

# Desenvolvimento de Software para Web

Benevaldo Pereira Gonçalves

[benevaldo.goncalves@ifam.edu.br](mailto:benevaldo.goncalves@ifam.edu.br)

(92) 98158-9743



# Aviso:

Em caso de ausência, justificar a falta pelo e-mail ***institucional do Aluno*** para o e-mail ***benevaldo.goncalves.ifam.edu.br*** durante o prazo de **48h**, a partir da data da ausência.

# Programação Web

- Desenvolvimento web é a área da tecnologia voltada à construção de sites, aplicativos, softwares integrados com bancos de dados e quaisquer outras ferramentas que, de certa forma, são construídos para o ambiente da internet (web).
- Apresenta uma implementação que utiliza o modelo ***Cliente/Servidor***, através do ***protocolo HTTP***.
- Existe uma divisão nas atividades voltadas ao desenvolvimento para web, a codificação do lado ***cliente (front-end)*** e a do lado do ***servidor (back-end)***.

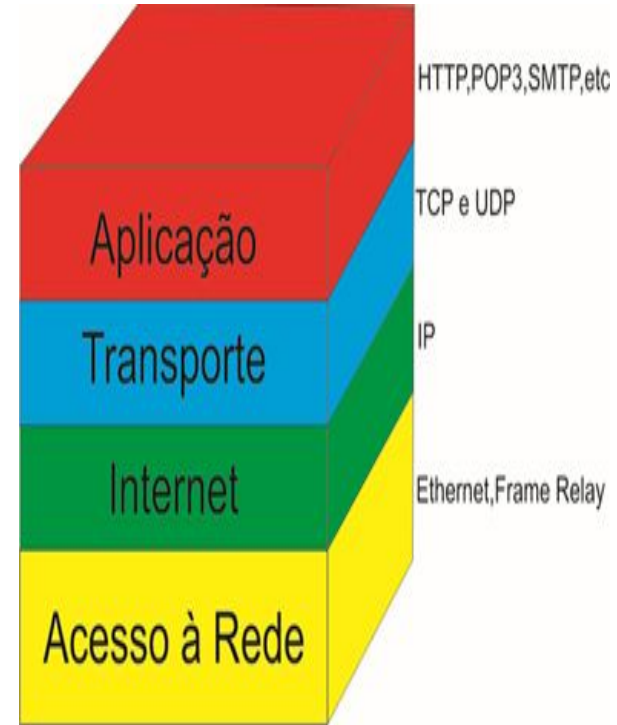
# Protocolo

É uma convenção que controla e possibilita uma conexão, comunicação, transferência de dados entre dois sistemas computacionais. De maneira simples, um protocolo pode ser definido como "as regras que governam" a sintaxe, semântica e sincronização da comunicação. Os protocolos podem ser implementados pelo hardware, software ou por uma combinação dos dois.

