**Evolution of HTTP**

HTTP (HyperText Transfer Protocol) é o protocolo subjacente da World Wide Web. Desenvolvido por Tim Berners-Lee e sua equipe entre 1989 e 1991, o HTTP passou por muitas mudanças que ajudaram a manter sua simplicidade e, ao mesmo tempo, moldaram sua flexibilidade. Continue lendo para saber como o HTTP evoluiu de um protocolo projetado para a troca de arquivos em um ambiente de laboratório semiconfiável para um labirinto moderno da internet que transmite imagens e vídeos em alta resolução e 3D.

**Invention of the World Wide Web**

Em 1989, enquanto trabalhava no CERN, Tim Berners-Lee escreveu uma proposta para construir um sistema de hipertexto na internet. Inicialmente chamado de Mesh, foi posteriormente renomeado para World Wide Web durante sua implementação em 1990. Construído sobre os protocolos TCP e IP existentes, consistia em quatro blocos de construção:

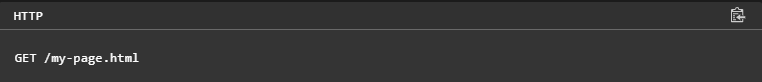
* Um formato textual para representar documentos de hipertexto, a HyperText Markup Language (HTML).
* Um protocolo para troca desses documentos é o HyperText Transfer Protocol (HTTP).
* Um cliente para exibir (e editar) esses documentos, o primeiro navegador da web chamado WorldWideWeb.
* Um servidor para dar acesso ao documento, uma versão inicial do httpd

Esses quatro blocos de construção foram concluídos no final de 1990, e os primeiros servidores estavam operando fora do CERN no início de 1991. Em 6 de agosto de 1991, Tim Berners-Lee publicou uma publicação no grupo de notícias público alt.hypertext. Este é hoje considerado o início oficial da World Wide Web como um projeto público.

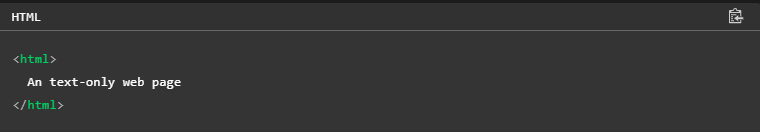
O protocolo HTTP usado nessas fases iniciais era muito simples. Mais tarde, foi apelidado de HTTP/0.9 e, às vezes, é chamado de protocolo de uma linha.

**HTTP/0.9 – O protocolo de uma linha**

A versão inicial do HTTP não tinha número de versão; posteriormente, foi chamada de 0.9 para diferenciá-la das versões posteriores. O HTTP/0.9 era extremamente simples: as solicitações consistiam em uma única linha e começavam com o único método possível, GET, seguido pelo caminho para o recurso. A URL completa não foi incluída, pois o protocolo, o servidor e a porta não eram necessários após a conexão com o servidor.



A resposta também foi extremamente simples: consistia apenas no arquivo em si.



Ao contrário das evoluções subsequentes, não havia cabeçalhos HTTP. Isso significava que apenas arquivos HTML podiam ser transmitidos. Não havia códigos de status ou erro. Se houvesse algum problema, um arquivo HTML específico era gerado e incluía uma descrição do problema para consumo humano.

**HTTP/1.0 – Construindo extensibilidade**

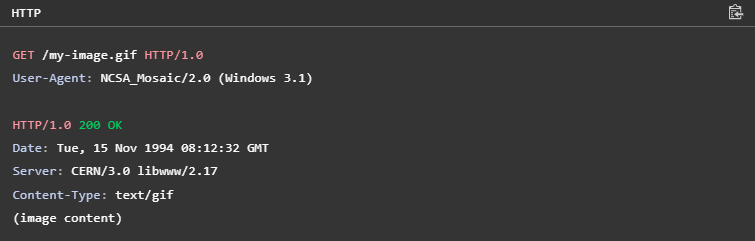
O HTTP/0.9 era muito limitado, mas navegadores e servidores rapidamente o tornaram mais versátil:

* Informações de versão foram enviadas dentro de cada solicitação (HTTP/1.0 foi anexado à linha GET).
* Uma linha de código de status também era enviada no início de uma resposta. Isso permitia que o próprio navegador reconhecesse o sucesso ou a falha de uma solicitação e adaptasse seu comportamento de acordo. Por exemplo, atualizando ou usando seu cache local de uma maneira específica.
* O conceito de cabeçalhos HTTP foi introduzido tanto para solicitações quanto para respostas. Metadados puderam ser transmitidos e o protocolo tornou-se extremamente flexível e extensível.
* Documentos diferentes de arquivos HTML simples podem ser transmitidos graças ao cabeçalho Content-Type.

