GUIA ESTELAR

by Rocketseat

**HTTP**

**HiperText Transfer Protocol**

**Instructor: Mayk Brito**

**Lucas Omar Andrade Leal**

 HTTP é um protocolo (protocol) que permite a obtenção de recursos, como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web. Um documento completo é reconstruído a partir dos diferentes sub-documentos obtidos, como por exemplo texto, descrição do layout, imagens, vídeos, scripts e muito mais.

Clientes e servidores se comunicam trocando mensagens individuais (ao contrário de um fluxo de dados). As mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de solicitações (requests), ou também requisições, e as mensagens enviadas pelo servidor como resposta são chamadas de respostas (responses).

Projetado no início da década de 1990, o protocolo HTTP é extensível e evoluiu ao longo do tempo. Atua na camada de aplicação e é enviado sobre o protocoloTCP, ou em uma conexão TCP criptografada com TLS, embora qualquer protocolo de transporte confiável possa, teoricamente, ser usado. Devido à sua extensibilidade, ele é usado não só para buscar documentos de hipertexto, mas também imagens e vídeos ou publicar conteúdo em servidores, como nos resultados de formulário HTML (veja os elementos <html> e <form>). O HTTP também pode ser usado para buscar partes de documentos para atualizar páginas da Web sob demanda.

## Componentes de sistemas baseados em HTTP

O HTTP é um protocolo cliente-servidor: as requisições são enviados por uma entidade, o agente-usuário (ou um proxy em nome dele). A maior parte do tempo, o agente-usuário é um navegador da Web, mas pode ser qualquer coisa, como por exemplo um robô que varre a Web para preencher e manter um índice de mecanismo de pesquisa e coletar informações.

Cada requisição individual é enviada para um servidor, que irá lidar com isso 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.

Na realidade, existem muitos outros computadores entre o navegador e o servidor que está tratando a requisição: existem roteadores, modems e muito mais. Graças ao modelo de camadas da Web, essas funcionalidades estão escondidas nas camadas de rede e transporte, respectivamente. O HTTP está no topo da camada de aplicação. Apesar de ser importante diagnosticar problemas de conectividade, as camadas subjacentes são irrelevantes para a descrição do HTTP.

### Cliente: o agente-usuário

Oagente-usuário é qualquer ferramenta que age em nome do usuário. Essa função é predominantemente realizada pelo navegador Web; algumas poucas exceções são programas usados por engenheiros e desenvolvedores Web para debugar as suas aplicações.

O navegador sempre é a entidade que inicia as requisições, nunca o lado do servidor (embora alguns mecanismos tenham sido adicionados ao longo dos anos para simular mensagens iniciadas pelo servidor).

Para mostrar uma página Web, o navegador envia uma requisição para buscar o documento HTML da página. Ele então realiza uma análise sintática desse arquivo, buscando requisições adicionais correspondentes a scripts de execução, informações de layout (CSS) para apresentação e subrecursos contidos na página (geralmente imagens e vídeos). Depois o navegador interpreta esses recursos para mostrar ao usuário a página completa. Existem scripts executados pelo navegador que buscam mais recursos em fases subsequentes e conforme o uso da página e o navegador atualiza a página de acordo.

Uma página Web é um documento de hipertexto (HTML). Isso significa que algumas partes do texto mostrado são links (vínculos com outras páginas ou recursos da Web), os quais podem ser ativados (normalmente pelo clique do mouse) para buscar uma nova página, permitindo ao usuário redirecionar seu agente-usuário e navegar pela internet. O navegador traduz esses endereços em requisições HTTP e depois interpreta as respostas HTTP para mostrar ao usuário uma resposta transparente.

### O servidor de páginas Web

Do outro lado do canal de comunicação está o servidor que serve o documento requisitado pelo usuário. Um servidor se apresenta virtualmente apenas como uma máquina: isto porque o servidor pode ser uma coleção de servidores dividindo a carga (através de uma técnica chamada balanceamento de carga) ou também como um programa complexo que acessa outros servidores (como um cache, um servidor de banco de dados, servidores de e-commerce (lojas virtuais), etc.), gerando toda ou parte do documento solicitado.

Um servidor não é necessáriamente apenas uma máquina, mas vários servidores podem estar hospedados na mesma máquina. Com o HTTP/1.1 e o cabeçalho Host, eles podem até compartilhar o mesmo endereço IP.

### Proxies (ou representantes)

Entre o navegador Web e o servidor, vários computadores e máquinas transmitem as mensagens HTTP. Devido a estrutura em camadas da pilha Web, a maioria dessas máquinas operam em alguma das camadas: de transporte, de rede ou física, sendo transparente na camada da aplicação HTTP, e potencialmente exercendo um grande impacto na performance. Essas máquinas que operam na camada de aplicação são normalmente conhecidas como proxies (ou representantes, ou procuradores, etc). Eles podem ser transparentes ou não (alterações nas requisições não passam por eles), e podem desempenhar várias funções:

* cacheamento (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)

## Aspectos básicos 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.

### HTTP é extensível

Introduzidos no HTTP/1.0, os cabeçalhos HTTP fazem com que este protocolo seja fácil para estender e usá-lo para experimentos. Novas funcionalidades podem até ser introduzidas pelo simples acordo entre um cliente e um servidor sobre a nova semântica de um cabeçalho.

