

**HTTP** é um protocolo, uma forma de conversa entre duas máquinas, que permite transferir hiper-texto de um lado a outro. Daí o nome Hyper Text Transport Protcolo.

É o protocolo que te permite comprar passagens de avião pela internet, conversar com amigos pelas redes sociais e assistir mandar videos de gatos para sua familia. É o protocolo por trás da Web. Com ele, é possível que um estudante num café em São Paulo leia um artigo sobre o império mongol que está armazenado em um servidor nos Estados Unidos.

Entender bem o protocolo HTTP pode te ajudar a desenvolver melhores aplicações web e a debugá-las quando as coisas derem errado. Para entender bem o HTTP, vale entender **como o navegador web funciona**:

https://www.youtube.com/watch?v=kDy62zaCHZE

**Recursos, URLs e URIs**

A parte que conhecemos melhor do protocolo HTTP é o endereço HTTP de um site. Por exemplo, quando quero comprar pomada para o meu bigode, abro o navegador e digito, por exemplo, [*http://pomadasparabigode.com*](http://pomadasparabigode.com/). Meu navegador entende a sintaxe e faz uma requisição para um servidor chamado *pomadasparabigode.com*.

O endereço [*http://pomadasparabigode.com*](http://pomadasparabigode.com/) é o que chamamos de URL - **U**niform **R**esource **L**ocator(localizador de recurso uniforme). Ela representa um recurso específico na web. Recursos são coisas que eu quero interagir, como: imagens, páginas, arquivos, e videos. Neste caso, o recurso é a página inicial do site *pomadasparabigode.com*, normalmente um HTML.

Imagine a seguinte URL, que representa umaa fictícia Pomada do Bem: [*http://pomadasparabigode.com/produto/pomada-do-bem-10773*](http://pomadasparabigode.com/produto/pomada-do-bem-10773)

Podemos quebrar ela em 3 partes:

* http, a parte antes de "://" é o que chamamos de **URL Scheme**\_**(esquema da URL). O esquema descreve**como**acessar um recurso em particular. Nesse caso, estamos falando para o navegador usar o**Hypertext Transfer Protocol\*\*, o HTTP. Existem outros esquemas, tais como: https, TCP, FTP, mailto. Tudo que vier depois de "://" é específico do protocolo que estiver sendo utilizado.
* pomadasparabigode.com é o nome do **host**(servidor), que seria o nome do computador que armazena nosso recurso. Nosso navegador vai realizar um processo conhecido como [**DNS**](https://www.alura.com.br/artigos/dns-o-que-e-qual-escolher) Lookup para traduzir *pomadasparabigode.com* em um endereço de rede. E vai enviar a requisição para esse endereço.
* /produto/pomada-do-bem-10773 é o que chamamos de **URL Path** (caminho da URL). O servidor irá identificar qual é o recurso especifico que deve devolver para este caminho quando a requisição chegar. URLs podem apontar para arquivos físicos, como por exemplo [*http://pomadasparabigode.com/foto.jpg*](http://pomadasparabigode.com/foto.jpg) provavelmente para um arquivo físico do servidor, uma foto em formato jpg. Já no caso da URL <http://pomadasparabigode.com/listar-pomadas> provavelmente não aponta para um arquivo físico diretamente.

Nesse caso, provavelmente o que vai acontecer é: uma aplicação desenvolvida em alguma tecnologia como ASP.NET, PHP, Rails, [Java](https://www.alura.com.br/artigos/java) irá tratar a requisição no servidor, fará uma consulta em um banco de dados e o recurso que será devolvido vai ser construído dinamicamente por esta aplicação.

**Números de porta**

http://pomadasparabigode.com:80/listar-pomadas

Esse número 80 representa o número da porta que o servidor está usando para "ouvir" requisições HTTP. 80 é a padrão e é opcional no caso do uso do endereço em um navegador, então normalmente você não vê esse 80 nas URLs. É mais comum especificarmos esta porta quando estamos testando a aplicação em ambiente de homologação/testes. O 443 aparece para o https.

**Query strings**

http://pomadasparabigode.com/busca?nome=pomadalegal

Tudo o que vem depois da ? é o que chamamos de **query string**. Nesse caso ?nome=pomadalegal. Geralmente colocamos na query string informações que serão interpretadas de alguma forma pela aplicação que é executada no servidor. Não existe uma regra formal de como as query strings são montadas, mas a forma mais comum de utilização é através de pares chave-valor, separados por &, como em ?nome=pomadalegal&tipo=2&categoria=bigodesruivos:

http://pomadasparabigode.com/busca?nome=pomadalegal&tipo=2&categoria=bigodesruivos

**Fragmento**

http://pomadasparabigode.com/produto/pomada-especial#descricao

Esse #descricao na URL não é interpretado pelo servidor, mas sim pelo navegador do usuário. Depois de carregar o recurso que é especificado através dessa URL, o navegador irá procurar um elemento com o id descricao na página e irá posicionar a barra de rolagem a partir do início dele, ou seja, onde começa a descrição do produto.

