTP.
- O HTTP não requer que o protocolo de transporte utilizado seja baseado em conexões, só requer que seja **confiável ou não perca pacotes** (sem pelo menos apresentar erros).
- Os protocolos de transporte mais comuns na internet são TCP e UDP. O **TCP é confiável** e o UDP não. Portanto, o HTTP utiliza o padrão TCP, que é baseado em conexão

HTTP - Hypertext Transfer Protocol

Gerenciamento de Conexões

- Evolução dos gerenciamento de conexões:
  - **HTTP/1.0**: uma conexão TCP era aberta para cada par de requisição/resposta trocada, problema:
    - abrir uma conexão requer várias trocas de mensagens: é lento, mas se torna mais eficiente quando mensagens são enviadas em maior número ou maior frequência: "conexões quentes" são mais eficientes que "conexões frias" (que envia poucas mensagens ou com baixa frequência).
  - **HTTP/1.1**: introduziu o conceito de linhas de produção (ou pipelining) e conexões persistentes: as conexões TCPs podem ser parcialmente controladas usando o cabeçalho HTTP Connect
  - **HTTP/2.0**: permite multiplexar várias mensagens através de uma única conexão, ajudando a manter a conexão mais quente, e mais eficiente.

Curiosidade: Google possui um projeto chamado **QUIC**, trata-se de uma proposta de HTTP construído sobre **UDP** para prover um protocolo de transporte mais eficiente.

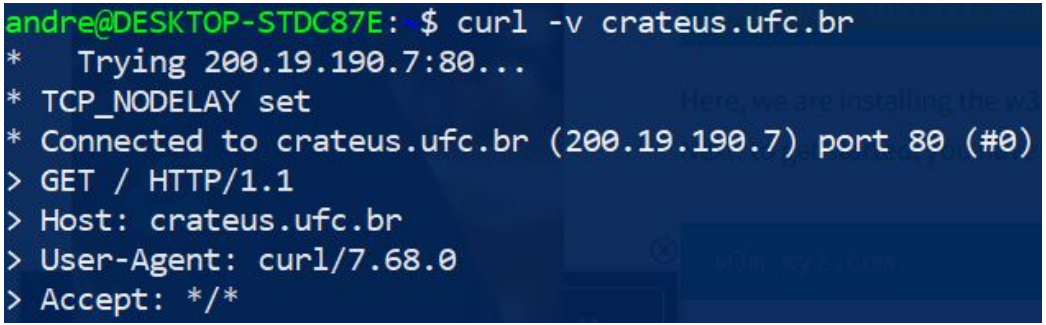
HTTP - Hypertext Transfer Protocol

Fluxo HTTP

1. **Abre uma conexão TCP**: A conexão TCP será usada para enviar uma requisição, ou várias, e receber uma resposta
2. **Envia uma mensagem HTTP**: mensagens HTTP (antes do HTTP/2.0) são legíveis às pessoas. Com o HTTP/2.0, essas mensagens simples são encapsuladas dentro de quadros (frames), tornando-as impossíveis de ler diretamente

Execute um teste com CURL (Linux):

```
# curl -v crateus.ufc.br
```



HTTP - Hypertext Transfer Protocol

Fluxo HTTP

3. **Recebe resposta do servidor**

Teste também outros endereços:

```
# curl -v crateus.ufc.br
# curl -v site.crateus.ufc.br
# curl -v https://crateus.ufc.br
# curl -v https://google.com
```

```
< HTTP/1.1 302 Found
< Date: Wed, 16 Mar 2022 11:51:32 GMT
< Server: Apache/2.2.22 (Debian)
< Location: https://site.crateus.ufc.br//
< Vary: Accept-Encoding
< Content-Length: 3
< Content-Type: text/html
<
* Connection #0 to host crateus.ufc.br left intact
```