### HTTP não tem estado, mas tem sessões

HTTP é sem estado: não existe uma relação entre duas requisições sendo feitas através da mesma conexão. Isso traz um problema imediato para usuários que interagem com algumas páginas de forma coerente, por exemplo, usando um carrinho de compras de e-commerces\*. Mas como o fundamento básico do HTTP é não manter estados, cookies HTTP permitem que as sessões tenham estados. Usando a extensibilidade dos cabeçalhos, os cookies são adicionados ao fluxo do HTTP, permitindo que a criação de sessão em cada requisição HTTP compartilhem o mesmo contexto, ou o mesmo estado.

\* O problema do carrinho de compras de e-commerces e o protocolo HTTP: como o protocolo HTTP não guarda o estado das requisições e respostas, é impossível fazer com que um site guarde as informações de um carrinho de compras somente através do HTTP. Por exemplo, imagine que você irá comprar um computador novo e um jogo de xícaras de chá. Para que esses dados possam ser mantidos enquanto você navega no site do e-commerce olhando mais produtos (cada página visitada gera um novo par de requisição/resposta), duas estratégias podem ser usadas, já que o HTTP por si só, não permitiria isso:

1. Você possui um cadastro no e-commerce e um programa escrito no servidor é responsável por armazenar suas informações do carrinho; ou
2. Um programa escrito em linguagem 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 e conexões

Uma conexão é controlada na camada de transporte, e portanto fundamentalmente fora do controle do HTTP. Entretanto o HTTP não requer que o protocolo de transporte utilizado seja baseado em conexões, só requer que seja confiável ou não perca mensagens (sem pelo menos apresentar erros). Dentre os dois protocolos de transporte mais comuns na internet, o TCP é confiável e o UDP não. Portanto, o HTTP utiliza o padrão TCP, que é baseado em conexão, mesmo que nem sempre seja obrigatório o uso de uma conexão.

No protocolo HTTP/1.0 uma conexão TCP era aberta para cada par de requisição/resposta trocada, introduzindo duas grandes falhas: abrir uma conexão requer várias viagens de ida/volta de mensagens, e portanto 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).

Para contornar essas falhas, o protocolo HTTP/1.1 introduziu o conceito de linhas de produção (ou pipelining) — que se provou difícil de ser implementado — e conexões persistentes: as conexões TCPs feitas embaixo, podem ser parcialmente controladas usando o cabeçalho HTTP Connection. O HTTP/2.0 foi mais além, multiplexando várias mensagens através de uma única conexão, ajudando a manter a conexão mais quente, e mais eficiente.

Experimentos estão sendo feitos para projetar um protocolo de transporte mais adequado para o HTTP. Por exemplo, a Google está fazendo testes com o QUIC que é construído sobre o UDP para prover um protocolo de transporte mais confiável e eficiente.

## O que pode ser controlado pelo HTTP?

A natureza extensível do HTTP tem permitido mais controle e funcionalidade para a internet, ao longo do tempo. Cache e autenticação são funcionalidades suportadas desde o início da história do HTTP. A habilidade de relaxar as restrições na origem, em contraste, foi adicionada nos anos 2010s.

Aqui está uma lista de funcionalidades comuns, controláveis com HTTP:

* Cache  
  A forma como documentos são cacheados pode ser controlada pelo HTTP. O servidor pode instruir proxies e clientes, sobre o que cachear e por quanto tempo. O cliente pode instruir proxies de cache intermediários a ignorar o documento armazenado.
* Relaxamento das restrições na origem  
  Para prevenir bisbilhoteiros e outros invasores de privacidade, os navegadores reforçam estritamente a separação dos sites Web. Somente páginas de mesma origem podem acessar todas as informações de uma página Web. Apesar dessa restrição ser um fardo grande aos servidores, 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 (e pode até ter razões específicas de segurança para se fazer isso), como um tecido de retalhos.
* Autenticação  
  Algumas páginas podem ser protegidas para que apenas usuários específicos possam acessá-la. Autenticação básica pode ser fornecida pelo HTTP, usando tanto o cabeçalho WWW-Authenticate e similares, quanto configurando uma sessão específica usando cookies HTTP.
* Proxy e tunelamento  
  Servidores e/ou clientes estão frequentemente localizados em intranets e escondem seu verdadeiro endereço IP aos outros. Requisições HTTP recorrem aos proxies para contornar essa barreira na rede. Mas nem todos os proxies são proxies HTTP. O protocolo SOCKS, por exemplo, opera em um nível mais baixo. Outros protocolos, como ftp, podem ser tratados por esses proxies.
* 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 não só para os carrinhos de compras de e-commerces, mas também para qualquer site que permita customização das respostas a nível de usuário.

