

Linguagens de Marcação e Scripts

Prof. Aníbal Cavalcante de Oliveira

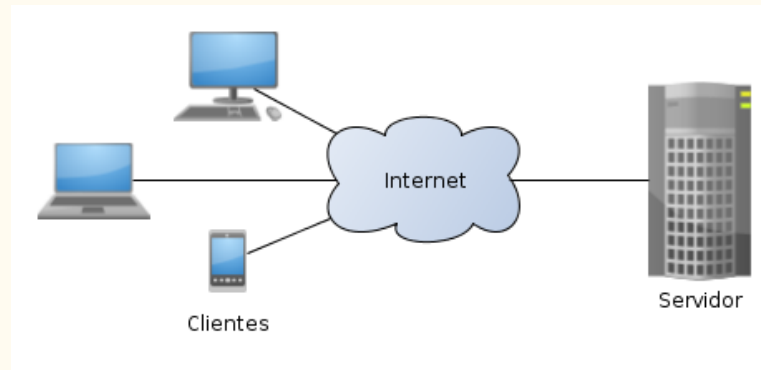
UFC - QXD0164 - 2019.2

Agenda - Aula 18

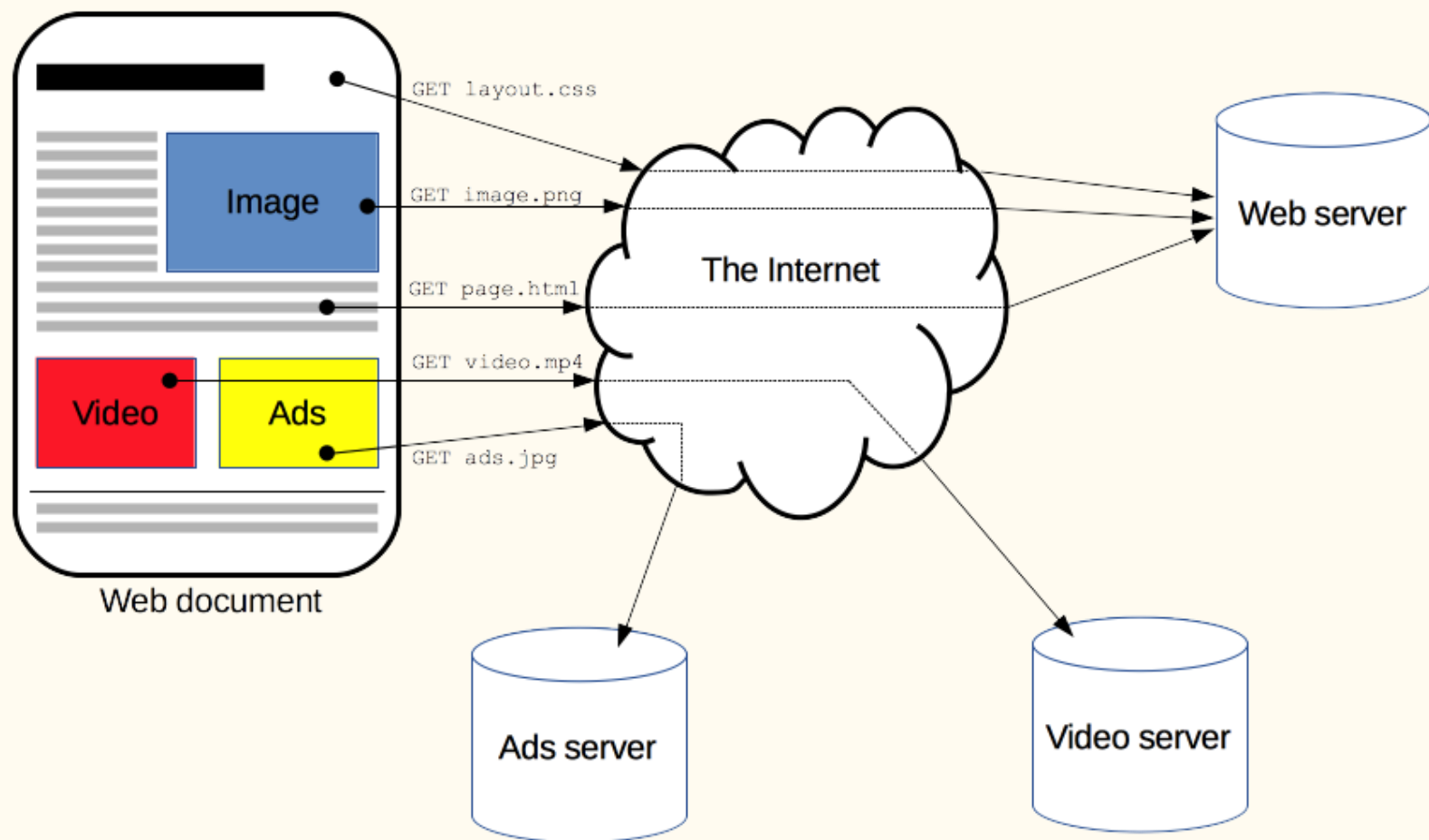
- Introdução ao protocolo HTTP

O protocolo HTTP

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

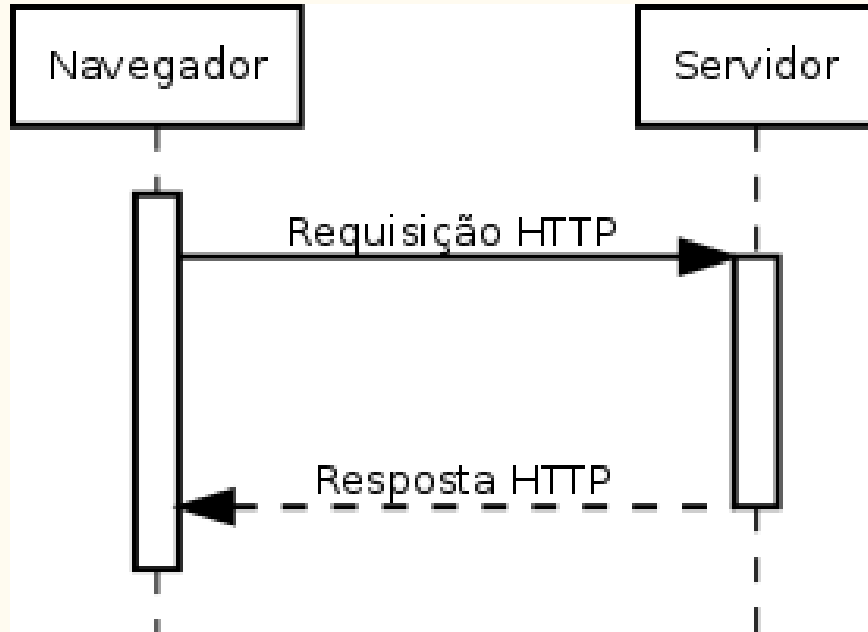


Relembrando o protocolo HTTP e a Internet com sua Arquitetura Distribuída



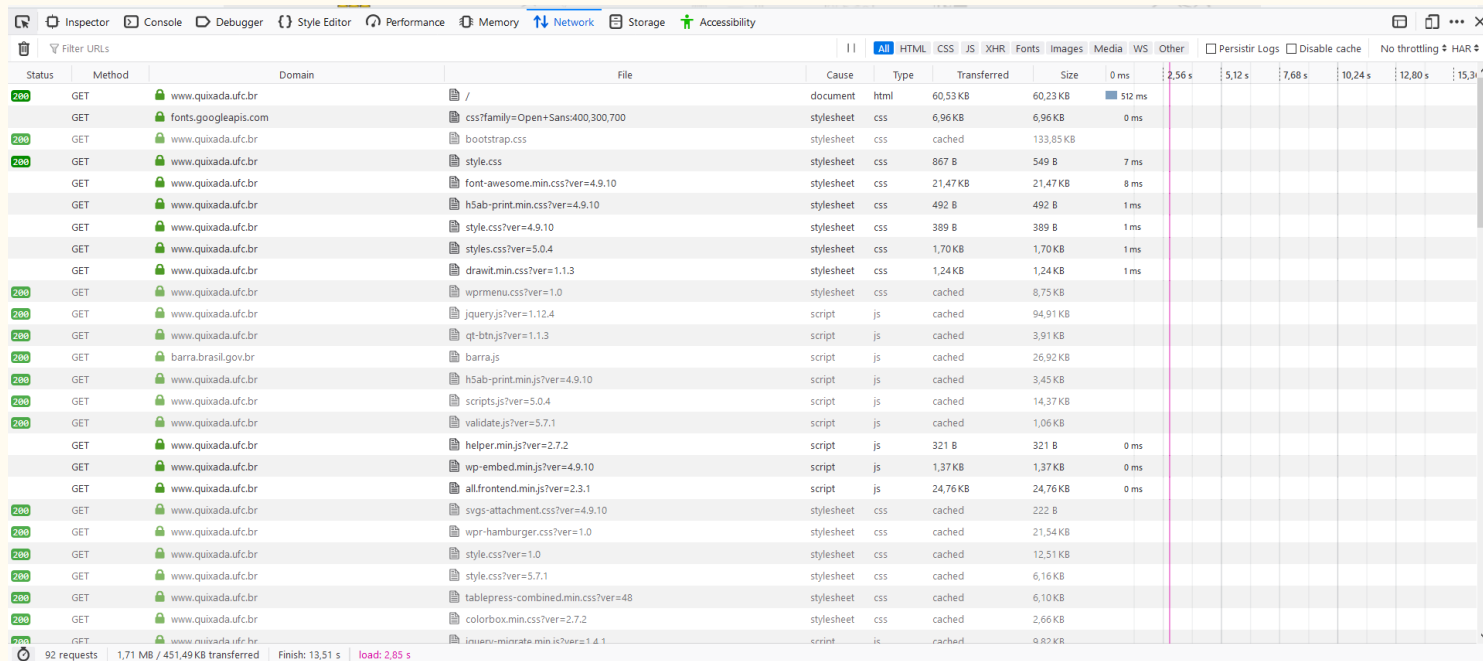
0 ciclo de requisições e respostas (requests e responses)

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**).



Exemplo de várias requisições e respostas no carregamento da página do campus

O browser vai requisitando e recebendo cada recurso da página Web e tenta realizar a renderização da página web por completo, com HTML, CSS, JS, Imagens e etc...



The screenshot shows the Chrome DevTools Network tab with 92 requests listed. The requests are categorized by status (200, 304, 404) and type (HTML, CSS, JS, XHR, Fonts, Images, Media, WS, Other). The table includes columns for Status, Method, Domain, File, Cause, Type, Transferred, Size, and Time. A vertical pink line is drawn at approximately 2.85 seconds, indicating the total load time. The bottom summary bar shows 92 requests, 1.71 MB / 451.49 KB transferred, and a total load time of 2.85 s.

