Revisão - Protocolo HTTP

Neste material, vamos revisar os principais conceitos sobre a Internet e o Protocolo HTTP

O que é Internet?

A Internet é uma rede global de computadores. Imagine um sistema postal que entrega e que recebe pacotes em velocidades extremamente rápidas.

Este sistema utiliza os Protocolos TCP/IP e HTTP para se comunicar. Qualquer indivíduo com permissão é capaz de obter informações de outro computador.

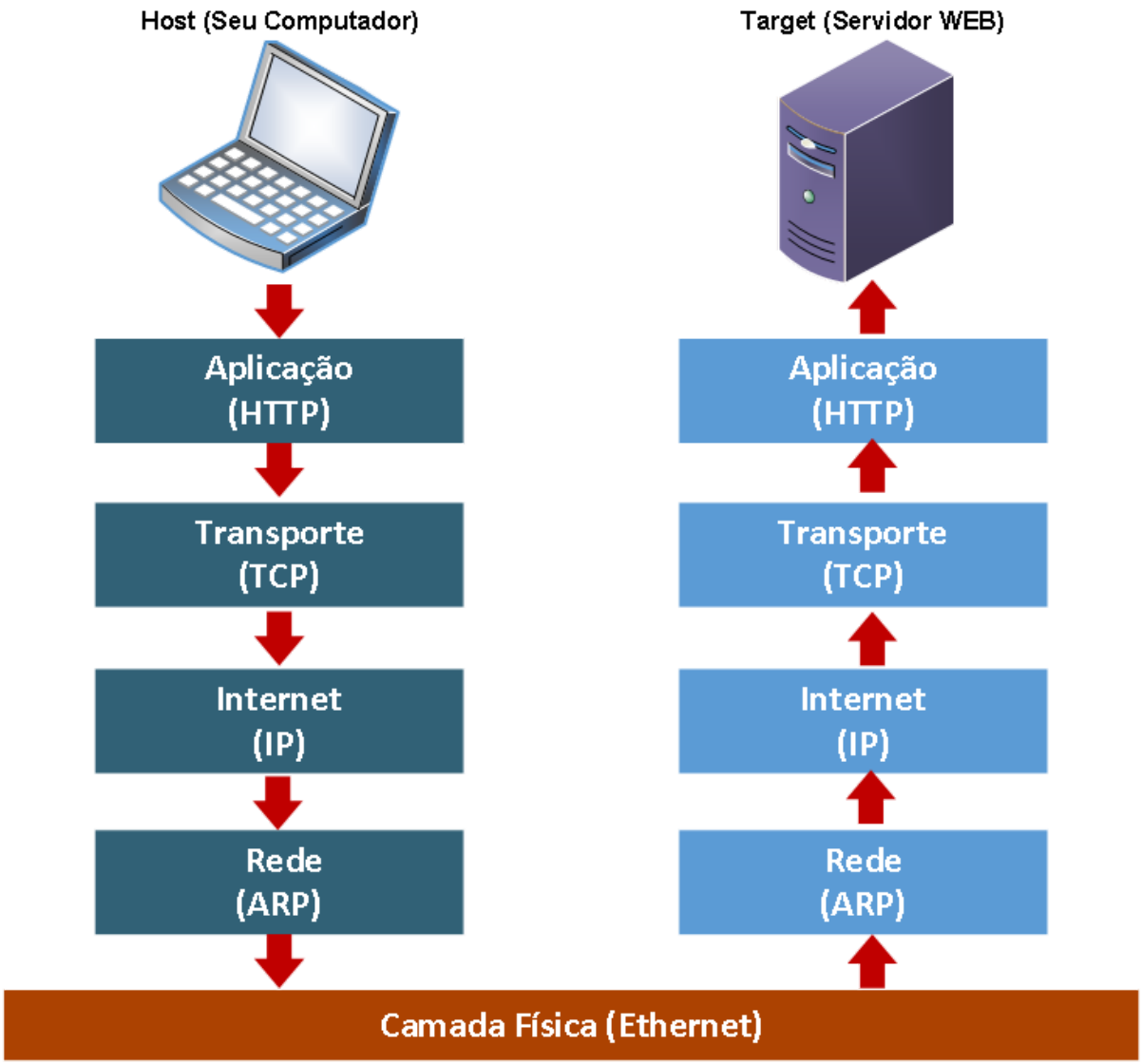
Protocolo

Um **protocolo** é um sistema de regras que define como o dado é trafegado dentro ou entre computadores. Comunicações entre dispositivos requer que estes concordem com o formato do dado que estiver sendo trafegado. O conjunto de regras que define esse formato é chamado de protocolo.