Resumidamente, uma url pode ser quebrada então no seguinte formato:

*[****esquema****]://[****servidor****]:[****porta****]/[****caminho****]?[****querystring****]#[****fragmento****]*

**Media Types**

Um recurso pode ser várias coisas diferentes: imagens, arquivos HTML, XML, videos e muitos mais. Pra que um servidor possa **servir** um recurso e para que o cliente possa consumi-lo apropriadamente, as partes envolvidas(cliente e servidor) têm de ser específicas e precisas quanto ao tipo do recurso. Afinal, não faz o menor sentido que meu navegador tente renderizar como imagem um arquivo XML, certo?

Quando um servidor responde uma requisição HTTP, ele devolve o recurso e o seu tipo - chamado de **Content-Type**(também conhecido como media type). Para especificar tipos de recurso, o HTTP usa um outro protocolo(que inicialmente foi feito para comunicação através de e-mail) chamado **MIME**: **M**ultipurpose **I**nternet **M**ail **E**xtensions.

O content-type tem duas partes: tipo e subtipo. Por exemplo:, um servidor pode devolver uma imagem no formato png. O content-type da resposta viria como image/png. Se fosse um jpg, o content-type seria image/jpg. E se fosse um arquivo html? text/html. E um json? text/json. O navegador olha o Media Type para saber o que fazer com um arquivo.

**Content Type Negotiation**

Como já falamos, um recurso pode ter diferentes representações. Vamos pegar como exemplo a URL que representa o manual de como cuidar do seu bigode:

http://pomadasparabigode.com/comocuidardoseubigode

Este manual poderia, por exemplo, ter diferentes representações no site para diferentes idiomas. Poderia até ser disponibilizado em diferentes formatos: *PDF, DOC, html*. Neste caso, seria o **mesmo recurso**, mas em **formatos diferentes**.

Como o servidor vai saber em que formato deverá enviar o recurso? É aí que entra o mecanismo de [Content Negotiation](https://en.wikipedia.org/wiki/Content_negotiation" \t "_blank) especificado pelo protocolo HTTP: quando um cliente faz uma requisição, ele pode especificar quais Media Types ele aceita. Desta forma, aplicações diferentes podem solicitar o mesmo recurso - mas em formatos diferentes. Se o servidor irá conseguir devolver o recurso naquele formato já é outra conversa, que cabe a quem desenvolver o serviço que roda no servidor :)

**O processo Request-Response**

Se você chega e me pergunta: *Qual a próxima turma de C# na escola Caelum?*

Para que eu possa responder essa pergunta corretamente, são necessárias algumas coisas:

* eu preciso entender a sua pergunta. Se você me perguntar em um idioma que não conheço, provavelmente não conseguirei te dar uma resposta;
* preciso de acesso a algum lugar que conste as próximas turmas da Caelum.

O HTTP funciona mais ou menos desta mesma forma: um cliente precisa de um recurso que está em um outro computador. Então, o cliente você faz uma requisição (**request**) para um servidor usando uma linguagem e vocabulário que espera que o servidor consiga entender. Se o servidor conseguir entender sua requisição e tiver o recurso disponível, ele irá responder com uma resposta(**response**). Caso o servidor entenda a requisição mas não tenha o recurso, provavelmente ele vai responder que não tem. Caso ele não entenda a requisição, você pode não ter resposta.

Request e Response são dois tipos de mensagem diferentes quando falamos de HTTP. A especificação HTTP diz exatamente o que podemos colocar dentro de cada um destes tipos de mensagem para que todos que "falem" o idioma HTTP consigam trocar infomações corretamente.

Uma requisição HTTP tem o seguinte formato:



E uma resposta HTTP:



Então resumidamente, não existe nada mágico quando você digita um endereço no navegador: ele abre uma conexão com o servidor em questão e envia para ele um monte de **texto** seguindo regrinhas especificadas pelo protocolo :)

E ai curtiu? Fique atento para o próximo post! E se quiser saber ainda mais sobre HTTP, temos [um curso no Alura](https://www.alura.com.br/curso-online-fundamentos-http) que fala especificamente sobre isso :)

Temos [um curso sobre HTTP](https://www.alura.com.br/curso-online-fundamentos-http) que te ensina o protocolo HTTP por baixo dos panos.

Mais? Leia sobre as [diferenças dos métodos HTTP](https://www.alura.com.br/artigos/diferencas-entre-get-e-post), em especial GET e POST. Também aqui na Alura um artigo sobre a [diferença entre HTTP e HTTPS](https://www.alura.com.br/artigos/qual-e-diferenca-entre-http-e-https).