HTTP: O Protocolo mais importante da Internet

HTTP x HTTPS

- **Internet e Web - Uma visão simplificada do que acontece quando você vê uma página em um navegador**

  A Internet é a infraestrutura técnica que faz a Web possível.

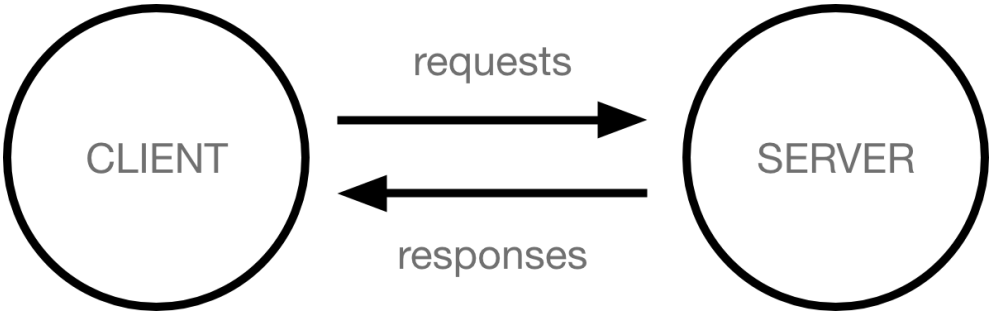
  Basicamente, a Internet é uma gigantesca rede de computadores que se comunicam juntos.

  A Web é uma das várias ferramentas de acesso a essa rede.

  - Como a internet funciona - Deeper Dive
    - Quando dois computadores precisam se comunicar, você precisa conecta-los, seja fisicamente (normalmente com um Cabo de rede) ou de uma forma sem fio (por exemplo com sistemas WiFi ou Bluetooth). Todos os computadores modernos suportam alguma(s) dessas conexões.
    - Cada computador na rede está conectado à um pequeno computador especial chamado de roteador. Este roteador tem um único trabalho: como um sinalizador em uma estação de trem, ter certeza de que a mensagem enviada por um determinado computador chegue ao computador destinatário corretamente. Para enviar uma mensagem para o computador B, o computador A deve enviar a mensagem para o roteador, que por sua vez encaminha a mensagem para o computador B e tem a certeza de que a mensagem não foi entregue ao computador C.
    - Mas como conectar centenas, milhares, bilhões de computadores? Claro que um único roteador não pode se adaptar para tanto.
    - A estrutura do telefone já conecta nossa casa com o resto do mundo. Para conectar nossa rede a rede telefônica, precisamos de um equipamento especial chamado modem. Este modem transforma a informação da nossa rede em uma informação gerenciável pela rede telefônica e vice-versa.
    - Então nós estamos conectados à infraestrutura telefônica. O próximo passo é enviar mensagens da nossa rede para a rede que nós desejamos alcançar. Para fazer isto, vamos precisar conectar nossa rede a um Provedor de Serviço de Internet (ISP, em inglês).
    - Um ISP é uma companhia que gerencia alguns roteadores especiais que são conectados e pode também acessar roteadores de outros ISPs. Então a mensagem da nossa rede é transportada para a rede de redes do ISP e então para a rede de destino. A Internet é composta por toda esta infraestrutura de redes.

- **Como a Web Funciona**

  Computadores conectados à web são chamados clientes e servidores. Um diagrama simplificado de como eles interagem pode ter essa aparência:



  - Clientes são dispositivos conectados à internet dos usuários da web (por exemplo, seu computador conectado ao seu WiFi ou seu telefone conectado à sua rede móvel) e programas de acesso à Web disponíveis nesses dispositivos (geralmente um navegador como Firefox ou Chrome).
  - Servidores são computadores que armazenam páginas, sites ou aplicativos. Quando o dispositivo de um cliente quer acessar uma página, uma cópia dela é baixada do servidor para a máquina do cliente para ser apresentada no navegador web do usuário.

  Esse assunto será aprofundado melhor no tópico sobre Protocolos.