Status	Method	Domain	File	Cause	Type	Transferred	Size	Time
200	GET	www.quixada.ufc.br	/	document	html	60,53 KB	60,23 KB	512 ms
200	GET	fonts.googleapis.com	css?family=Open+Sans:400,300,700	stylesheet	css	6,96 KB	6,96 KB	0 ms
200	GET	www.quixada.ufc.br	bootstrap.css	stylesheet	css	cached	133,85 KB	0 ms
200	GET	www.quixada.ufc.br	style.css	stylesheet	css	867 B	549 B	7 ms
200	GET	www.quixada.ufc.br	font-awesome.min.css?ver=4.9.10	stylesheet	css	21,47 KB	21,47 KB	8 ms
200	GET	www.quixada.ufc.br	h5ab-print.min.css?ver=4.9.10	stylesheet	css	492 B	492 B	1 ms
200	GET	www.quixada.ufc.br	style.css?ver=4.9.10	stylesheet	css	389 B	389 B	1 ms
200	GET	www.quixada.ufc.br	styles.css?ver=5.0.4	stylesheet	css	1,70 KB	1,70 KB	1 ms
200	GET	www.quixada.ufc.br	drawit.min.css?ver=1.1.3	stylesheet	css	1,24 KB	1,24 KB	1 ms
200	GET	www.quixada.ufc.br	wprmenu.css?ver=1.0	stylesheet	css	cached	8,75 KB	0 ms
200	GET	www.quixada.ufc.br	jquery.js?ver=1.12.4	script	js	cached	94,91 KB	0 ms
200	GET	www.quixada.ufc.br	qt-btn.js?ver=1.1.3	script	js	cached	3,91 KB	0 ms
200	GET	barra.brasil.gov.br	barra.js	script	js	cached	26,92 KB	0 ms
200	GET	www.quixada.ufc.br	h5ab-print.min.js?ver=4.9.10	script	js	cached	3,45 KB	0 ms
200	GET	www.quixada.ufc.br	scripts.js?ver=5.0.4	script	js	cached	14,37 KB	0 ms
200	GET	www.quixada.ufc.br	validate.js?ver=5.7.1	script	js	cached	1,06 KB	0 ms
200	GET	www.quixada.ufc.br	helper.min.js?ver=2.7.2	script	js	321 B	321 B	0 ms
200	GET	www.quixada.ufc.br	wp-embed.min.js?ver=4.9.10	script	js	1,37 KB	1,37 KB	0 ms
200	GET	www.quixada.ufc.br	all.frontend.min.js?ver=2.3.1	script	js	24,76 KB	24,76 KB	0 ms
200	GET	www.quixada.ufc.br	svgs-attachment.css?ver=4.9.10	stylesheet	css	cached	222 B	0 ms
200	GET	www.quixada.ufc.br	wpr-hamburger.css?ver=1.0	stylesheet	css	cached	21,54 KB	0 ms
200	GET	www.quixada.ufc.br	style.css?ver=1.0	stylesheet	css	cached	12,51 KB	0 ms
200	GET	www.quixada.ufc.br	style.css?ver=5.7.1	stylesheet	css	cached	6,16 KB	0 ms
200	GET	www.quixada.ufc.br	tablepress-combined.min.css?ver=48	stylesheet	css	cached	6,10 KB	0 ms
200	GET	www.quixada.ufc.br	colorbox.min.css?ver=2.7.2	stylesheet	css	cached	2,66 KB	0 ms
200	GET	www.quixada.ufc.br	jquery-migrate.min.js?ver=1.4.1	script	js	cached	9,87 KB	0 ms

92 requests | 1,71 MB / 451,49 KB transferred | Finish: 13,51 s | load: 2,85 s

Cliente: o agente-usuário (Realiza a requisição)

Um cliente é qualquer ferramenta que age em nome do usuário. Essa função é predominantemente realizada pelo navegador Web;

O navegador é sempre a entidade que inicia as requisições. Nunca é o servidor.

Para mostrar uma página Web, o navegador envia uma requisição para buscar o documento HTML da página, ou qualquer outra informação que vamos chamar de **recurso (resource)**.

Ele então realiza uma análise sintática desse arquivo, e busca requisições adicionais correspondentes a códigos JS, CSS, Imagens e Vídeos, para apresentação da página.

Depois o navegador interpreta esses recursos para mostrar ao usuário o documento completo, a página Web.

Servidor: envia uma resposta para cada requisição feita

Do outro lado do canal de comunicação está o servidor que serve o **recurso** 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.