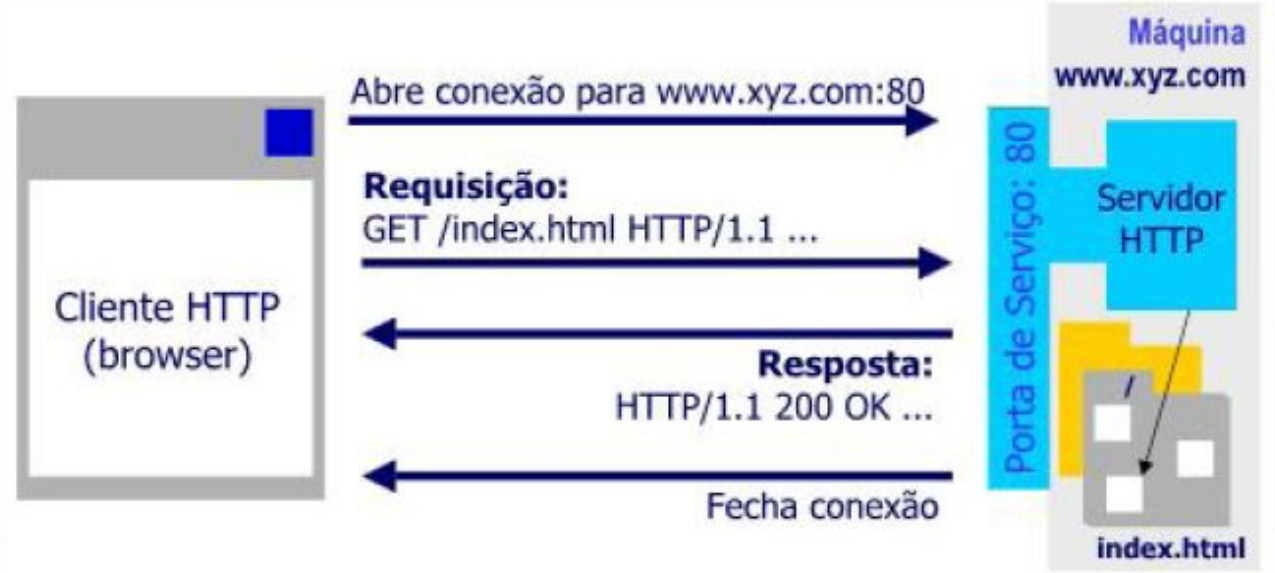
4. **Fecha ou reutiliza a conexão** para requisições futuras.

Fundamentos da Web e Servidores HTTP

Agora, vamos praticar
um pouco!

Vá para a Aula 01P
e execute até o slide 8

HTTP - Hypertext Transfer Protocol

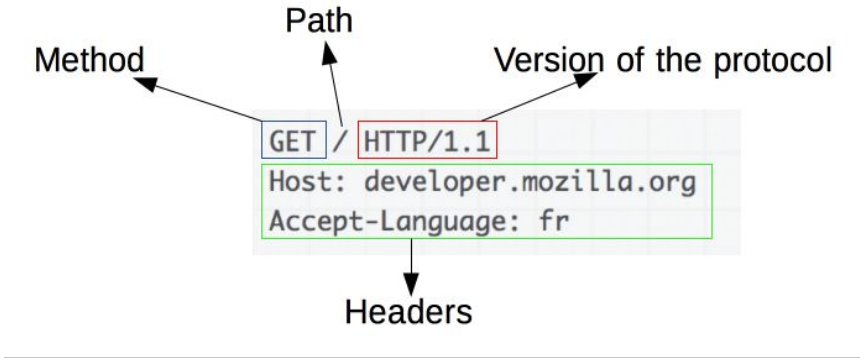


Gerenciamento da conexão TCP no Fluxo HTTP

HTTP - Hypertext Transfer Protocol

Componentes das Requisições

- **Método (Method)**: geralmente é um verbo como GET, POST, DELETE, PUT, que define qual operação o cliente quer fazer
- **Versão** do protocolo HTTP

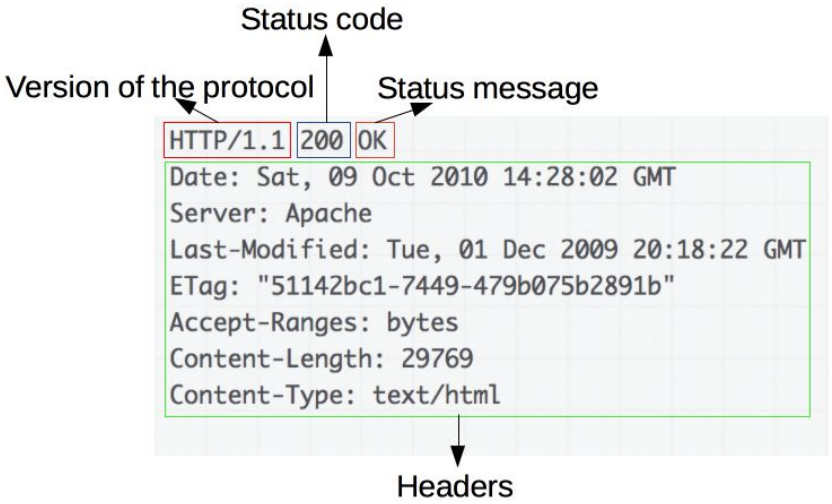


- **Caminho (Path)**: a URL do recurso sem os elementos que são de contexto, como protocolo, domínio ou porta
- **Cabeçalhos (Headers) opcionais** que contém informações adicionais para os servidores.
- Opcionalmente, **Corpo de dados (Body)**, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.