- **IP**
  - IP é sigla de *Internet Protocol* (Protocolo de Internet)
  - Como a Internet é uma rede global de computadores, cada computador conectado à Internet deve ter um endereço exclusivo.
  - Os endereços de IP estão no formato nnn.nnn.nnn.nnn, em que nnn deve ser um número de 0 a 255. Esse endereço é conhecido como endereço IP.
  - Se você estiver conectado à Internet, seu computador possui um endereço IP exclusivo.
  - Se você se conectar à Internet por meio de um Provedor de Serviços de Internet, normalmente receberá um endereço IP temporário durante a sessão de discagem.
  - Se você se conectar à Internet de uma rede local, seu computador pode ter um endereço IP permanente ou pode obter um temporário de um servidor DHCP (Dynamic Host Configuration Protocol).
  - IPv6 e IPv4

    IPv4 e IPV6 são versões do sistema IP.

    Desde a criação da internet, o protocolo usado até então era o IPv4, que é a sigla para Internet Protocol version 4. Possui endereços no padrão 32 bits e sustenta cerca de 4 bilhões de combinações de endereços IP em todo o mundo.

No entanto, o IPv4 não possui mais capacidade de expansão para sustentar a demanda de internet global, além de já apresentar falhas de segurança.

O IPv6 é, então, a versão 6 do Protocolo de Internet, com endereços no padrão 128 bits. Ou seja, trata-se do sucessor do IPv4, uma vez que o antigo protocolo não mais suporta a demanda de endereços. Atuando em 128 bits, o IPv6 suporta cerca de 340 undecilhões de endereços, contra 4 bilhões suportados pelo IPv4.

- **DNS**
  - Quando navegamos na Web com nossos navegadores, normalmente utilizamos os nomes de domínios para chegar a um website.
  - O navegador precisa encontrar em qual servidor web a página está hospedada para que ele possa mandar mensagens HTTP ao lugar certo.
  - Como o seu navegador sabe onde um servidor da web está localizado? A resposta é o Domain Name Service ou DNS. Esses são servidores especiais que relacionam o endereço da web que você digita no seu navegador (como "google.com") com o endereço real do site (IP).
  - "Nomes de Domínios" são uma parte fundamental da infraestrutura da Internet. Eles provêm um endereço legível para qualquer servidor web disponível na Internet.
  - É um banco de dados distribuído que rastreia os nomes dos computadores e seus endereços IP correspondentes, como um catálogo de endereços para sites.

- **Pacotes de rede**
  - O termo pacote de rede ou pacote de dados aparece sempre que alguém fala em protocolos de transmissão. O termo "pacotes" é usado para descrever o formato no qual os dados são enviados do servidor para o cliente.
  - Basicamente, quando os dados são enviados pela web, eles são enviados como milhares de pequenos blocos, para que muitos usuários diferentes possam baixar o mesmo site ao mesmo tempo.
  - Se os websites fossem enviados como um grande bloco, somente um usuário por vez poderia baixá-los, o que, obviamente, tornaria a web muito ineficiente e nada prática.

Protocolos

- Introdução

  Podemos dizer que protocolos são as regras de comunicação na internet.

Hoje existem milhares de protocolos no mundo de TI, os mais comuns são:

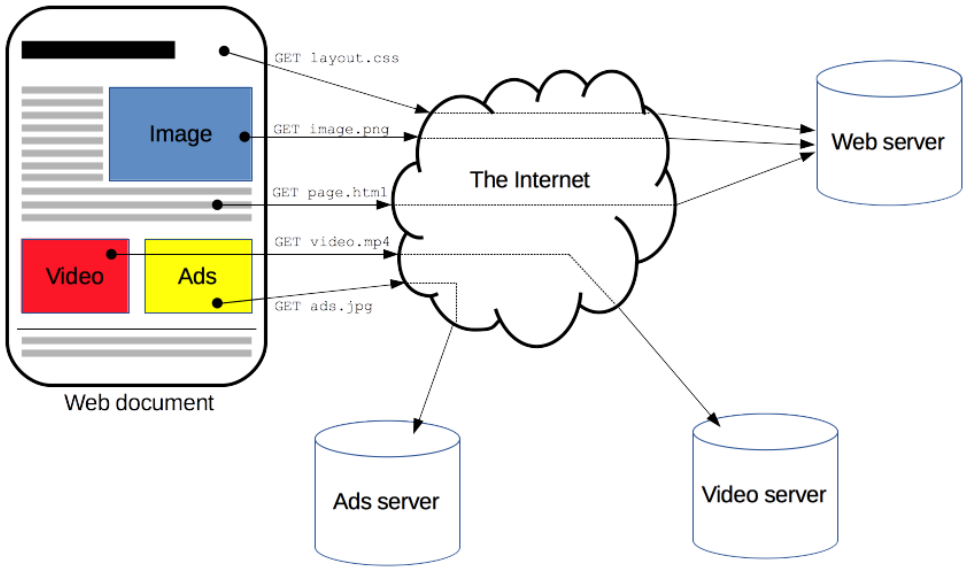
- HTTP: Navegação na Web
- SMTP: Envio de Email
- FTP: Transporte de Arquivos
- POP3: Recebimento de email
- SSH: Login remoto seguro
- WAP: Telefonia Móvel
- IMAP: Troca de email
- TCP/IP: Envio e recebimento de dados entre computadores