Protocolo TCP/IP

O TCP/IP é um conjunto de protocolos de comunicação. O nome vem de dois protocolos **TCP** (*Transmission Control Protocol*) e o **IP** (*Internet Protocol*). Ele tem por objetivo padronizar todas as comunicações de rede, principalmente as comunicações na web. O TCP/IP é o Protocolo que define a Internet.

Modelo TCP/IP



O Modelo TCP/IP

O modelo TCP/IP possui 4 camadas:

Camada 4: Aplicação

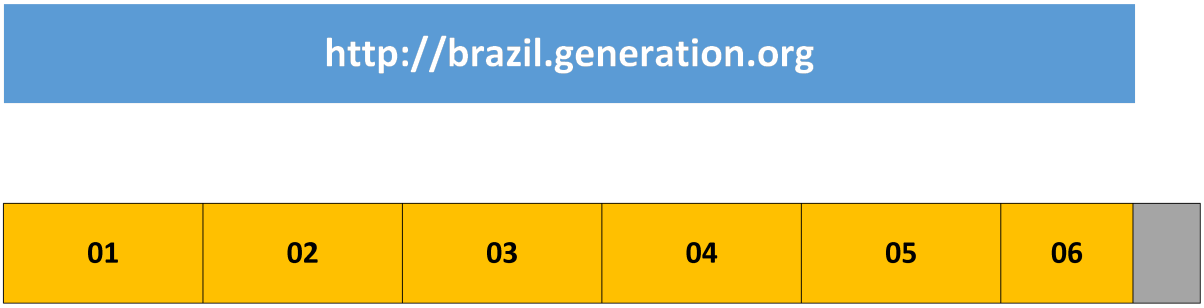
Nesta camada encontra-se todos os protocolos de serviço que efetuam a comunicação direta com o software para identificar o tipo de requisição que está sendo realizada. O protocolo mais popular é o HTTP.

Camada 3: Transporte

Esta camada é Responsável pela comunicação entre hosts (Clientes e Servidores) envolvidos. Ela tem como função principal verificar se o pacote de dados alcançou seu destino e se os dados nele contidos chegaram de maneira integra. O protocolo mais popular é o TCP.

Pacote

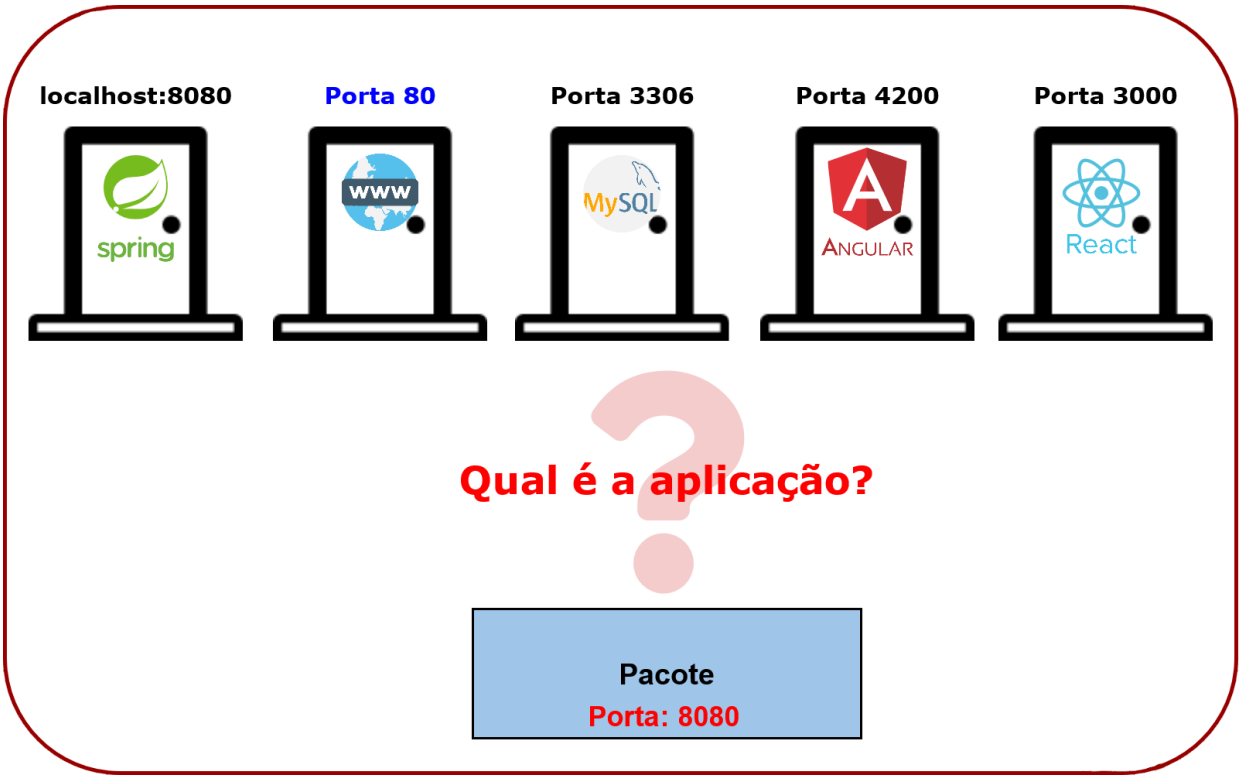
Pacote é o formato no qual os dados são enviados do servidor para o cliente e vice e versa. Basicamente, quando os dados são enviados pela web, eles são enviados como milhares de pequenos blocos, para agilizar e simplificar o tráfico de dados pela Internet.



