



PMR3412 - Redes Industriais - 2021

Aula 06 - Aplicações TCP/IP: Domain Name System e HTTP/1.1

Prof. Dr. André Kubagawa Sato

Prof. Dr. Marcos de Sales Guerra Tsuzuki

23 de Setembro de 2021

PMR-EPUSP

- ▶ Até agora, vimos as seguintes aplicações TCP/IP:
 - ▶ Telnet: controle remoto, desenvolvido em 1969.
 - ▶ FTP: transferência de arquivo, desenvolvido em 1971.
 - ▶ SMTP: mensagem eletrônica (email), desenvolvido em 1980.
- ▶ Porém, a Internet só se estabeleceu no final da década de 80 / início de 90.

Domain Name System (DNS)

- ▶ As camadas inferiores do modelo TCP/IP permitem acesso a outros hosts em uma rede local ou internet utilizando o endereço IP, que possui a forma 192.168.1.200.
- ▶ Em rede pequena, é possível memorizar ou anotar os endereços de cada máquina. Porém, esta solução não é “escalável”, i.e. logo se torna inviável com o crescimento no número de hosts.
- ▶ Sendo assim, a próxima evolução foi utilizar nomes simbólicos. Assim, o comando:

```
ping 192.168.1.110
```

- ▶ pode ser substituído por:

```
ping myHost
```

- ▶ Os mapeamentos nome-end. IP eram armazenados em um arquivo HOSTS.TXT. Essa solução logo se mostrou pouco prática com o crescimento explosivo do número de hosts.

- ▶ A solução foi adotar o Domain Name System (DNS).
- ▶ O DNS é o mecanismo atual que permite traduzir um nome simbólico para um endereço IP. Assim, ao invés de escrever no seu *browser*:

172.217.162.110

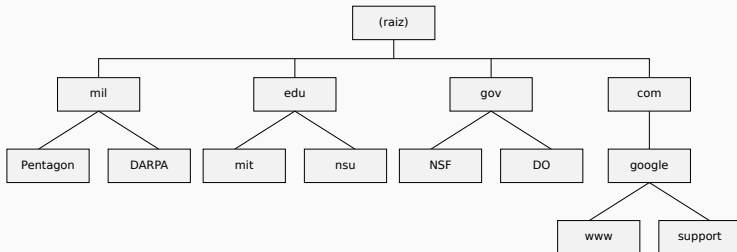
- ▶ Você geralmente escreve:

www.google.com

- ▶ O que permite que o DNS consiga resolver nomes para toda a internet é a possibilidade de um programa executando no host realizar o mapeamento para qualquer outro host na Internet.
- ▶ Assim, não é necessário que nenhum host possua o mapeamento completo de todos os pares de nome-end. IP da Internet.

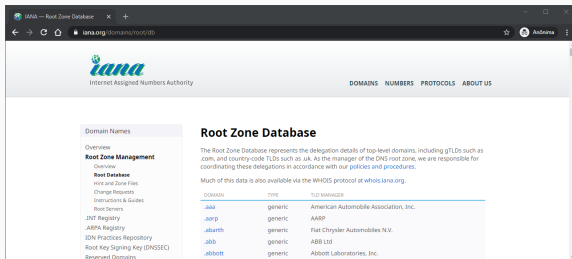
DNS - O Namespace Hierárquico

- ▶ No exemplo `support.google.com`, o host `support` existe dentro do subdomínio `google.com`. Por sua vez, `google.com` é um subdomínio de `com`.
- ▶ No DNS, é comum trabalhar apenas com parte do domínio (p. ex: `google.com`). Para demonstrar que o nome do domínio está completo, deve se adicionar um ponto ao final dele. Por exemplo: `support.google.com.`
- ▶ Este é conhecido como um *fully qualified domain name* (FQDN).