Destes protocolos listados, o mais importante para nós é o O HTTP.

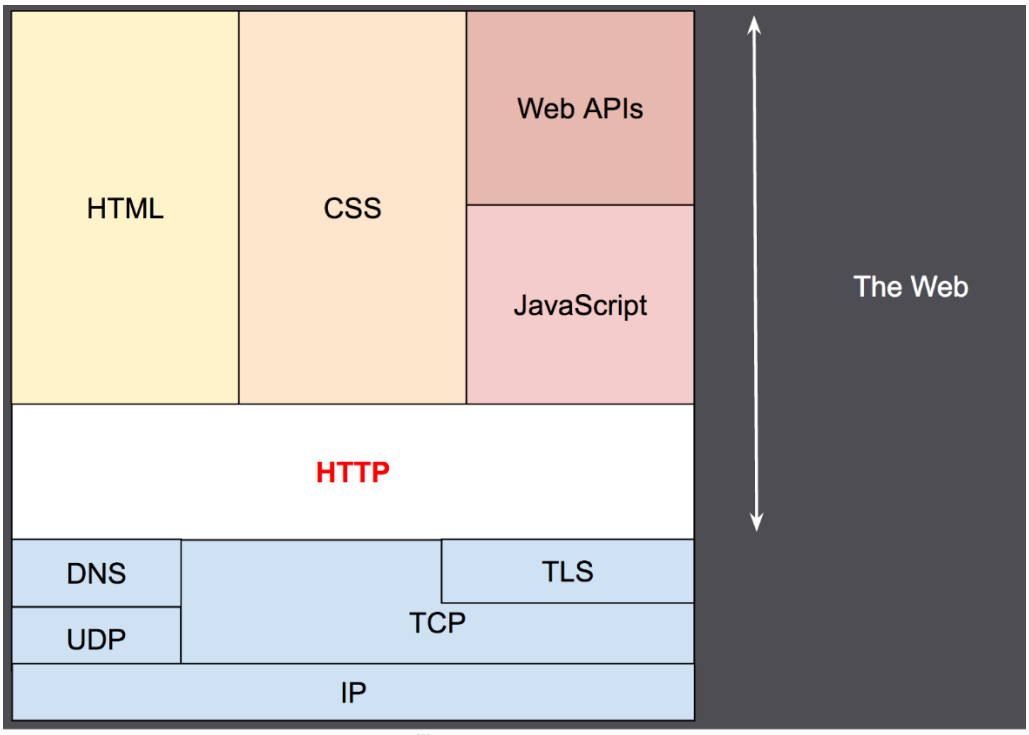
- HTTP

  Protocolo de Transferência de Hypertexto (Hypertext Transfer Protocol)

  É um protocolo que permite a obtenção de recursos, tais 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 (em oposição a 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 HTTP é um protocolo extensível que evoluiu ao longo do tempo. É um protocolo de camada de aplicação que é enviado sobre TCP, 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 para não apenas buscar documentos de hipertexto, mas também imagens e vídeos ou publicar conteúdo em servidores, como nos resultados de formulário HTML .

O HTTP também pode ser usado para buscar partes de documentos para atualizar páginas da Web sob demanda.

- TCP/IP

  TCP (Transmission Control Protocol, em português, Protocolo de Controle de Transmissão) é um importante protocolo de rede que permite dois hosts se conectem e troquem dados. TCP garante a entrega de dados e pacotes na mesma ordem que foram enviados.