Em azul temos o grande bloco de dados e logo abaixo, na cor amarela, o bloco dividido em pacotes menores que serão identificados e enviados para o seu destino

Portas

A camada de transporte utiliza portas lógicas para garantir que a aplicação que iniciou a conversação encontrará no seu destino a aplicação desejada.
Essas portas lógicas são canais virtuais, geralmente definidos pelo Sistema Operacional, que se abrem conforme o tipo de aplicação que se está executando.
O **HTTP** utiliza as **portas 8080** (Localmente, ou seja, no seu Computador) e a **porta 80** (Internet).



No Exemplo acima, o pacote que chegou na Camada de Aplicação contém a informação **Porta: 8080**. Observe que neste cenário o Spring está utilizando a porta 8080, logo o TCP irá redirecionar o pacote para a nossa aplicação Spring. Se a Porta, por exemplo, estivesse com o valor 3306 o TCP redirecionaria para o MySQL.

 **Lista de Portas TCP:**
https://pt.wikipedia.org/wiki/Lista_de_portas_dos_protocolos_TCP_e_UDP (Acessado em 05/2021)

Camada 2: Internet

Nesta camada encontramos os endereços IP de origem e destino de uma conexão. O IP também é conhecido como o endereço lógico (pode ser alterado).

Camada 1: Rede

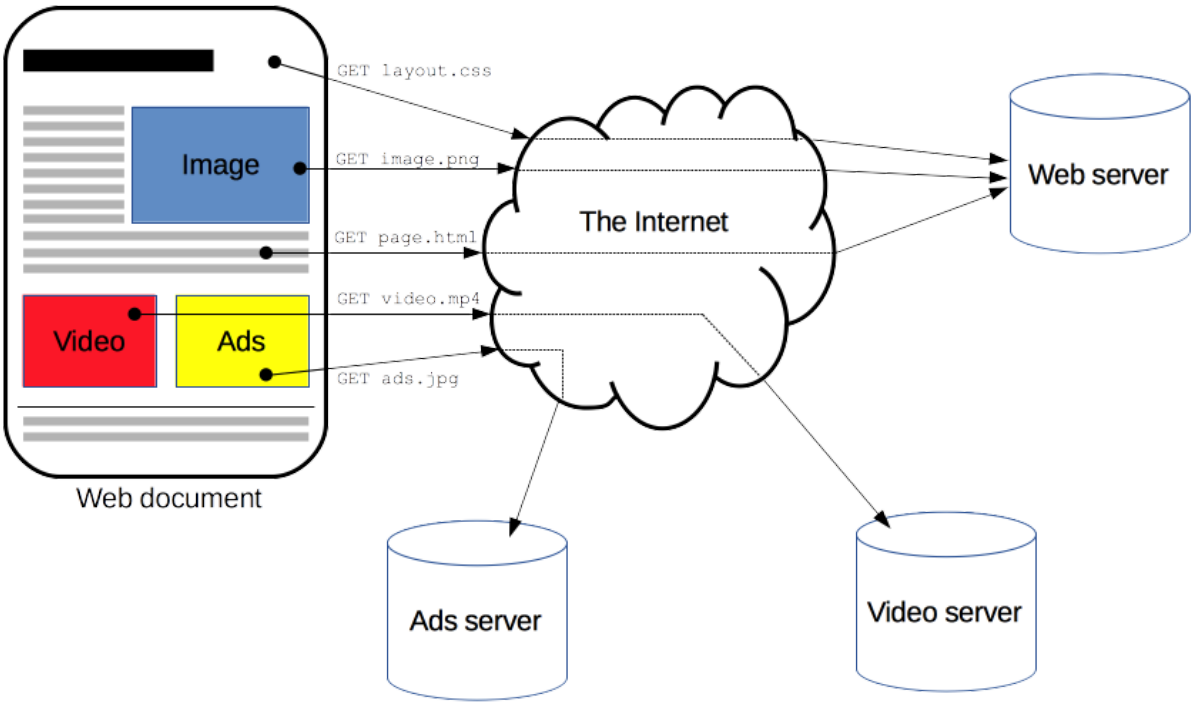
Nesta camada encontramos a conexão física da rede pela qual o pacote trafega. Cada equipamento conectado na rede carrega consigo a identidade do hardware que deu origem ao envio do pacote, armazenando o seu endereço MAC (Media Access Control). O MAC também é conhecido como Endereço físico (não pode ser alterado).