HTTP - Hypertext Transfer Protocol

Componentes das Respostas

- **Versão** do protocolo HTTP
- **Código de status:** indica se a requisição foi bem sucedida, ou não, e por quê
- **Mensagem de status:** pequena descrição informal sobre o código de status
- **Cabeçalhos (Headers) opcionais** que contém informações adicionais para os clientes



- **Corpo de dados (Body)**, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.

HTTP - Hypertext Transfer Protocol

REQUEST METHODS (RFC7231 e RFC5789)

- **GET**: solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.
- **POST**: utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.
- **PUT**: substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.
- **PATCH**: é utilizado para aplicar modificações parciais em um recurso
- **DELETE**: remove um recurso específico.
- **HEAD**: solicita uma resposta de forma idêntica ao método GET, porém sem conter o corpo da resposta.
- ***CONNECT***: estabelece um túnel para o servidor identificado pelo recurso de destino.
- ***OPTIONS***: é usado para descrever as opções de comunicação com o recurso de destino.
- ***TRACE***: executa um teste de chamada *loop-back* junto com o caminho para o recurso de destino.

HTTP - Hypertext Transfer Protocol

RESPONSE STATUS (https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status)

Código	Categoria	Informação
1xx	Informativo	Solicitação foi recebida e o processo continua.
2xx	Sucesso	Ação foi recebida, entendida e aceita com sucesso
3xx	Redirecionamento	Outras ações devem ser tomadas para concluir a solicitação.
4xx	Erro do cliente	Sintaxe incorreta ou não pode ser preenchida
5xx	Erro do servidor	Servidor não conseguiu atender a uma solicitação aparentemente válida.

HTTP - Hypertext Transfer Protocol

RESPONSE STATUS mais comuns

Código	Mensagem	Informação
200	OK	Ação foi recebida, entendida e aceita com sucesso
301	Moved Permanently	Outras ações devem ser tomadas para concluir a solicitação.
400	Bad Request	Sintaxe incorreta ou não pode ser preenchida
401	Unauthorized	Semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada.
403	Forbidden	O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.
404	Not found	O servidor não pode encontrar o recurso solicitado.
500	Internal Server Error	O servidor encontrou uma situação com a qual não sabe lidar.
504	Gateway timeout	O servidor está atuando como um gateway e não obtém uma resposta a tempo

HTTP - Hypertext Transfer Protocol

O que pode ser controlado pelo HTTP?

Cache

- A forma como documentos são armazenados por nós intermediários da rede pode ser controlada pelo HTTP
- O servidor pode instruir proxies e clientes, sobre o que fazer cache e por quanto tempo
- O cliente pode instruir proxies de cache intermediários a ignorar o documento armazenado.

Relaxamento das restrições na origem (CORS)

- Por segurança, os navegadores reforçam estritamente a separação dos sites Web: somente páginas de mesma origem podem acessar todas as informações de uma página Web. O
- Os cabeçalhos HTTP podem relaxar essa separação estrita no lado dos servidores, permitindo que um documento seja composto por várias fontes de informação em outros domínios

HTTP - Hypertext Transfer Protocol

O que pode ser controlado pelo HTTP?

Autenticação

- Algumas páginas podem ser protegidas para que apenas usuários específicos possam acessá-la.
- Cabeçalhos (exemplo: WWW-Authenticate) cookies de sessão são os recursos mais comuns para implementação de autenticação

Proxy e tunelamento

- Servidores e/ou clientes estão frequentemente localizados em intranets e escondem seu verdadeiro endereço IP
- Requisições HTTP recorrem aos proxies para contornar essa barreira na rede
- Proxies também podem ser usados para substituir o endereço do cliente (ex.: Proxy da UFC)
- Proxies podem operar em outros protocolos, como SOCKS, FTP

HTTP - Hypertext Transfer Protocol

O que pode ser controlado pelo HTTP?

Sessões

- Usando os cookies HTTP, permite você vincular requisições com o estado do servidor.
- Isso cria as sessões, apesar do protocolo HTTP básico não manter estado.
- Isso é útil para qualquer site que permita customização das respostas a nível de usuário.