DNS - Os Domínios de nível superior (TLD)

- ▶ Os domínios logo abaixo da raiz (*Top Level Domains* ou TLD) são controlados pela IANA e podem ser conferidos no endereço <https://www.iana.org/domains/root/db>.
- ▶ Existem domínios genéricos como **com** (comercial), **edu** (educacional), **org** (organizações não comerciais), **net** (infraestrutura de rede).
- ▶ Também existem domínios de país com dois caracteres como **br** (Brasil), **uk** (Reino Unido), **de** (Alemanha), entre outros.
- ▶ Em geral, os países possuem domínios de segundo nível que replicam os domínios genéricos. Por exemplo, **.co.uk** e **.ac.uk** correspondem aos domínios genéricos **.com** e **.edu**.



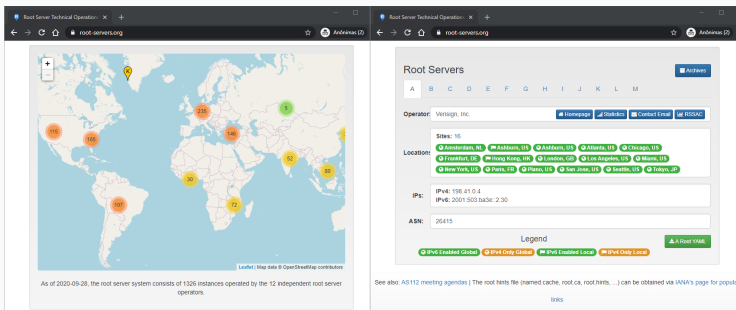
The screenshot shows the IANA Root Zone Database website. The page title is "Root Zone Database" and the URL is "iana.org/domains/root/db". The page features a navigation menu with "DOMAINS", "NUMBERS", "PROTOCOLS", and "ABOUT US". A sidebar on the left lists various domain management options. The main content area includes a description of the database and a table of domain entries.

DOMAIN	TYPE	TLD MANAGER
.aaa	generic	American Automobile Association, Inc.
.aarp	generic	AARP
.aartvliet	generic	Flat Chrysler Automobiles N.V.
.abb	generic	ABB Ltd
.abbott	generic	Abbott Laboratories, Inc.

- ▶ Nomes simbólicos são agrupados em zonas de autoridade (ou simplesmente zonas).
- ▶ Em cada zona, um ou mais hosts são responsáveis por manter o banco de dados e realizar a função de servidores DNS.
- ▶ Cada zona contém uma subárvore da árvore hierárquica e os nomes de uma zona são administrados independentemente de nomes em outra zona.
- ▶ Uma exceção é feita para os servidores raiz, cuja autoridade é distribuída para um conjunto de servidores.

DNS - Os servidores DNS raiz

- ▶ Os servidores raiz são operados por 12 organizações, que possuem diversos servidores em cada continente.
- ▶ As organizações e os endereços IP podem ser consultados em <https://root-servers.org/>.
- ▶ Cada cliente tem programado os endereços estaticamente, uma vez que não podem ser obtidos de outro modo.



The image shows two browser windows from the website root-servers.org. The left window displays a world map with 12 numbered locations representing root servers. The right window shows the 'Root Servers' page for the operator Verisign, Inc., listing 15 sites and their IP addresses.

As of 2020-09-28, the root server system consists of 1326 instances operated by the 12 independent root server operators.

Root Servers

Operator: Verisign, Inc. [Homepage](#) [Feedback](#) [Contact Email](#) [IISD/CIC](#)

Sites: 15

Location: [Amsterdam, NL](#) [Ashburn, US](#) [Ashburn, US](#) [Atlanta, US](#) [Chicago, US](#) [Frankfurt, DE](#) [Hong Kong, HK](#) [London, GB](#) [Los Angeles, US](#) [Miami, US](#) [New York, US](#) [Paris, FR](#) [Palo Alto, US](#) [San Jose, US](#) [Seattle, US](#) [Tokyo, JP](#)