Como funciona o Protocolo TCP/IP?

 Assista ao vídeo *Warriors of the NET - IP for Peace* no link:
https://youtu.be/lqcp3k8DgGw

Protocolo HTTP

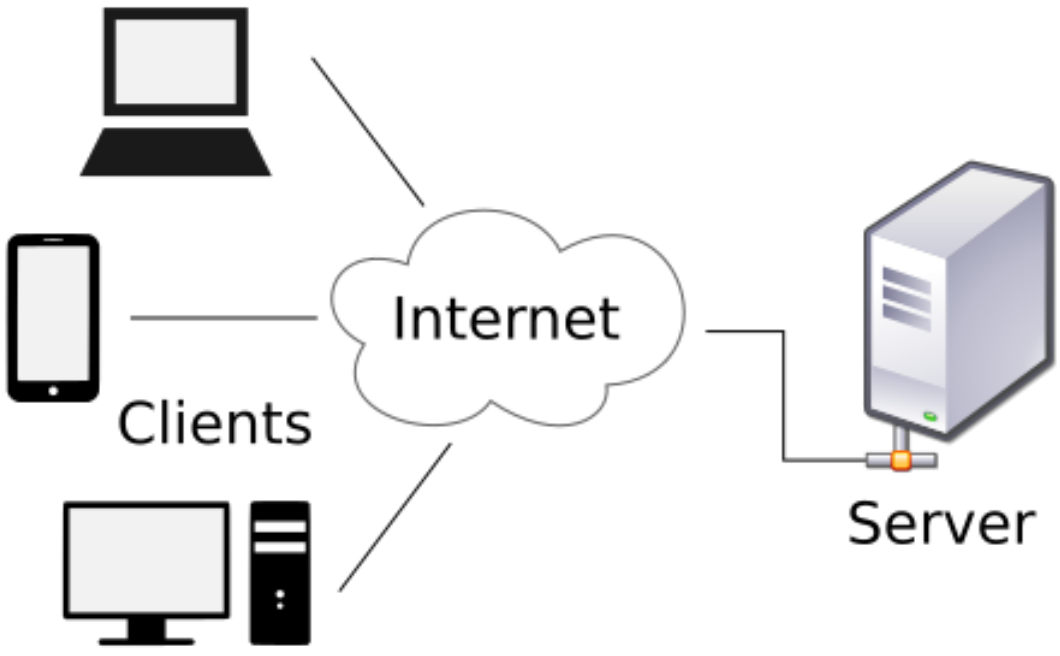
O **HTTP (Hypertext Transfer Protocol / Protocolo de Transferência de Hipertexto)** é um protocolo que permite a obtenção de recursos, como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, 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.



Página WEB composta por vários elementos

O HTTP também possui uma outra versão chamada de **HTTPS (Hyper Text Transfer Protocol Secure / Protocolo de Transferência de Hipertexto Seguro)**, que se trata de uma versão do protocolo criptografado, geralmente utilizado em transações bancárias. Ao utilizar o protocolo HTTPS, um cadeado 🔒 é exibido no Navegador para indicar que a conexão é segura.