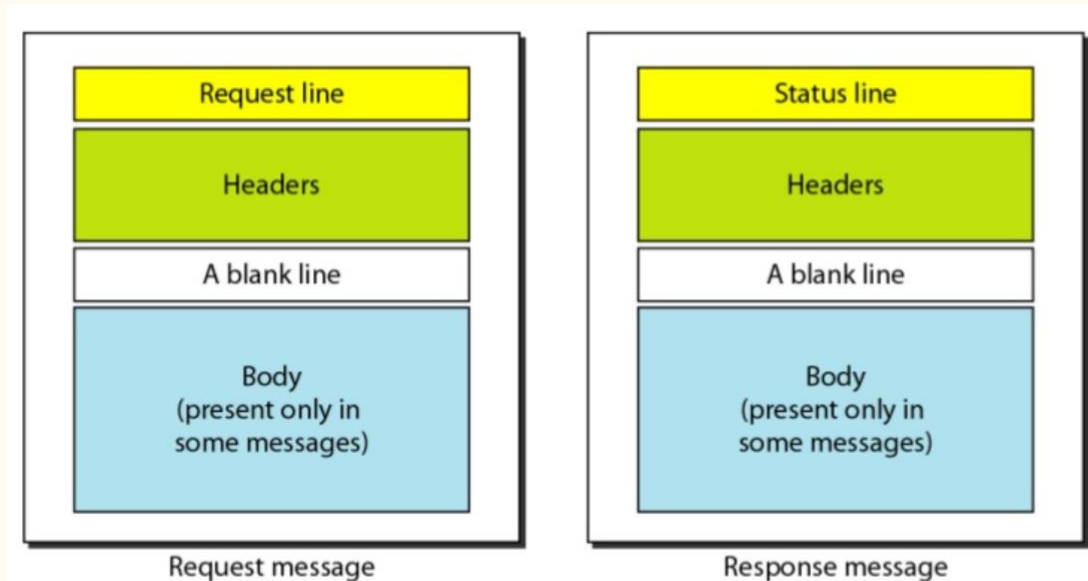
Fluxo HTTP

Resumidamente, quando o cliente quer comunicar com um servidor realiza os seguintes passos:

- 1 - Abre uma conexão (TCP ou UDP) e envia uma mensagem HTTP requisitando um recurso do servidor;
- 2 - Aguarda a resposta do servidor, sobre o recurso solicitado;
- 3 - O cliente lê a resposta do servidor;
- 4 - Finalmente, o cliente fecha ou reutiliza a mesma conexão para requisições futuras, como css, js, imagens e etc...

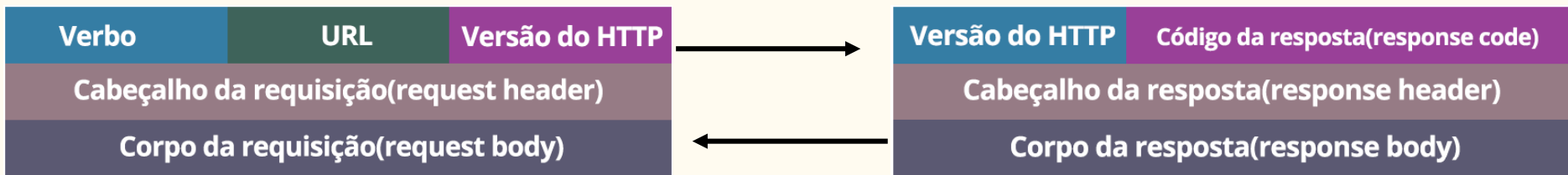
Mensagens HTTP

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



Transação HTTP

Para cada requisição realizado pelo cliente, o servidor envia uma resposta.



Requisições

As requisições consistem dos seguintes elementos:

- 1 - Um método HTTP, como: **GET**, **POST**, **DELETE** ou **PUT**, que define qual operação o cliente quer fazer.
- 2 - O caminho do recurso a ser buscado; a URL do recurso com a porta porta TCP (porta padrão é a 80)
- 3 - A versão do protocolo HTTP.
- 4 - Cabeçalhos opcionais que contém informações adicionais para os servidores.
- 5 - Um corpo de dados, que para alguns métodos como POST, contém informações que serão enviadas para o servidor.