IPs: IPv4: 198.41.0.4
IPv6: 2001:503:ba3e::2:30

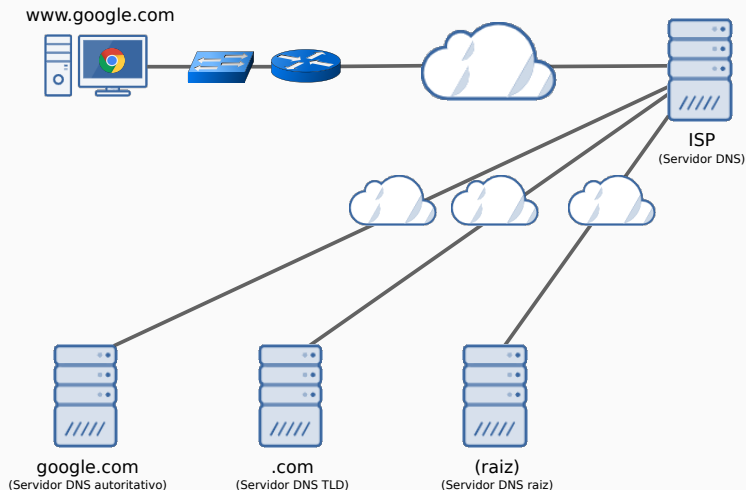
ASN: 26415

Legend: [IPv6 Enabled Global](#) [IPv6 Only Global](#) [IPv6 Enabled Local](#) [IPv6 Only Local](#) [A Root V4v6](#)

See also: [AS112 meeting agendas](#) | The root hints file (named cache, root.ca, root.hints, ...) can be obtained via IANA's page for popular links

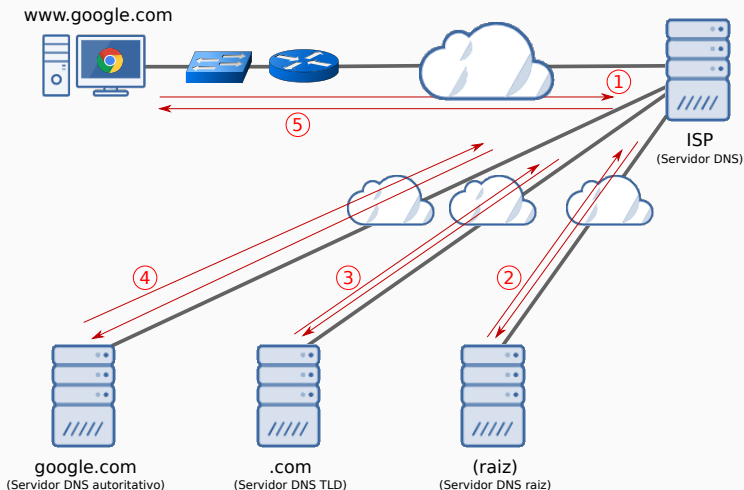
DNS - Exemplo de resolução de nome

- ▶ Exemplo: resolver o nome `www.google.com`.



DNS - Exemplo de resolução de nome

- ▶ Exemplo: resolver o nome `www.google.com`.



DNS - Exemplo de resolução de nome

- Realizando o trace com a ferramenta dig (<https://www.isc.org/download/>)

```

C:\Users\aksato> dig +trace www.google.com
;<<>> Dig 9.14.2 <<>> +trace www.google.com
;; global options: +cmd
256234 IN NS f.root-servers.net.
256234 IN NS g.root-servers.net.
256234 IN NS h.root-servers.net.
256234 IN NS i.root-servers.net.
256234 IN NS a.root-servers.net.
256234 IN NS j.root-servers.net.
256234 IN NS k.root-servers.net.
256234 IN NS l.root-servers.net.
256234 IN NS m.root-servers.net.
256234 IN NS b.root-servers.net.
256234 IN NS c.root-servers.net.
256234 IN NS d.root-servers.net.
256234 IN NS e.root-servers.net.
;; Received 811 bytes from 192.168.1.1453(192.168.1.1) in 3 ms

com. 172800 IN NS l.gtld-servers.net.
com. 172800 IN NS b.gtld-servers.net.
com. 172800 IN NS c.gtld-servers.net.
com. 172800 IN NS d.gtld-servers.net.
com. 172800 IN NS e.gtld-servers.net.
com. 172800 IN NS f.gtld-servers.net.
com. 172800 IN NS g.gtld-servers.net.
com. 172800 IN NS h.gtld-servers.net.
com. 172800 IN NS i.gtld-servers.net.
com. 172800 IN NS j.gtld-servers.net.
com. 172800 IN NS k.gtld-servers.net.
com. 172800 IN NS l.gtld-servers.net.
com. 86400 IN DS 38909 8 2 E2D3C916F6DEEAC
73294E8268F85885844A833FC5495988F4A9184CF C41A5766
80 20200928040000 46594 . Clearl8pWx1JMoECTyV61hXqb88qnUj8PiSkedUtp76N
3Q0TSHo6R orjIEGt+xGvYaJH611ld7p9VjJGc/29h7Pmbl6ldjsYfPc17iXGnj3 n2vMH
ThVpPVgKlUpiCQYSHuXk3Q8G/hB72ASgBmqjnoxUAc0e8B4 kyioWytWtIReEhndNBQcK
Ru/azA4NBspdy9yVDFpY8B0MwQzCcp hMB1q83h1c18ymqfJ5au/fjmsUCpof7c8en
0Y=118DfByFrbXNOM0 jx1kn7J3hqgwV/FngIHfHmqYFezkuwOF+HeFE6U0Vj3MB8PrYr/
w8 ucy7TA==
;; Received 1174 bytes from 192.5.5.241#53(f.root-servers.net) in 11 ms