O modelo Cliente Servidor



Os Clientes acessando o servidor via Internet

Cliente

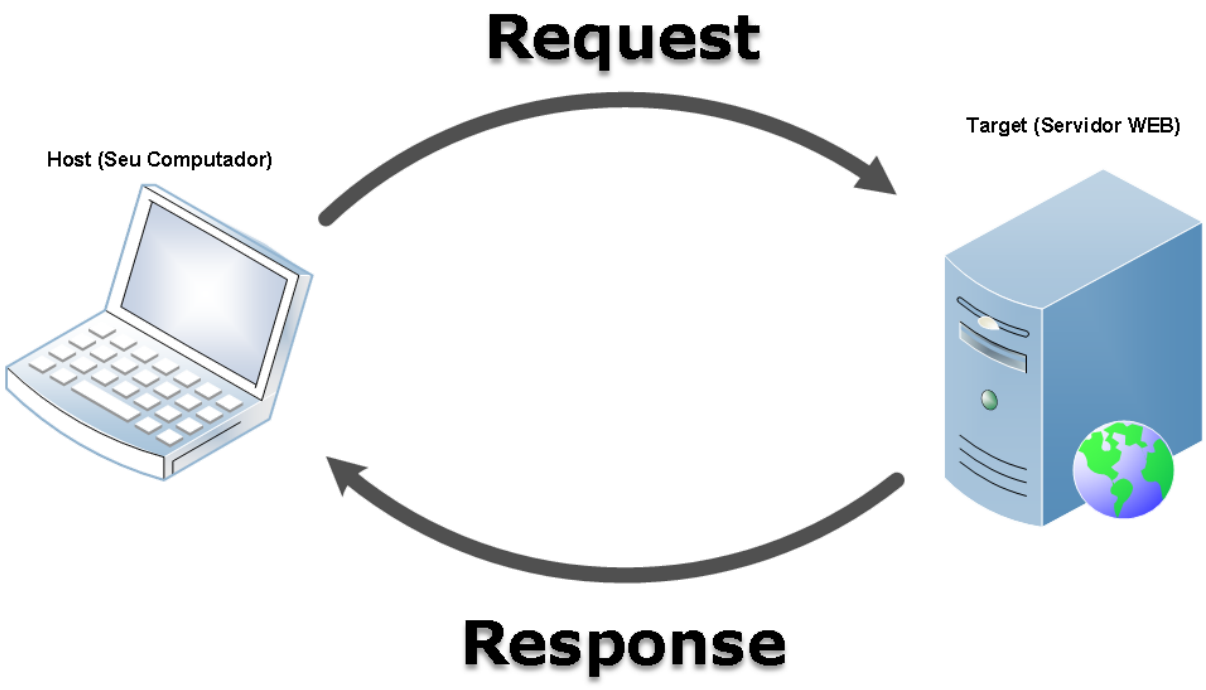
O Cliente é qualquer ferramenta que age em nome do usuário. Essa função é predominantemente realizada pelo navegador Web, salvo algumas poucas exceções, são programas usados por pessoas Desenvolvedoras Web para testar as suas aplicações (Exemplo: Postman).

Servidor

O servidor é a "máquina" que serve o documento requisitado pelo usuário através do endereço www. Um servidor se apresenta virtualmente apenas como uma máquina, porém ele pode ser uma coleção de servidores dividindo a carga ou um programa complexo que acessa outros servidores gerando toda ou parte do documento solicitado.

Fluxo HTTP

O HTTP funciona como um protocolo **Request-Response** (Solicitação-Resposta), entre um Cliente e um Servidor.