Nesse momento, uma solicitação e resposta típicas eram assim:



Seguiu-se uma segunda conexão e uma solicitação para buscar a imagem (com a resposta correspondente):



Entre 1991 e 1995, essas práticas foram introduzidas com uma abordagem experimental. Um servidor e um navegador adicionavam um recurso e verificavam se ele ganhava força. Problemas de interoperabilidade eram comuns. Em um esforço para solucionar esses problemas, um documento informativo descrevendo as práticas comuns foi publicado em novembro de 1996. Esse documento ficou conhecido como RFC 1945 e definiu o HTTP/1.0.

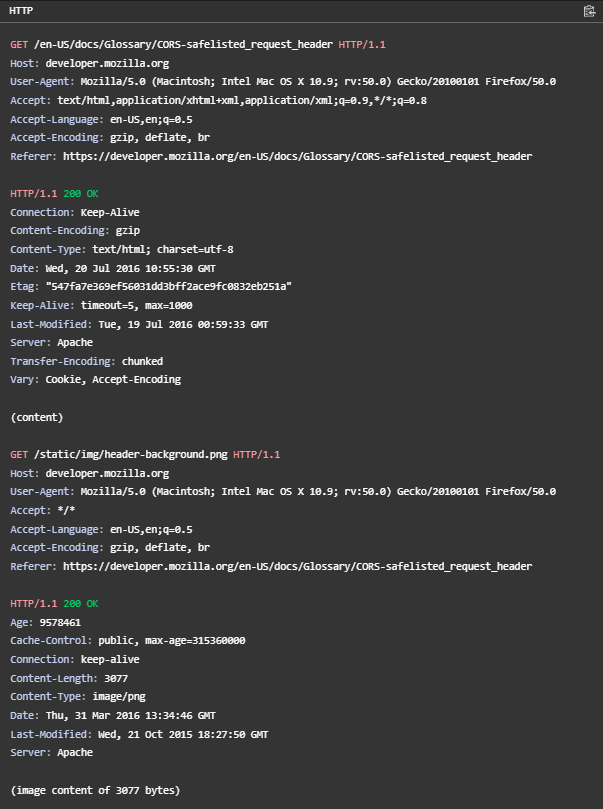
**HTTP/1.1 – O protocolo padronizado**

Enquanto isso, a padronização adequada estava em andamento. Isso ocorreu paralelamente às diversas implementações do HTTP/1.0. A primeira versão padronizada do HTTP, o HTTP/1.1, foi publicada no início de 1997, apenas alguns meses após o HTTP/1.0.

O HTTP/1.1 esclareceu ambiguidades e introduziu inúmeras melhorias:

* Uma conexão podia ser reutilizada, o que economizava tempo. Não era mais necessário abri-la várias vezes para exibir os recursos incorporados no mesmo documento original.
* O pipeline foi adicionado. Isso permitiu o envio de uma segunda solicitação antes que a resposta à primeira fosse totalmente transmitida. Isso reduziu a latência da comunicação.
* Respostas fragmentadas também foram suportadas.
* Mecanismos adicionais de controle de cache foram introduzidos.
* A negociação de conteúdo, incluindo idioma, codificação e tipo, foi introduzida. Um cliente e um servidor agora podiam concordar sobre qual conteúdo trocar.
* Graças ao cabeçalho Host, a capacidade de hospedar diferentes domínios a partir do mesmo endereço IP permitiu a colocação de servidores.

Um fluxo típico de solicitações, tudo por meio de uma única conexão, teria a seguinte aparência:



O HTTP/1.1 foi publicado pela primeira vez como RFC 2068 em janeiro de 1997.