Mensagens HTTP: Requisição

Exemplo de uma Requisição:

GET /index.html HTTP/1.1

Request Line

Date: Thu, 20 May 2004 21:12:55 GMT

Connection: close

General Headers

Host: www.myfavoriteamazingsite.com

From: joeblow@somewebsitesomewhere.com

Accept: text/html, text/plain

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Request Headers

Entity Headers

**HTTP
Request**

Message Body

Respostas

As respostas consistem dos seguintes elementos:

1 - Uma linha inicial com a versão do protocolo e o código da resposta, por exemplo: **HTTP/1.1 200 OK**.

2 - Cabeçalho com informações gerais: tipo de dado, data da resposta, codificação, tamanho do arquivo do corpo, etc...

3 - Corpo da resposta:

3.1 - Um único arquivo com tamanho conhecido ou não.

3.2 - Várias partes contendo diferentes seções de informação. (raro)

Mensagens HTTP: Resposta

Exemplo de uma Resposta:

HTTP/1.1 200 OK	Status Line	HTTP Response
Date: Thu, 20 May 2004 21:12:58 GMT	General Headers	
Connection: close		
Server: Apache/1.3.27	Response Headers	
Accept-Ranges: bytes		
Content-Type: text/html	Entity Headers	
Content-Length: 170		
Last-Modified: Tue, 18 May 2004 10:14:49 GMT		
<html>		Message Body
<head>		
<title>Welcome to the Amazing Site!</title>		
</head>		
<body>		
<p>This site is under construction. Please come back later. Sorry!</p>		
</body>		
</html>		

Respostas

Os principais códigos das resposta HTTP.

Código	Frase	Descrição
200	OK	A solicitação foi bem sucedida
400	Bad Request	Erro de sintaxe na solicitação
401	Unauthorized	A solicitação não tem autorização suficiente para ser executada.
403	Forbidden	Serviço negado
404	Not Found	O Documento não foi encontrado
500	Not implemented	Há um erro, como um crash, por exemplo, no servidor
503	Service unavaivable	O serviço está temporariamente indisponível mas poderá ser solicitado no futuro

Métodos de Requisição HTTP

As requisições HTTP são realizadas através métodos ou verbos que definem qual operação o cliente quer fazer.

Método GET: É o método mais simples do protocolo HTTP, e seu principal trabalho é pedir ao servidor um determinado recurso.

Ele é mais utilizado quando enviamos informações na própria URL da requisição.

Em formulários, por exemplo, os dados vão todos na URL, visível pela barra de navegação do browser.

Vamos criar um formulário de login com o método **GET**.

Métodos de Requisição HTTP

Exemplo de transação GET, onde a informação solicitada vai na própria URL:



Métodos de Requisição HTTP

O método **POST** é uma solicitação mais poderosa, e é utilizado quando queremos criar um novo recurso no servidor, ou enviar alguma informação sigilosa. Quando usamos **POST**, os dados vão no corpo da requisição e não na URL.

Outra diferença é que o método **GET** está restrito 256 bytes, em geral, pelo servidor. Se digitarmos uma URL muito extensa, ele pode não funcionar.

Utilizamos **POST** para fazer uploads de arquivos para o servidor por exemplo.

Altere o formulário de login para o método **POST**.

Métodos de Requisição HTTP

Outros métodos HTTP.

PUT

Requisita que um recurso seja "guardado" na URI fornecida. Se o recurso já existir, ele deve ser atualizado. Se não existir, pode ser criado.

DELETE

Exclui o recurso especificado.

TRACE

Devolve a mesma requisição que for enviada veja se houve mudança e/ou adições feitas por servidores intermediários.

OPTIONS

Retorna os métodos HTTP suportados pelo servidor para a URL especificada.

PATCH

Serve para atualizar **partes** de um recurso, e não o recurso todo.

CONNECT

Converte a requisição de conexão para um túnel TCP/IP transparente, geralmente para facilitar a comunicação criptografada com SSL (HTTPS) através de um proxy HTTP não criptografado.

Métodos de Requisição HTTP

Vamos testar um serviço na nuvem chamado: <https://randomuser.me/api/>

Vamos utilizar o programa chamado POSTMAN para testá-la:

<https://www.getpostman.com/downloads/>