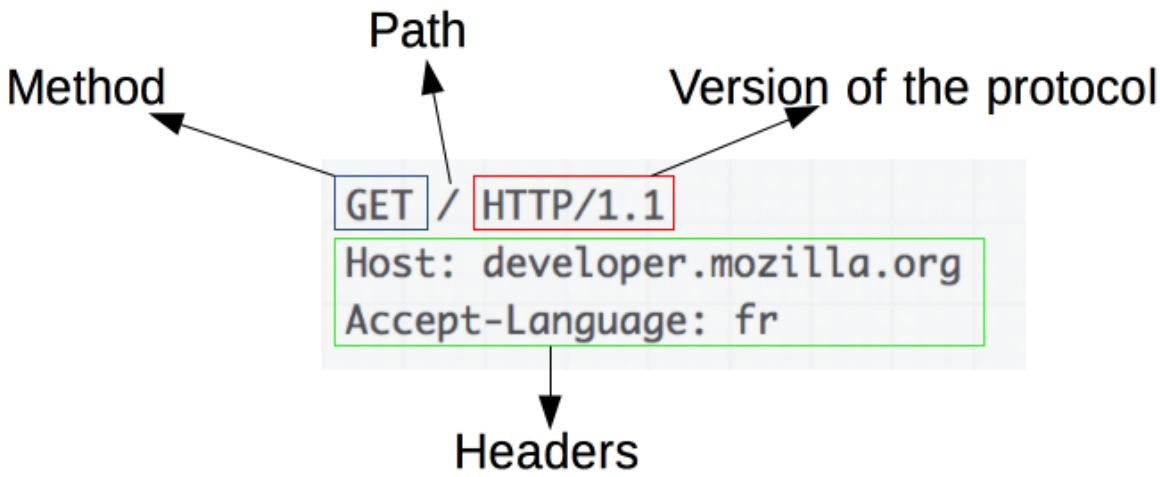
Fluxo de Funcionamento HTTP

Exemplo: Um cliente (Navegador da Internet) envia uma **Request** HTTP ao servidor e o servidor retorna uma **Response** ao cliente.

Para o cliente se comunicar com um servidor, ele realiza os seguintes passos:

1. Abre uma conexão TCP, que será usada para enviar uma requisição, ou várias, e receber uma resposta.
2. Envia uma mensagem HTTP (Request).
3. Lê a resposta do servidor (Response).
4. Fecha ou reutiliza a conexão para requisições futuras.

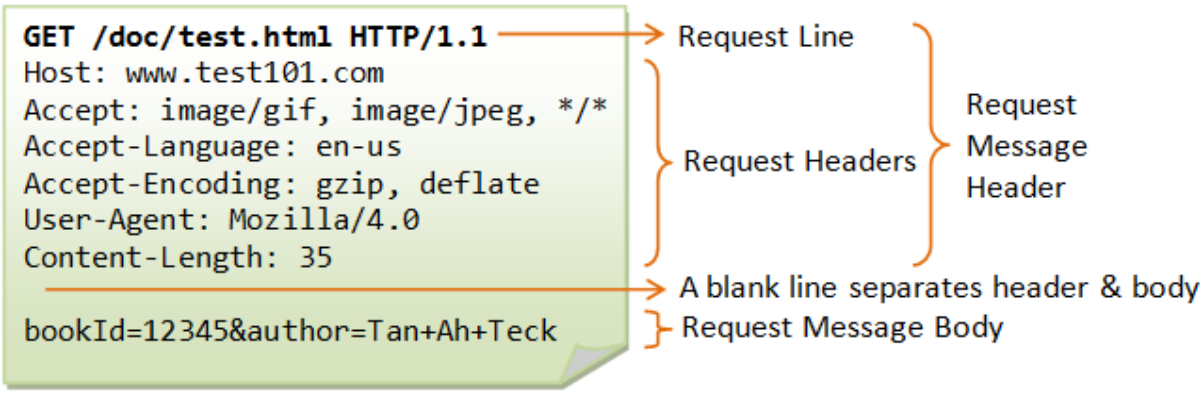
Request



Request Message Header HTTP

As requisições consistem dos seguintes elementos:

1. Um **método HTTP**, geralmente é um verbo como GET, POST, DELETE e PUT.
2. O **caminho do recurso**, ou seja a **URL** do recurso sem os elementos que são de contexto, como o protocolo (http://). A
3. **versão do protocolo HTTP**.
4. Cabeçalhos opcionais que contém informações adicionais para os servidores.
5. Body (corpo) para alguns métodos como o POST e o PUT, que enviam dados para o recurso solicitado dentro da requisição.



Request HTTP

Métodos HTTP (Verbos)

O protocolo HTTP define um conjunto de **métodos de requisição** responsáveis por indicar a ação a ser executada para um dado recurso. Estes métodos são referenciados como **Verbos HTTP**.