**Mais de duas décadas de desenvolvimento**

A extensibilidade do HTTP facilitou a criação de novos cabeçalhos e métodos. Embora o protocolo HTTP/1.1 tenha sido refinado em duas revisões, a RFC 2616 publicada em junho de 1999 e a RFC 7230-RFC 7235 publicadas em junho de 2014 antes do lançamento do HTTP/2, ele permaneceu extremamente estável por mais de 15 anos. O HTTP/1.1 foi atualizado novamente em 2022 com a RFC 9110. Não apenas o HTTP/1.1 foi atualizado, mas todo o HTTP foi revisado e agora está dividido nos seguintes documentos: semântica (RFC 9110), cache (RFC 9111) aplicável a todas as versões do HTTP, e HTTP/1.1 (RFC 9112), HTTP/2 (RFC 9113) e HTTP/3 (RFC 9114). Além disso, a especificação finalmente alcançou o status de Padrão da Internet (STD 97), enquanto antes era sempre um padrão proposto/rascunho.

Usando HTTP para transmissões seguras

A maior mudança no HTTP ocorreu no final de 1994. Em vez de enviar HTTP por uma pilha TCP/IP básica, a empresa de serviços de informática Netscape Communications criou uma camada adicional de transmissão criptografada: o SSL. O SSL 1.0 nunca foi lançado ao público, mas o SSL 2.0 e seu sucessor, o SSL 3.0, permitiram a criação de sites de comércio eletrônico. Para isso, criptografaram e garantiram a autenticidade das mensagens trocadas entre o servidor e o cliente. O SSL foi eventualmente padronizado e se tornou o TLS.

No mesmo período, tornou-se evidente a necessidade de uma camada de transporte criptografada. A web deixou de ser uma rede predominantemente acadêmica e, em vez disso, tornou-se uma selva onde anunciantes, indivíduos aleatórios e criminosos competiam pelo máximo de dados privados possível. À medida que os aplicativos desenvolvidos sobre HTTP se tornaram mais poderosos e passaram a exigir acesso a informações privadas, como catálogos de endereços, e-mails e localização do usuário, o TLS tornou-se necessário fora do contexto de uso do comércio eletrônico.

Usando HTTP para aplicações complexas

Tim Berners-Lee não imaginou originalmente o HTTP como um meio somente leitura. Ele queria criar uma web onde as pessoas pudessem adicionar e mover documentos remotamente — uma espécie de sistema de arquivos distribuído. Por volta de 1996, o HTTP foi expandido para permitir a criação de arquivos, e um padrão chamado WebDAV foi criado. Ele cresceu para incluir aplicativos específicos como o CardDAV para lidar com entradas do catálogo de endereços e o CalDAV para lidar com calendários. Mas todas essas extensões \*DAV tinham uma falha: elas só eram utilizáveis ​​quando implementadas pelos servidores.

Em 2000, um novo padrão para o uso do HTTP foi criado: a transferência de estado representacional (ou REST). A API não se baseava nos novos métodos HTTP, mas sim no acesso a URIs específicos com métodos básicos do HTTP/1.1. Isso permitiu que qualquer aplicação web deixasse uma API recuperar e modificar seus dados sem precisar atualizar os navegadores ou os servidores. Todas as informações necessárias eram incorporadas aos arquivos que os sites serviam por meio do HTTP/1.1 padrão. A desvantagem do modelo REST era que cada site definia sua própria API RESTful não padrão e tinha controle total sobre ela. Isso diferia das extensões \*DAV, nas quais clientes e servidores eram interoperáveis. As APIs RESTful se tornaram muito comuns na década de 2010.

Desde 2005, mais APIs foram disponibilizadas para páginas da web. Várias dessas APIs criam extensões do protocolo HTTP para fins específicos:

* Eventos enviados pelo servidor, onde o servidor pode enviar mensagens ocasionais para o navegador.
* WebSocket, um novo protocolo que pode ser configurado atualizando uma conexão HTTP existente.

Relaxando o modelo de segurança da web

O HTTP é independente do modelo de segurança da web, conhecido como política de mesma origem. De fato, o modelo de segurança da web atual foi desenvolvido após a criação do HTTP! Ao longo dos anos, provou-se útil remover algumas restrições dessa política sob certas restrições. O servidor transmitia ao cliente o quanto e quando remover tais restrições usando um novo conjunto de cabeçalhos HTTP. Estes foram definidos em especificações como o Cross-Origin Resource Sharing (CORS) e a Content Security Policy (CSP).