```

```

com. 172800 IN NS f.gtld-servers.net.
com. 172800 IN NS g.gtld-servers.net.
com. 172800 IN NS a.gtld-servers.net.
com. 172800 IN NS h.gtld-servers.net.
com. 172800 IN NS i.gtld-servers.net.
com. 172800 IN NS j.gtld-servers.net.
com. 172800 IN NS k.gtld-servers.net.
com. 172800 IN NS a.gtld-servers.net.
com. 86400 IN DS 38909 8 2 E2D3C916F6DEEAC
73294E8268F85885844A833FC5495988F4A9184CF C41A5766
80 20200928040000 46594 . Clearl8pWx1JMoECTyV61hXqb88qnUj8PiSkedUtp76N
3Q0TSHo6R orjIEGt+xGvYaJH611ld7p9VjJGc/29h7Pmbl6ldjsYfPc17iXGnj3 n2vMH
ThVpPVgKlUpiCQYSHuXk3Q8G/hB72ASgBmqjnoxUAc0e8B4 kyioWytWtIReEhndNBQcK
Ru/azA4NBspdy9yVDFpY8B0MwQzCcp hMB1q83h1c18ymqfJ5au/fjmsUCpof7c8en
0Y=118DfByFrbXNOM0 jx1kn7J3hqgwV/FngIHfHmqYFezkuwOF+HeFE6U0Vj3MB8PrYr/
w8 ucy7TA==
;; Received 1174 bytes from 192.5.5.241#53(f.root-servers.net) in 11 ms

google.com. 172800 IN NS ns2.google.com.
google.com. 172800 IN NS ns1.google.com.
google.com. 172800 IN NS ns3.google.com.
google.com. 172800 IN NS ns4.google.com.
com. 172800 IN NS ns4.google.com.
com. 172800 IN NS ns1.google.com.
com. 172800 IN NS ns4.google.com.
com. 172800 IN NS ns1.google.com.
com. 86400 IN NSec3 1 1 0 - CW0QIGiN3hl
ARRC9DSN6QQR081HM9A NS SOA ARRSIG DNSKEY NSEC3PARAM
CW0PJMGB74LJREF7FN843BQVT18B5N.com. 86400 IN ARRSIG NSec3 8 2 86400 2020
1004#44123 20200927833123 24966.com. #MfJ0mWhY6r9ePa3ZB43j8Qj/8UB+hH0a
QcTms/Dk04vx3RgNqplc1qCNOd9v52iZ+hFOXmWpX1PZiZjMMWv+12VfVJHRNVLZEhnsD
6J+h IwRaFqx798V8L5SR02RL+7zqZRenAEdfIno24W9Dq6LBB51+tc50 dymKNGwudjJN
Z8k/o/1c7nnVbJmY9YhXi04BL1575CpeJuw==
S84BDVNH8AGDS17F5J003NRHUBG7Q.com. 86400 IN NSec3 1 1 0 - S84CDV5VPRE
ADP6KHPADMH6108H NS DS ARRSIG
S84BDVNH8AGDS17F5J003NRHUBG7Q.com. 86400 IN ARRSIG NSec3 8 2 86400 2020
1004#43616 20200928032616 24966.com. qIma1ySniXmFqRcX/ZhXtQeRkUtnMEHR
JrsPYRrARhmq1q8GheH 59EUoYqV4/Emg/blu+Qw81YkRh19iLLf2fyfaXmJQ100y2uNm
zP7Q aMK+40248PFXlcv1PTkKw31BusE2EG6hP213lR9SvQd0aAwmotstez LrxZn/F9td
NDvsh/KMx1Y+zYKZme1lR2EXKRxZyG6G==
;; Received 848 bytes from 192.31.80.38#53(d.gtld-servers.net) in 208 ms

www.google.com. 380 IN A 172.217.28.228
;; Received 89 bytes from 216.239.36.10#53(ns3.google.com) in 136 ms