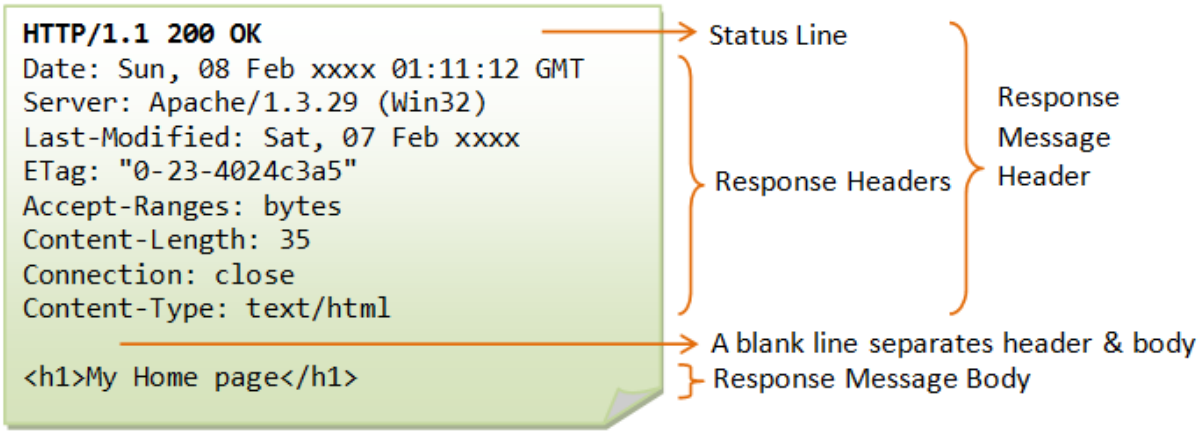
Principais Verbos HTTP

Verbo	**Descrição**
GET	O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados. Exemplo: Consultar um registro no Banco de Dados.
POST	O método POST é utilizado para submeter uma entidade a um recurso específico para o servidor. Exemplo: Inserir um novo registro no Banco de Dados.
PUT	O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição. Exemplo: Atualizar os atributos de um registro no Banco de Dados.
DELETE	O método DELETE remove um recurso específico. Exemplo: Apagar um registro no Banco de Dados.

HTTP Methods Request:
https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods(Acessado em 05/2021)

Response

A Response (resposta) nada mais é do que a resposta que o servidor envia ao cliente. Essa resposta pode conter os dados que realmente o cliente esperava receber ou uma resposta informando que alguma coisa deu errado.



Response HTTP

O protocolo HTTP também define um conjunto de **Códigos de Status de respostas** responsáveis por indicar se a ação executada para um dado recurso foi executada corretamente ou não. Estes métodos são comumente referenciados como **HTTP Status Code**.
As respostas são agrupadas em cinco classes:

1. Respostas de informação (`100` - `199`),
2. Respostas de sucesso (`200` - `299`),
3. Redirecionamentos (`300` - `399`)
4. Erros do cliente (`400` - `499`)
5. Erros do servidor (`500` - `599`).

Principais HTTP Status Code

✅ Respostas de Sucesso

Código	Resposta	Descrição
200	*OK*	Estas requisição foi bem sucedida. O significado do sucesso varia de acordo com o método HTTP.
201	*CREATED*	A requisição foi bem sucedida e um novo recurso foi criado como resultado. Esta é uma típica resposta enviada após uma requisição POST.
204	*NO CONTENT*	Não há conteúdo para enviar para esta solicitação, mas os cabeçalhos podem ser úteis. O user-agent pode atualizar seus cabeçalhos em cache para este recurso com os novos. Essa resposta é muito comum após uma Requisição DELETE.

☹️ Respostas de Erro do Cliente

Código	Resposta	Descrição
400	*BAD REQUEST*	Essa resposta significa que o servidor não entendeu a requisição pois está com uma sintaxe inválida.
401	*UNAUTHORIZED*	Embora o padrão HTTP especifique "unauthorized", semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada. Esta resposta é muito comum quando habilitamos a Segurança na API (Login, Senha e Token).
403	*FORBIDDEN*	O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.
404	*NOT FOUND*	O servidor não pode encontrar o recurso solicitado. Este código de resposta acontece com frequência na web.

Respostas de Erro do Servidor

Código	Resposta	Descrição
500	*INTERNAL SERVER ERROR*	O servidor encontrou uma situação com a qual não sabe lidar.
501	*NOT IMPLEMETED*	O método da requisição não é suportado pelo servidor e não pode ser manipulado.
502	*BAD GATEWAY*	O servidor, ao trabalhar como um gateway a fim de obter uma resposta necessária para manipular a requisição, obteve uma resposta inválida.
503	*SERVICE UNAVAILABLE*	O servidor não está pronto para a requisição. Causas comuns são um servidor em manutenção ou sobrecarregado.