## Fluxo HTTP

Quando o cliente quer comunicar com um servidor, este sendo um servidor final ou um proxy, ele realiza os seguintes passos:

1. Abre uma conexão TCP: A conexão TCP será usada para enviar uma requisição, ou várias, e receber uma resposta. O cliente pode abrir uma nova conexão, reusar uma conexão existente, ou abrir várias conexões aos servidores.
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, mas o princípio se mantém o mesmo.

GET / HTTP/1.1

Host: developer.mozilla.org

Accept-Language: fr

1. Lê a resposta do servidor:

HTTP/1.1 200 OK

Date: Sat, 09 Oct 2010 14:28:02 GMT

Server: Apache

Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT

ETag: "51142bc1-7449-479b075b2891b"

Accept-Ranges: bytes

Content-Length: 29769

Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)

1. Fecha ou reutiliza a conexão para requisições futuras.

Se a linha de montagem (pipelining) estiver ativada, várias requisições podem ser enviadas sem que a primeira resposta seja totalmente recebida. A linha de montagem HTTP se provou difícil de ser implementada nas redes existentes, onde peças antigas de software coexistem com versões modernas. A linha de montagem HTTP tem sido substituída no HTTP/2.0 com multiplexação mais robusta de requisições dentro de um quadro (frame).

## Mensagens HTTP

HTTP/1.1 e mensagens mais antigas HTTP são legíveis às pessoas. No HTTP/2.0, essas mensagens são embutidas numa nova estrutura binária, um quadro, permitindo otimizações como compressão de cabeçalhos e multiplexação. Mesmo se somente parte da mensagem HTTP original for enviada nessa versão do HTTP, a semântica de cada mensagem permanece inalterada e o cliente reconstitui (virtualmente) a requisição HTTP/1.1 original. É portanto útil entender as mensagens HTTP/2.0 no formato da versão HTTP/1.1.

Existem dois tipos de mensagens, requisições e respostas, cada uma com seu próprio formato.

### Requisições

Exemplo de uma requisição HTTP:

As requisições consistem dos seguintes elementos:

* Um método HTTP, geralmente é um verbo como GET, POST, DELETE, PUT, etc, ou um substantivo como OPTIONS ou HEAD que define qual operação o cliente quer fazer. Tipicamente, um cliente que pegar um recurso (usando GET) ou publicar dados de um formulário HTML (usando POST), embora mais operações podem ser necessárias em outros casos.
* O caminho do recurso a ser buscado; a URL do recurso sem os elementos que são de contexto, por exemplo sem o protocolo protocol (http://), o domínio domain (aqui como developer.mozilla.org), ou a porta port TCP (aqui indicada pelo 80 que é ocultado por ser o número da porta padrão)
* A versão do protocolo HTTP.
* Cabeçalhos opcionais que contém informações adicionais para os servidores.
* Ou um corpo de dados, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.

### Respostas

Examplo de resposta HTTP:

Respostas consistem dos seguintes elementos:

* A versão do protocolo HTTP que elas seguem.
* Um código de status, indicando se a requisição foi bem sucedida, ou não, e por quê.
* Uma mensagem de status, uma pequena descrição informal sobre o código de status.
* Cabeçalhos HTTP, como aqueles das requisições.
* Opcionalmente, um corpo com dados do recurso requisitado.

## APIs baseadas no HTTP

A API mais utilizada construída em cima do HTTP é a XMLHttpRequest, que pode ser usada para trocar dados entre um user agent e um servidor.

Outra API, de eventos enviados pelo servidor, é um serviço de mão-única que permite um servidor enviar eventos ao cliente, usando HTTP como um mecanismo de transporte. Usando a interface EventSource, o cliente abre uma conexão e estabelece os manipuladores de evento. O navegador do cliente converte automaticamente as mensagens que chegam pelo fluxo HTTP em objetos Event apropriados, entregando-os aos manipuladores de evento que foram registrados para os tipos de eventos type se conhecidos, ou para o manipulador de evento onmessage se nenhum manipulador de evento específico ao tipo foi definido.

## Conclusão

O HTTP é um protocolo extensível que é fácil de se usar. A arquitetura cliente-servidor, combinada com a habilidade de simplesmente adicionar cabeçalhos, permite que o HTTP avance suas funcionalidades juntamente com a elasticidade da Web.

Embora o HTTP/2.0 adicione mais complexidade, embutindo mensagens HTTP em quadros para melhorar a performance, a estrutura básica das mensagens continua a mesma desde o HTTP/1.0. Fluxo de sessões permanece simples, permitindo-o a ser investigado, e depurado com um simples monitor de mensagens HTTP.