```

- ▶ O DNS segue o modelo cliente servidor: o cliente é o DNS resolver e servidor é o DNS server.

Domain Name Resolver

- ▶ As queries do resolver podem ser de dois tipos: iterativas ou recursivas.
- ▶ Quando um servidor recebe uma requisição que não consegue responder completamente:
 - ▶ query recursiva: servidor gera novas queries para determinar a resposta e encaminhar para o resolver.
 - ▶ query iterativa: servidor retorna as informações incompletas e lista de servidores adicionais para que o usuário possa concluir a query
- ▶ As respostas podem ser de dois tipos: autoritativas (quando o servidor possui autoridade) ou não autoritativas.

Domain Name Server

- ▶ Possuem autoridade sobre zero ou mais zonas
- ▶ Podem ser de três tipos: primário (possui autoridade e armazena informação em disco), secundário (possui autoridade, mas obtém informação do servidor primário) ou caching-only (não possui autoridade).

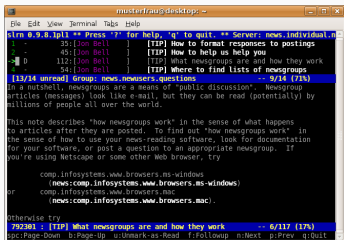
- ▶ Até agora, assumimos que o banco de dados do servidor DNS armazena apenas o mapeamento entre nomes e endereços.
- ▶ Na realidade, o banco de dados distribuído do DNS é composto de entradas de *resource records*, que mapeiam o nome do domínio para um objeto de rede.
- ▶ Uma zona é um agrupamento de *resource records*, iniciando pela entrada do *Start of Authority (SOA)*, contendo o nome do domínio.
- ▶ Abaixo são exibidos alguns tipos de *resource records*:

Tipo	Significado	RFC
A	Um endereço do host	1035
NS	Um nome de servidor autoritativo	1035
SOA	Marca o início da zona de autoridade	1035
MX	Mail-Exchange	1035
AAAA	Um endereço do host em IPv6	3596

Hypertext Transfer Protocol (HTTP)

HTTP - Antes do HTTP

- ▶ Antes do HTTP, era pré Internet, a comunicação ocorria em uma arquitetura *dial-up* (discado).
- ▶ Exemplos: Usenet e *bulletin board system* (BBS), que funcionavam como um fórum Web.
- ▶ Principal diferença entre eles: as BBSs tinham um servidor central e o Usenet era distribuído.



```
musterfraugdesktop: ~
File Edit View Terminal Tabs Help
nlm 0.9.0.ip11 ** Press '?' for help, 'q' to quit. ** Server: news.individual.n
1 - 35:Jon Bell [TIP] How to format responses to postings
2 - 45:Jon Bell [TIP] How to help us help you
3 - 112:Jon Bell [TIP] What newsgroups are and how they work
4 - 34:Jon Bell [TIP] Where to find lists of newsgroups
[13/14 unread] Group: news.newusers.questions -- 9/14 (71%)
In a nutshell newsgroups are a means of "public discussion". Newsgroup
articles (messages) look like e-mail, but they can be read (potentially) by
millions of people all over the world.

This note describes "how newsgroups work" in the sense of what happens
to articles after they are posted. To find out "how newsgroups work" in
the sense of how to use your news-reading software, look for documentation
for your software, or post a question to an appropriate newsgroup. If
you're using Netscape or some other Web browser, try

    comp.infosystems.www.browsers.ms-windows
(news:comp.infosystems.www.browsers.ms-windows)
or
    comp.infosystems.www.browsers.mac
(news:comp.infosystems.www.browsers.mac).

Otherwise try
792381: [TIP] What newsgroups are and how they work -- 6/117 (17%)
bpc:Page-Down b:Page-Up u:Unmark-as-Read f:Followup n:Next p:prev q:Quit
```