  Ao se conectar a um site da Web, por exemplo em www.example.org, o usuário está usando o pacote de protocolos TCP/IP. O modelo TCP/ IP foi projetado em 1970 com 4 camadas distintas.

  1. O Link descreve o acesso à mídia física (por exemplo, usando a placa de rede)
  2. A Internet descreve o envelope e o roteamento dos dados - como são empacotados (IP)
  3. Transporte descreve a maneira como os dados são entregues do ponto de partida ao destino final (TCP, UDP)

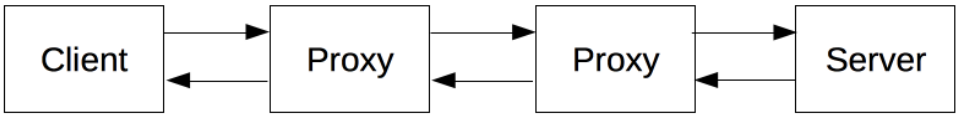
4. O aplicativo descreve o significado ou formato das mensagens transferidas (HTTP)

Transport Layer Security (TLS), previamente conhecido como Secure Sockets Layer (SSL), é um protocolo usado por aplicativos para se comunicar de forma segura em toda a rede, evitando adulteração e espionagem no email, navegador, mensagens e outros protocolos.

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 rastreia a Web para preencher e manter um índice de mecanismo de pesquisa.

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 (*layers*), 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

O *user-agent* 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 depurar as suas aplicações.

O navegador é **sempre** a entidade que inicia as requisições. Nunca é o 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 o documento completo, a página Web.
- Scripts executados pelo navegador podem buscar mais recursos em fases subsequentes e o navegador atualiza a página Web de acordo.

Uma página Web é um documento de hipertexto. 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*, etc.), gerando todo ou parte do documento solicitado.

Um servidor não é necessariamente apenas uma máquina, mas vários servidores podem estar hospedados na mesma máquina. Com o HTTP/1.1 e o cabeçalho `[Host](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/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.

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](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/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](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/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:

- 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.
- 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`

