

# Protocolo HTTP

6 de agosto de 2019

## 1 Introdução

- HTTP é o protocolo de rede da Web.
- É simples e poderoso.
- Versões: HTTP/1.0 e HTTP/1.1
- HTTP: *Hipertext Transfer Protocol*.
- É o protocolo utilizado para entregar todos os arquivos e outros tipos de dados na *World Wide Web*.
- Pilha de protocolos: HTTP / TCP / IP. Em geral, utiliza soquetes TCP/IP (IP:porta).
- Um cliente (*browser*) envia uma requisição para um servidor, que envia a resposta de volta ao cliente. Porta padrão: 80.
- HTTP é utilizado para transferir *recursos*: blocos de informação que podem ser identificados por uma URL. Em geral, arquivos ou saída de processamento de *scripts*.

## 2 Transações HTTP

- HTTP usa o modelo cliente/servidor:
  - O cliente abre a conexão.
  - O cliente envia ao servidor uma mensagem HTTP.
  - O servidor retorna uma mensagem de resposta contendo o recurso solicitado.
  - O servidor fecha a conexão.
- HTTP é um protocolo *stateless*: nenhuma informação de conexão é mantida entre as transações.

## 2.1 Formato

- O formato de uma mensagem HTTP é:
  - Uma linha inicial.
  - Zero ou mais linhas de cabeçalho.
  - Uma linha em branco (CR LF).
  - Corpo (opcional) da mensagem.

- Exemplo:

<linha inicial> (diversos formatos)

Header1: valor1

Header2: valor2

<linha em branco>

<mensagem opcional>

- As linhas inicial e de cabeçalhos devem ser encerradas com CR LF (ASCII 13 e 10).

## 2.2 Linha inicial de uma requisição

- Formato:

<método> <path local> <versão>

- Exemplo:

GET /path/to/file/index.html HTTP/1.0

- Métodos: GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS.

- Path local: parte da URL depois do nome do *host*.

- Versão: HTTP/*x.x*.

## 2.3 Linha inicial de uma resposta

- Formato:

<versão> <*status code*> <*reason phrase*>

- Exemplos:

HTTP/1.0 200 OK

HTTP/1.0 404 Not Found

- Versão: formato idêntico ao da requisição.

- *Status code*: resultado legível por máquina.

- *Reason phrase*: resultado legível por humanos.

- Faixas de *status code*:

- 1xx: mensagens informativas.
- 2xx: sucesso.
- 3xx: redireção.
- 4xx: erro do cliente.
- 5xx: erro do servidor.

- Exemplos:

- 200 Ok
- 403 Access Denied
- 404 Not Found
- 500 Server Error

## 2.4 Cabeçalhos

- Linhas de cabeçalho fornecem informações sobre a requisição/resposta ou sobre o objeto enviado no corpo.
- Formato: uma linha por cabeçalho, no formato abaixo, terminada com CR LF:  
Header: valor
- Na versão 1.0, existem 16 cabeçalhos, todos opcionais.
- Na versão 1.1, existem 46 cabeçalhos, um deles (*Host*) obrigatório.
- Exemplo:  
From: email@teste.com  
User Agent: Mozilla/3.0  
Last-Modified: Fri 31 Dec 1999 23:59:59 GTM

## 2.5 Corpo da mensagem

- O corpo opcional da mensagem é usado para enviar informações ou para retornar o recurso.
- Cabeçalhos relacionados:
  - *Content-type*: código *mime* indicando o tipo de informação contida no corpo (text/html, image.gif, etc.)
  - *Content-length*: tamanho do corpo (em bytes).
- Exemplo de requisição:  
GET /path/file.html HTTP/1.0  
*<linha em branco>*

- Exemplo de resposta:

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-type: text/html
Content-length: 1354
<linha em branco>
<html>
<body>
<h1> Título </h1>
...
...
```

### 3 Observações

#### 3.1 Métodos

- O protocolo define os seguintes métodos:

- GET: solicita a recuperação de um recurso.
- HEAD: solicita informações sobre um recurso. A resposta contém apenas a linha inicial e os cabeçalhos, mas não o recurso em si.
- POST: envia dados ao servidor para processamento/resposta.
- PUT: envia um recurso para armazenamento no servidor.
- DELETE: solicita a remoção de um recurso do servidor.
- TRACE: solicita que os *proxies* entre o cliente e o servidor se identifiquem nos cabeçalhos, mostrando o caminho tomado pelo recurso.
- OPTIONS: solicita informações sobre outros métodos que podem ser utilizados com o recurso/servidor.
- Outros: LINK, UNLINK, PATCH, CONNECT.

- Exemplo de requisição POST:

```
POST /path/script.cgi HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 32
<linha em branco>
nome=Manoel+Silva&favorito=azul
```

- O exemplo acima mostra a requisição enviada pela submissão de um formulário com dois campos *nome* e *favorito*, preenchidos respectivamente com *Manoel Silva* e *azul*. O formato utilizado denomina-se *URL-encoding* (ver abaixo).

### 3.2 Versão 1.1

- Esta versão traz diversas modificações importantes. Destacam-se:
  - Conexões persistentes. A conexão não é automaticamente fechada pelo servidor ao término da transação, a não ser que a requisição tenha especificado o cabeçalho *Connection: close*.
  - Fornece suporte a *cache*.
  - Permite múltiplos domínios para o mesmo endereço IP. O cabeçalho *Host: <domínio>* (para especificar o domínio desejado) é obrigatório. Se só houver um domínio, a informação *<domínio>* pode ser deixada em branco.

### 3.3 URL-encoding

- Para colocar uma determinada informação no formato URL-encoding:
  1. Valores “inseguros” (=, &, %, +, caracteres não imprimíveis) são convertidos para o formato  $\%xx$  (onde  $xx$  é o código ASCII do caracter).
  2. Espaços em branco são convertidos para o caracter “+”.
  3. Pares nome/valor são ligados com o caracter “&”.