Usenet



```
Safra, (255 SYSOP) (2) 249 mins, 0
WORLDWIDE
Read Messages (F) List Files (S) Stats
Enter Message (FR) Reverse List (W) User Info
Page Sysop (N) New Files (G) Goodbye
CF Conference Flags (Z) Search (M) Ansi On/Off
ZOOM Gather Mail (R) Adjust File Flags (X) Expert Mode
(S) Comment to Sysop (D) Download (RZ) Quick Upload
Join Conference (U) Upload (V) View a File
User: Safra from eXTRADITION! at 13:21:51
Extradition [2:AmiExpress] Time: 249 mins. left : █
```

BBS

- ▶ HTTP/0.9: Primeira versão do HTTP, extremamente simples. Cada requisição consistia em uma linha, que utilizava o método GET para especificar o arquivo HTML a ser obtido:

```
GET /mypage.html
```

cuja resposta era o próprio arquivo HTML.

- ▶ HTML/1.0: primeira versão especificada (RFC 1945), introduziu o conceito de MIME types para tratar diferentes tipos de recursos. Além disso, adicionou cabeçalhos e códigos de resposta:

```
GET /mypage.html HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK
Date: Tue, 15 Nov 1994 08:12:31 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/html
<HTML>
A page with an image
  <IMG SRC="/myimage.gif">
</HTML>
```

- ▶ HTTP/1.1: O protocolo HTML/1.0 demonstrava dois problemas principais: falta de uma padronização e impossibilidade de reutilizar a conexão para obter múltiplos arquivos. O protocolo HTTP/1.1, que foi a primeira versão padronizada (RFC 2616 em 1997), solucionou estes problemas, além de introduzir controle de cache e negociação de conteúdo.
- ▶ HTTP/2: oficialmente padronizado em 2015, visou melhor desempenho para o protocolo HTTP, cujo uso se tornou muito mais complexo. A principal diferença em relação à versão 1.1 é a transmissão em binário, que pode ser multiplexada, gerando requisições em paralelo, que podem ser processadas na mesma conexão.
- ▶ HTTP/3: ainda não lançada, atualmente está em versão *draft*. A principal proposta é a substituição do TCP pelo protocolo QUIC, desenvolvido pela Google, que é adaptado para multiplexação das requisições.

- ▶ O HTTP foi desenvolvido para possibilitar a transferência de arquivos HTML, que permite a criação de arquivos hipertexto com imagens, áudio e vídeo.
- ▶ No entanto, acabou se tornando o protocolo fundamental para troca de recursos na Internet.
- ▶ Assim como outros protocolos estudados, adota o modelo cliente-servidor com requisições e respostas. Utiliza o protocolo TCP, com a porta padrão 80.
- ▶ A aplicação cliente é geralmente um *Web browser*, que reconstrói o documento completo, juntando as partes, como arquivos de imagens, scripts e estilos.
- ▶ O protocolo HTTP é considerado *stateless*, uma vez que não mantém informações entre requisições/conexões. No entanto, é possível estabelecer sessões utilizando *cookies*.
- ▶ Muitas vezes a conexão entre o cliente e o servidor não é direto. Existem intermediários, chamados *proxies*, que podem ser transparentes ou não e possuem função importante de *caching*.