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)
```

- 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

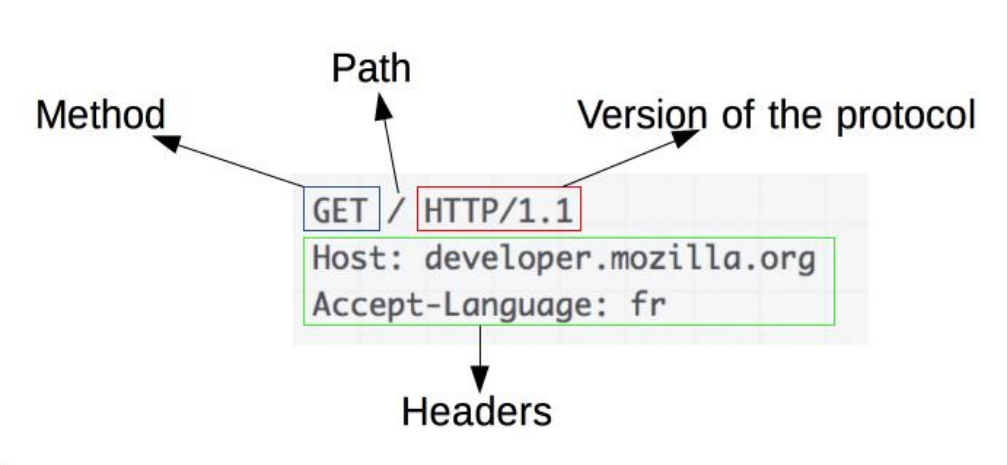
HTTP/1.1 e mensagens mais antigas HTTP são legíveis às pessoas. No HTTP/2.0, essas mensagens são embutidas numa nova estrutura binária, um

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:



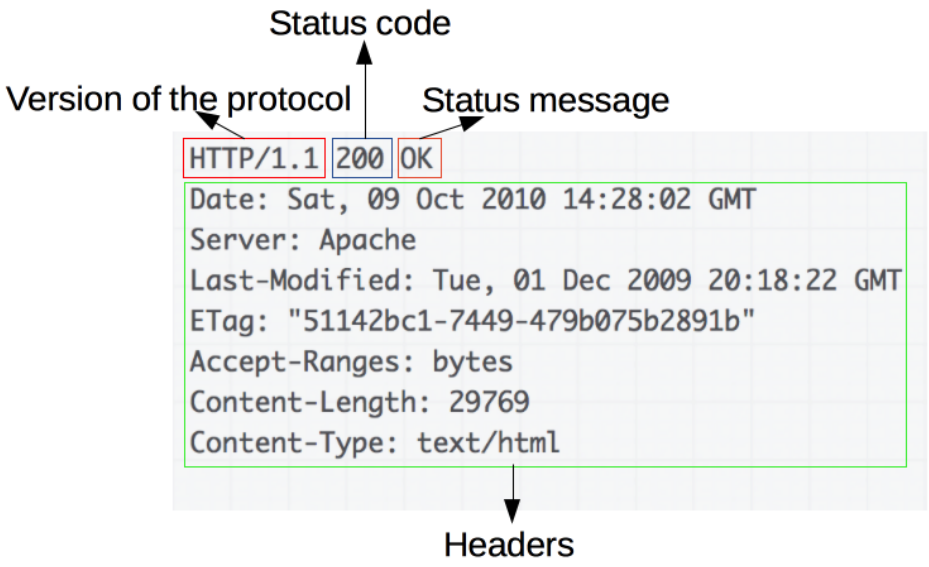
As requisições consistem dos seguintes elementos:

- Um método HTTP, geralmente é um verbo como `[GET](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/GET>)`, `[POST](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/POST>)`, `[DELETE](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/DELETE>)`, `[PUT](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/PUT>)`, etc, ou um substantivo como `[OPTIONS](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/OPTIONS>)` ou `[HEAD](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/HEAD>)` que define qual operação o cliente quer fazer. Tipicamente, um cliente que pegar um recurso (usando `[GET](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/GET>)`) ou publicar dados de um formulário HTML (usando `[POST](<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/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](<https://developer.mozilla.org/pt-BR/docs/Web/API/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](<https://developer.mozilla.org/pt-BR/docs/Web/API/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](<https://developer.mozilla.org/pt-BR/docs/Web/API/Event>)` apropriados, entregando-os aos manipuladores de evento que foram registrados para os tipos de eventos `[type](<https://developer.mozilla.org/pt-BR/docs/Web/API/Event/type>)` se conhecidos, ou para o manipulador de evento `[onmessage](<https://developer.mozilla.org/pt-BR/docs/Web/API/EventSource/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.

- Segurança na Web

Então, o que acontece exatamente?

Quando você digita um endereço da web no seu navegador:

1. O navegador vai para o servidor de DNS e encontra o endereço verdadeiro de onde o site está hospedado.
2. O navegador manda uma mensagem de requisição HTTP para o servidor, pedindo que envie uma cópia do site ao cliente. Esta mensagem e todos os outros dados enviados entre o cliente e o servidor são enviados pela sua conexão à internet usando TCP/IP.
3. Se o servidor aprovar a requisição do cliente, o servidor enviará ao cliente uma mensagem "200 OK", que significa "Claro que você pode ver esse site! Aqui está" e então começa a enviar os arquivos do site para o navegador como uma série de pequenos pedaços chamados pacotes de dados.
4. O navegador monta os pequenos pedaços em um site completo e o mostra a você.

Fontes utilizadas: Notion Laura

MDN Web Docs

Professor Messer's CompTIA N10-007 Network+ Course - Professor Messer IT Certification Training Courses

Aula 1 12/09/2020 (sábado)

Apresentação do time - boas-vindas.

Resolução dos exercicios de fixação:

AULA1

HTTP: O Protocolo mais importante da Internet

https://liniribeiro.github.io/2020-08-08-http/

HTTP x HTTPS

https://liniribeiro.github.io/2020-08-30-httpxhttps/

Maria vai com as Devs

Olá! Fico muito feliz que você chegou até aqui! Primeiramente gostaria de te dizer: "Lindo diaaaaaa!"

Criei alguns exercícios para ajudar na fixação do assunto apresentado.

Gostaria de lembrar que, não há nenhum tipo de pontuação, as respostas são anonimas e você pode realizá-las quantas vezes precisar.

1) Qual protocolo de comunicação realiza a transferência de hipertexto de forma criptografada, aumentando, significativamente, a segurança dos dados na internet?

EHTP (Encrypted Hypertext Transfer Protocol)

EHLP (Encrypted Hypertext Layer Protocol)

HSLP (Hypertext Secure Layer Protocol)

HTTPE (Hypertext Transfer Protocol Encrypted)

(X)HTTPS (Hypertext Transfer Protocol Secure)

Feedback