Além dessas grandes extensões, muitos outros cabeçalhos foram adicionados, às vezes apenas experimentalmente. Cabeçalhos notáveis ​​são o cabeçalho Do Not Track (DNT) para controle de privacidade, X-Frame-Options e Upgrade-Insecure-Requests, mas existem muitos outros.

**HTTP/2 – Um protocolo para maior desempenho**

Com o passar dos anos, as páginas da web tornaram-se mais complexas. Algumas delas eram até mesmo aplicativos por si só. Mais mídia visual foi exibida e o volume e o tamanho dos scripts que adicionavam interatividade também aumentaram. Muito mais dados foram transmitidos por meio de um número significativamente maior de solicitações HTTP, o que gerou mais complexidade e sobrecarga para conexões HTTP/1.1. Para compensar isso, o Google implementou um protocolo experimental SPDY no início da década de 2010. Essa forma alternativa de troca de dados entre cliente e servidor despertou o interesse de desenvolvedores que trabalhavam tanto em navegadores quanto em servidores. O SPDY definiu um aumento na responsividade e resolveu o problema da transmissão duplicada de dados, servindo como base para o protocolo HTTP/2.

O protocolo HTTP/2 difere do HTTP/1.1 em alguns aspectos:

* É um protocolo binário, e não um protocolo de texto. Não pode ser lido e criado manualmente. Apesar dessa dificuldade, permite a implementação de técnicas de otimização aprimoradas.
* É um protocolo multiplexado. Requisições paralelas podem ser feitas pela mesma conexão, removendo as restrições do protocolo HTTP/1.x.
* Ele compacta cabeçalhos. Como estes costumam ser semelhantes em um conjunto de solicitações, isso elimina a duplicação e a sobrecarga dos dados transmitidos.

Padronizado oficialmente em maio de 2015, o uso do HTTP/2 atingiu o pico em janeiro de 2022, com 46,9% de todos os sites (veja estas estatísticas). Sites com alto tráfego apresentaram a adoção mais rápida, em um esforço para economizar custos com transferência de dados e orçamentos subsequentes.

Essa rápida adoção provavelmente se deveu ao fato de o HTTP/2 não exigir alterações em sites e aplicativos. Para usá-lo, bastava um servidor atualizado que se comunicasse com um navegador recente. Apenas um conjunto limitado de grupos foi necessário para desencadear a adoção e, à medida que as versões antigas de navegadores e servidores foram renovadas, o uso aumentou naturalmente, sem trabalho significativo para os desenvolvedores web.

**Evolução pós-HTTP/2**

A extensibilidade do HTTP ainda está sendo utilizada para adicionar novos recursos. Notavelmente, podemos citar novas extensões do protocolo HTTP que surgiram em 2016:

* O suporte a Alt-Svc permitiu a dissociação da identificação e da localização de um determinado recurso. Isso resultou em um mecanismo de cache de CDN mais inteligente.
* A introdução de dicas de cliente permitiu que o navegador ou cliente comunicasse proativamente informações sobre seus requisitos e restrições de hardware ao servidor.
* A introdução de prefixos relacionados à segurança no cabeçalho Cookie ajudou a garantir que Cookie header não pudesse ser alterados.

**HTTP/3 - HTTP sobre QUIC**

A próxima versão principal do HTTP, o HTTP/3, tem a mesma semântica das versões anteriores, mas usa QUIC em vez de TCP para a camada de transporte. Em outubro de 2022, 26% de todos os sites usavam HTTP/3.

O QUIC foi projetado para fornecer latência muito menor para conexões HTTP. Assim como o HTTP/2, é um protocolo multiplexado, mas o HTTP/2 roda em uma única conexão TCP, de modo que a detecção de perda de pacotes e a retransmissão tratadas na camada TCP podem bloquear todos os fluxos. O QUIC executa múltiplos fluxos sobre UDP e implementa a detecção de perda de pacotes e a retransmissão de forma independente para cada fluxo, de modo que, se ocorrer um erro, apenas o fluxo com dados naquele pacote seja bloqueado.

Definido no RFC 9114, o HTTP/3 é suportado pela maioria dos principais navegadores, incluindo o Chromium (e suas variantes, como Chrome e Edge) e o Firefox.