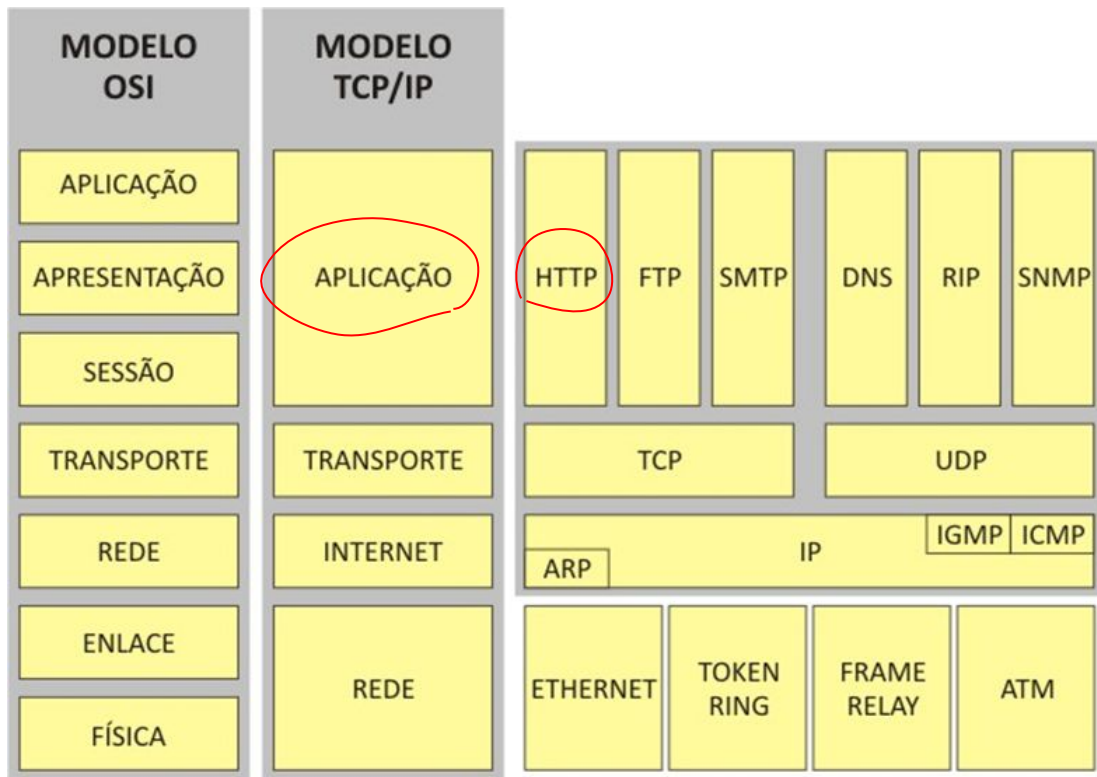


# Modelo TCP/IP

O TCP/IP (também chamado de pilha de protocolos TCP/IP) é um conjunto de protocolos de comunicação entre computadores em rede. Seu nome vem de dois protocolos: o TCP (Transmission Control Protocol - Protocolo de Controle de Transmissão) e o IP (Internet Protocol - Protocolo de Internet, ou ainda, protocolo de interconexão). O conjunto de protocolos pode ser visto como um modelo de camadas, onde cada camada é responsável por um grupo de tarefas, fornecendo um conjunto de serviços bem definidos para o protocolo da camada superior. As camadas mais altas, estão logicamente mais perto do usuário (chamada **camada de aplicação**) e lidam com dados mais abstratos, confiando em protocolos de camadas mais baixas para tarefas de menor nível de abstração.



# Fundamentos de Redes: Modelos OSI e TCP/IP



# Camada de Transporte: TCP: Porta

- Os protocolos na camada de transporte podem resolver problemas como confiabilidade (o dado alcançou seu destino?) e integridade (os dados chegaram na ordem correta?). O TCP também determina para qual aplicação um dado qualquer é destinado pela definição da sua Porta.
- Existem 65.536 portas TCP, numeradas de 1 a 65536. Cada porta pode ser usada por um programa ou serviço diferente, de forma que em teoria poderíamos ter até 65536 serviços diferentes ativos simultaneamente num mesmo servidor, com um único endereço IP válido.
- As portas TCP mais usadas são as portas de 1 a 1024, que são reservadas para serviços mais conhecidos e utilizados, como servidores Web, FTP, servidores de e-mail, compartilhamento de arquivos, etc. A porta 80 por exemplo é reservada para uso de servidores Web, enquanto a porta 21 é a porta padrão para servidores FTP. Além do endereço IP, qualquer pacote que circula na Internet precisa conter também a porta TCP a que se destina. É isso que faz com que um pacote chegue até o servidor Web e não ao servidor FTP instalado na mesma máquina.

Serviço	Porta
http	80
ftp	20 e 21
telnet	23
dhcp	67
dns	53
snmp	161 e 162
nfs	2049
smb	137, 138, 139 e 445
smtp	25
pop3	110

# Camada de Rede: IP

- Com o advento da internet novas funcionalidades foram adicionadas nesta camada, especialmente para a obtenção de dados da rede de origem e da rede de destino. Isso geralmente envolve rotear o pacote através de redes distintas que se relacionam através da internet.
- Na suíte de protocolos para a internet, o IP executa a tarefa básica de levar pacotes de dados da origem para o destino. O protocolo IP pode transmitir dados para diferentes protocolos de níveis mais altos, esses protocolos são identificados por um único número de protocolo IP.