Exatamente!! HTTPS é o protocolo de comunicação de forma criptografada!

2) O protocolo HTTP segue o modelo Cliente-Servidor. O que o navegador (como Chrome ou Firefox) representa nesse modelo? O cliente ou o servidor?

(X)Cliente

Servidor

Nenhuma das alternativas

Feedback

Exatamente! Nesse modelo, o navegador representa o cliente. É importante saber que nem só navegadores dominam o protocolo HTTP. podemos possuir dispositivos móveis ou aplicações desktop que podem ser o cliente.

3) O cliente inicia a comunicação e o servidor responde. No entanto, qual é o papel do HTTP entre o cliente e o servidor?

Definir o melhor algoritmo de pesquisa

Comprimir os dados

Definir uma estrutura de dados

Estabelecer regras de comunicação

(X)Estabelecer regras de comunicação

Feedback

Na internet, como já vimos, o idioma mais comum é o HTTP. Ele é responsável por definir a forma de como os dados são trafegados na rede através de várias regras. Portanto, todo mundo que conhece o idioma HTTP poderá receber e enviar dados e participar dessa conversa!

4) O HTTP não é o único protocolo de comunicação que existe. Aliás, existem milhares de protocolos no mundo de TI, no entanto o HTTP é de longe o mais popular.Na lista abaixo, há um item que não representa um protocolo para internet. Qual é exatamente? Pesquise se for necessário.

SMTP

(X)SQL

FTP

BitTorrent

Feedback

SQL (Structured Query Language) não é um protocolo para internet, e sim uma linguagem de consulta para banco de dados.

5) O que acontece com nossos dados quando usamos HTTP , ou seja sem a letra S ao final?

Os dados são criptografados, para impedir a visualização por intermediários.

(X)Os dados são transportados em texto puro para o servidor, visível para qualquer um.

Usamos automaticamente um certificado digital para provar a identidade de um site.

Feedback

Exato, nossos dados são enviados em texto puro, ficando visível para qualquer um que consiga interceptar nossa conexão!

6) Qual é a finalidade das autoridades certificadoras?

Garantir que podemos confiar naquele certificado (identidade).

Importar/Exportar chaves publicas do servidor.

Realizar a criptografia dos dados da requisição.

Usada para registrarmos nomes de domínio (DNS).

(X)Realizar a criptografia dos dados da requisição.

Feedback

Essa garantia é feita através de uma assinatura digital. A autoridade certificadora (CA) assina digitalmente o certificado! Como na vida real, existem também no mundo digital: assinaturas!

Uma autoridade certificadora (CA - Certificate Authority) é um órgão que garante ao navegador e ao usuário que a identidade de um servidor é realmente válida. Portanto, podemos trocar informações com este sem riscos!

7)Qual dessas alternativas é verdadeira?

Uma comunicação com HTTP sempre é iniciada pelo cliente que manda uma requisição ao servidor esperando por uma resposta.

Em HTTP o servidor sempre envia uma requisição ao cliente para poder alterar algo na tela.

Quando trabalhamos com HTTP, a comunicação é sempre iniciada pelo lado do cliente que envia uma requisição ao servidor em busca de uma resposta. Mas em alguns casos, o servidor também pode enviar uma requisição ao cliente.

Feedback

Correto!! É importante lembrarmos que a comunicação sempre começa com o cliente: é ele quem pede as informações. O servidor responde apenas o que foi requisitado e nunca inicia a comunicação :)=

No HTTP: Request -> espera -> Resposta

Vimos que há diversos códigos HTTP. Vendo os códigos abaixo, qual deles representa algum problema gerado no servidor?

300

404

301

(x)500

Feedback

Correto!

A classe 5xx significa que houve algum problema no servidor.

Por exemplo: 500 - Internal Server Error, ou outro famoso: 503 - Service Unavailable.

O código 500 acontece com frequência quando estamos desenvolvendo uma aplicação web e, ao testar, percebemos que erramos algo na lógica que gerou um problema no servidor.