HTTP Status Code:
https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status (Acessado em 05/2021)

WEB Site e WEB Application

WEB Site

Um Web Site é uma coleção de páginas da web e conteúdo relacionado que é identificado por um nome de domínio comum e publicado em pelo menos um servidor da web.

WEB Application

Um aplicativo da web é um software aplicativo executado em um servidor da web, ao contrário dos programas de software baseados em computador que são executados localmente no sistema operacional (SO) do dispositivo.

Desenvolvimento: Back-end e Front-end

Back-end

Apesar de mais abstrato, o conceito de Back-end também é simples de entender: os dados publicados em uma rede social, como fotos e vídeos, por meio da interface do usuário precisam ser armazenados em algum local para poderem ser acessados posteriormente.

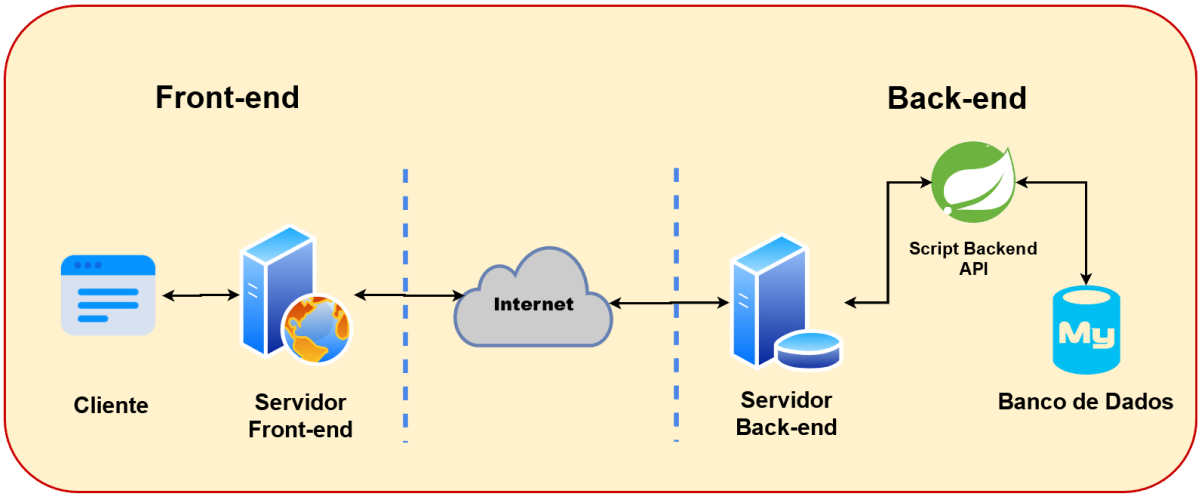
Esse envio não é feito diretamente do front-end para o banco de dados da rede social. Ao invés disso, existe uma parte da aplicação que é responsável por receber essas informações, fazer operações específicas — postagens, filtros, exclusões, verificações de segurança e validações e somente após isso, armazenar no banco de dados. Esse é o back-end.

As atribuições de um desenvolvedor back-end não incluem a criação de páginas bonitas ou responsivas, como acontece com profissionais front-end. Em vez disso, eles são responsáveis por implementar arquiteturas robustas, que se comuniquem com o banco de dados (MySQL) e que garantam a segurança dos dados enviados pelo usuário. Para isso utiliza linguagens como o Java e o Framework Spring.

Front-end

O Front-end é, de forma sucinta, toda parte visual de um site, a parte com a qual o usuário interage diretamente. O profissional responsável por trabalhar nessa área de um projeto desenvolve o código para a interface gráfica, normalmente por meio de linguagens.

Um desenvolvedor front-end normalmente trabalha criando toda a parte visual dos sites por meio de linguagens de marcação(HTML), estilização (CSS) e programação (JavaScript e TypeScript), além de bibliotecas (Bootstrap) e frameworks como o Angular.



No exemplo acima, temos uma visão geral da integração entre o Front-end e do Back-end da aplicação.

O que é Full Stack?

Full Stack está relacionado a pessoa desenvolvedora. Espera-se de uma pessoa desenvolvedora **Full Stack** que ela seja capacitada para **lidar, ao mesmo tempo, com o Front-end e o Back-end de uma aplicação**, mesmo que os dois lados utilizem tecnologias e linguagens diferentes.