HTTP - Fluxo de uma Sessão HTTP

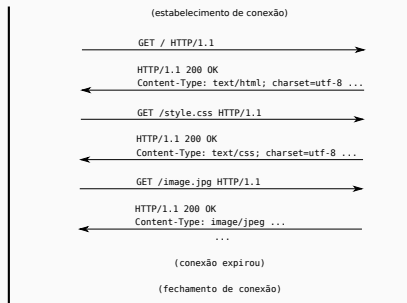
1. O cliente inicia a conexão e envia a requisição.
2. O servidor então processa a requisição e envia a resposta para o cliente.
3. O processo se repete até a conexão expirar. Deste modo, é possível obter vários arquivos em uma mesma requisição (implementada na versão HTTP/1.1).



cliente HTTP
(browser)



servidor HTTP
(Web Server)



- ▶ Cada requisição deve incluir um método para ser aplicado ao recurso.
- ▶ Os métodos podem ser
 - ▶ seguros, quando não geram efeitos colaterais, ou seja, quando apenas recuperam o conteúdo;
 - ▶ ou idempotentes, quando ações repetidas têm o mesmo resultado que uma única ação.
- ▶ Os métodos definidos na especificação são:

Método	Seguro	Idem.	Descrição
GET	X	X	Recupera o recurso.
POST			Aceita a entidade como subordinado ao recurso (definido pelo servidor).
PUT		X	Similar ao PUT, mas identifica o recurso na entidade.
DELETE		X	Apaga o recurso.
OPTIONS			Recupera informações sobre as opções de comunicação disponível.
HEAD	X	X	Similar ao GET, mas não recupera o corpo da resposta.
TRACE			Exibe como a mensagem foi recuperado (para testes).
CONNECT			Conecta a um proxy para abrir um tunel.

► Principais códigos de resposta:

Informacional	Sucesso	Redirecionamento
100 Continue 101 Switching Protocols	200 OK 201 Created 202 Accepted 203 Non-Authoritative Information 204 No Content 205 Reset Content 206 Partial Content	300 Multiple Choices 301 Moved Permanently 302 Moved Temporarily 303 See Other 304 Not Modified 305 Use Proxy
Erro Cliente		Erro Servidor
400 Bad Request 401 Unauthorized 402 Payment Required 403 Forbidden 404 Not Found 405 Method Not Allowed 406 Not Acceptable 407 Proxy Authentication Required	408 Request Timeout 409 Conflict 410 Gone 411 Length Required 412 Precondition Failed 413 Request Entity Too Large 414 Request-URI Too Long 415 Unsupported Media Type	500 Internal Server Error 501 Not Implemented 502 Bad Gateway 503 Service Unavailable 504 Gateway Timeout 505 HTTP Version Not Supported

- ▶ Do ponto de vista do HTTP, um URI (Uniform Resource Identifiers) é uma *string* formatada que identifica um recurso.
- ▶ O esquema `http` é utilizado para localizar recursos de rede no protocolo HTTP, gerando URLs (Uniform Resource Locators) do tipo:

`http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#Somewhere`

- ▶ que pode ser dividido em:
 - ▶ *Scheme*: `http`
 - ▶ *Authority*: `www.example.com`
 - ▶ *Port*: `80`
 - ▶ *Path*: `path/to/myfile.html`
 - ▶ *Query*: `key1=value1&key2=value2`
 - ▶ *Fragment*: `Somewhere`

HTTP - Formato da Mensagem

- ▶ Exemplo de formato de mensagem de requisição:

```
GET / HTTP/1.1      (start line)
Host: localhost:8000 (Message Headers)
```

- ▶ Exemplo de formato de mensagem de resposta:

```
HTTP/1.1 200 OK ← start line
Date: Mon, 19 Oct 2020 17:23:33 GMT
Server: SimpleHTTP/0.6 Python/3.8.3
Last-Modified: Mon, 19 Oct 2020 17:16:40 GMT
Content-Length: 330
Content-Type: text/html
← Message Headers
linha vazia
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
  <title>Hello World Page</title>
</head>
<body>
  <h1>Hello World</h1>
</body>
</html>
Body
```


Referências

- ▶ Para o curso: livro da IBM “TCP/IP Tutorial and technical overview” (disponível em <https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf>).
- ▶ Para esta aula: seções 12.1, 12.2, 16.3 e RFC 2616 (HTTP/1.1).
- ▶ MDN Web Docs sobre HTTP:
<https://developer.mozilla.org/en-US/docs/Web/HTTP>

The End!