Um endereço IPv4 (notação decimal com pontos)

**172 . 16 . 254 . 1**

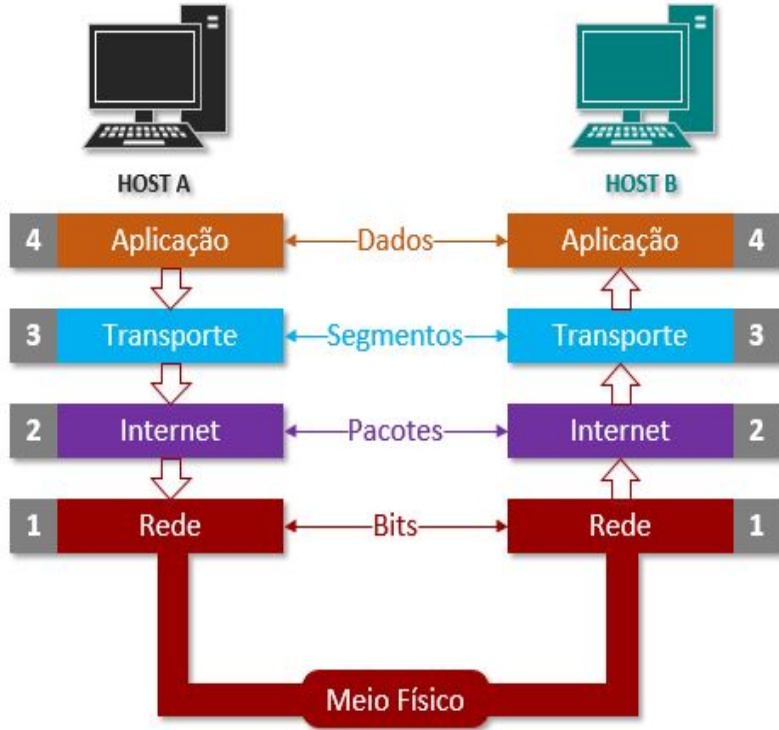
↓ ↓ ↓ ↓  
10101100 . 00010000 . 11111110 . 00000001

Um byte=Oito bits

Trinta e dois bits (4 x 8) ou 4 bytes

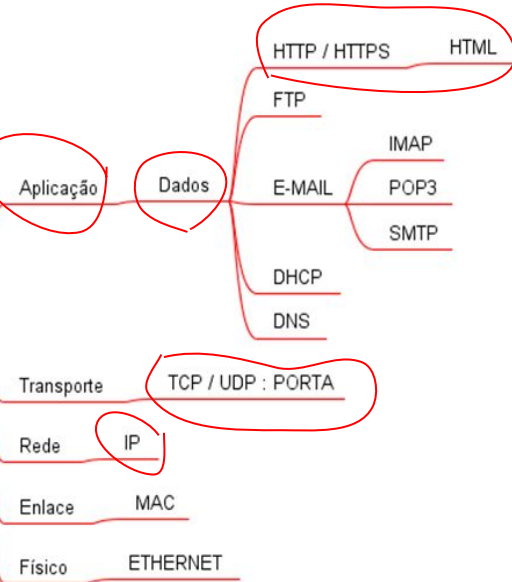


# Modelo Internet TCP/IP



## Modelo TCP/IP

Camadas



# Modelo Cliente/Servidor

- O termo **Cliente/Servidor** refere-se ao método de distribuição de aplicações computacionais através de muitas plataformas. Tipicamente essas aplicações estão divididas entre um provedor de acesso e uma central de dados e numerosos clientes contendo uma interface gráfica para usuário acessar e manipular dados.
- **Cliente/Servidor** geralmente refere-se a um modelo onde dois ou mais computadores interagem de modo que um oferece os serviços aos outros. Este modelo permite aos usuários acessarem informações e serviços de qualquer lugar.
- **Cliente/Servidor** é um modelo computacional que envolve **requisições** de serviços de **clientes** para **servidores**

# Funções do Cliente

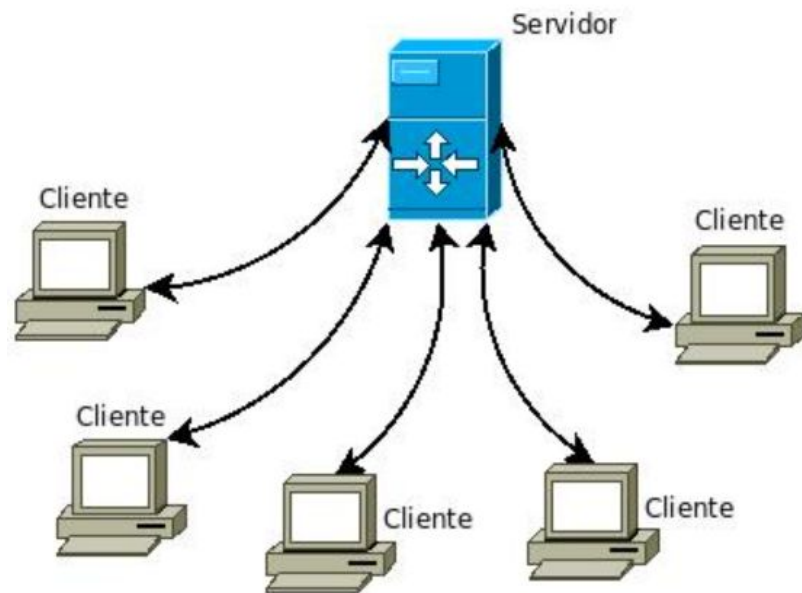
Cliente, também denominado de “front-end” ou “WorkStation”, é um processo que interage com o usuário através de uma interface gráfica ou não, permitindo consultas ou comandos para recuperação de dados. Além disso, apresenta algumas características distintas:

- É o processo ativo na relação Cliente/Servidor.
- Inicia e termina as conversações com os Servidores, solicitando recursos (site, mídia, serviços, etc...)
- Não se comunica com outros Clientes.
- Torna a rede transparente ao usuário.

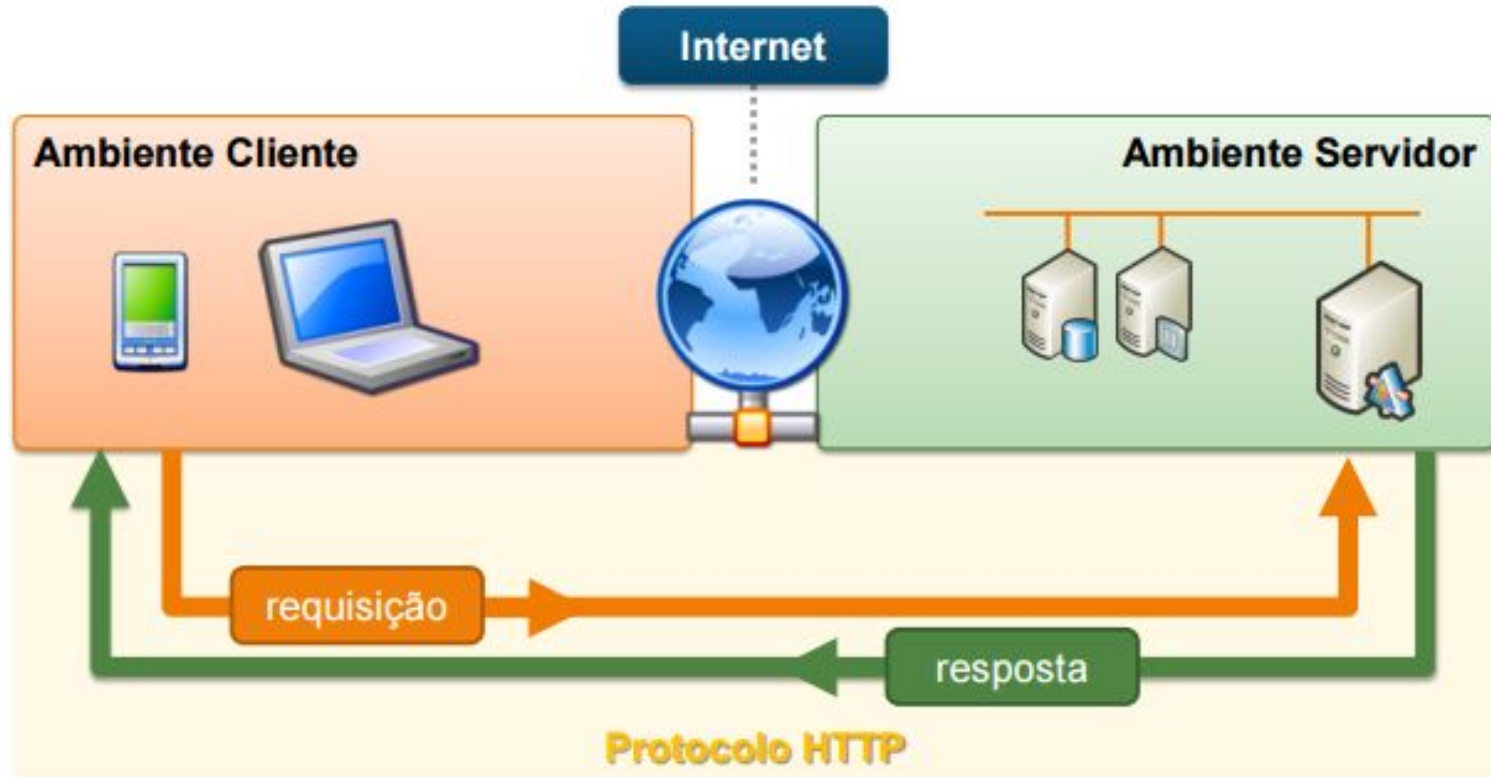
# Funções do Servidor

Também denominado **Servidor** ou “back-end”, fornece um determinado recurso e/ou serviço que fica disponível para todo Cliente que o necessita. A natureza e escopo do serviço são definidos pelo objetivo da aplicação Cliente/Servidor. Além disso, ele apresenta ainda algumas propriedades distintas:

- É o processo reativo na relação Cliente/Servidor.
- Possui uma execução contínua.
- Recebe e responde às solicitações dos Clientes.
- Não se comunica com outros Servidores enquanto estiver fazendo o papel de Servidor.
- Atende a diversos Clientes simultaneamente.

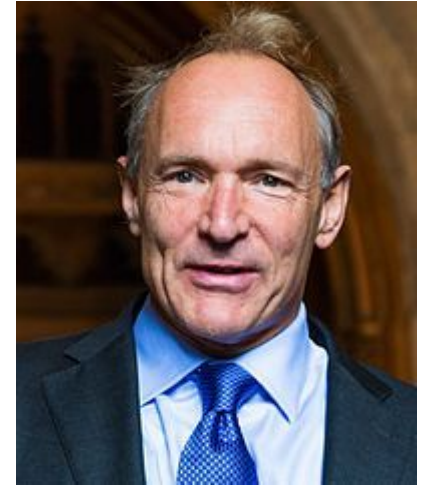


# Modelo Cliente/Servidor para ambiente Web



# Timothy John Berners-Lee

- Timothy John Berners-Lee, nasceu em Londres, 8 de junho de 1955. É um físico britânico, cientista da computação e professor do MIT. É o criador da **World Wide Web**, tendo feito a primeira proposta para sua criação a 12 de março de 1989. Em 25 de dezembro de 1990, com a ajuda de Robert Cailliau e um jovem estudante do CERN, implementou a **primeira comunicação bem-sucedida** entre um **cliente HTTP e o servidor HTTP** através da **internet**.
- Berners-Lee é o diretor do **World Wide Web Consortium (W3C)**, que supervisiona o desenvolvimento continuado da web. Também é o fundador da **World Wide Web Foundation** e é um pesquisador sênior e titular e fundador da **3Com** no Laboratório de Inteligência Artificial e Ciência da Computação do MIT.



Tim Berners-Lee

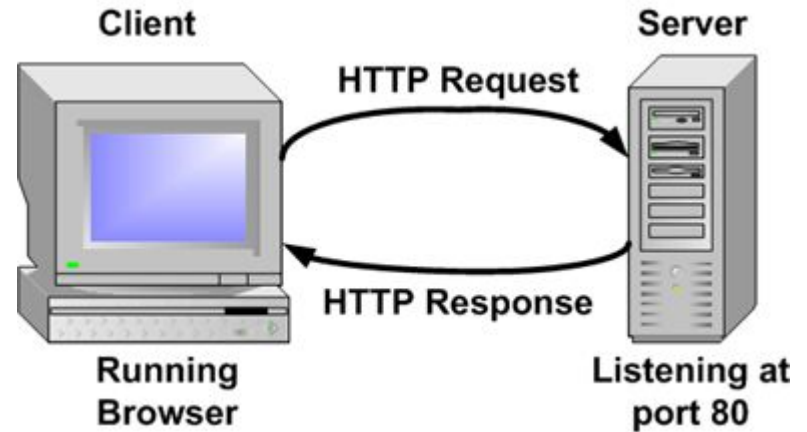
# Protocolo HTTP

- **H**yper**T**ext **T**ransfer **P**rotocol (HTTP) é o protocolo de comunicações usado para acessar a World Wide Web e por todos os aplicativos da web de hoje. É um protocolo simples que foi originalmente desenvolvido para a recuperação de recursos estáticos baseado em texto. Desde então, foi ampliado e alavancado em várias maneiras para apoiar as complexas aplicações que são agora comuns. Trata-se de um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores, principalmente na World Wide Web (Internet).
- Para que a transferência de dados na Internet seja realizada, o protocolo HTTP necessita estar agregado a outros dois protocolos de rede: **TCP** (Transmission Control Protocol) e **IP** (Internet Protocol), necessário para a conexão entre computadores ***clientes/servidores***

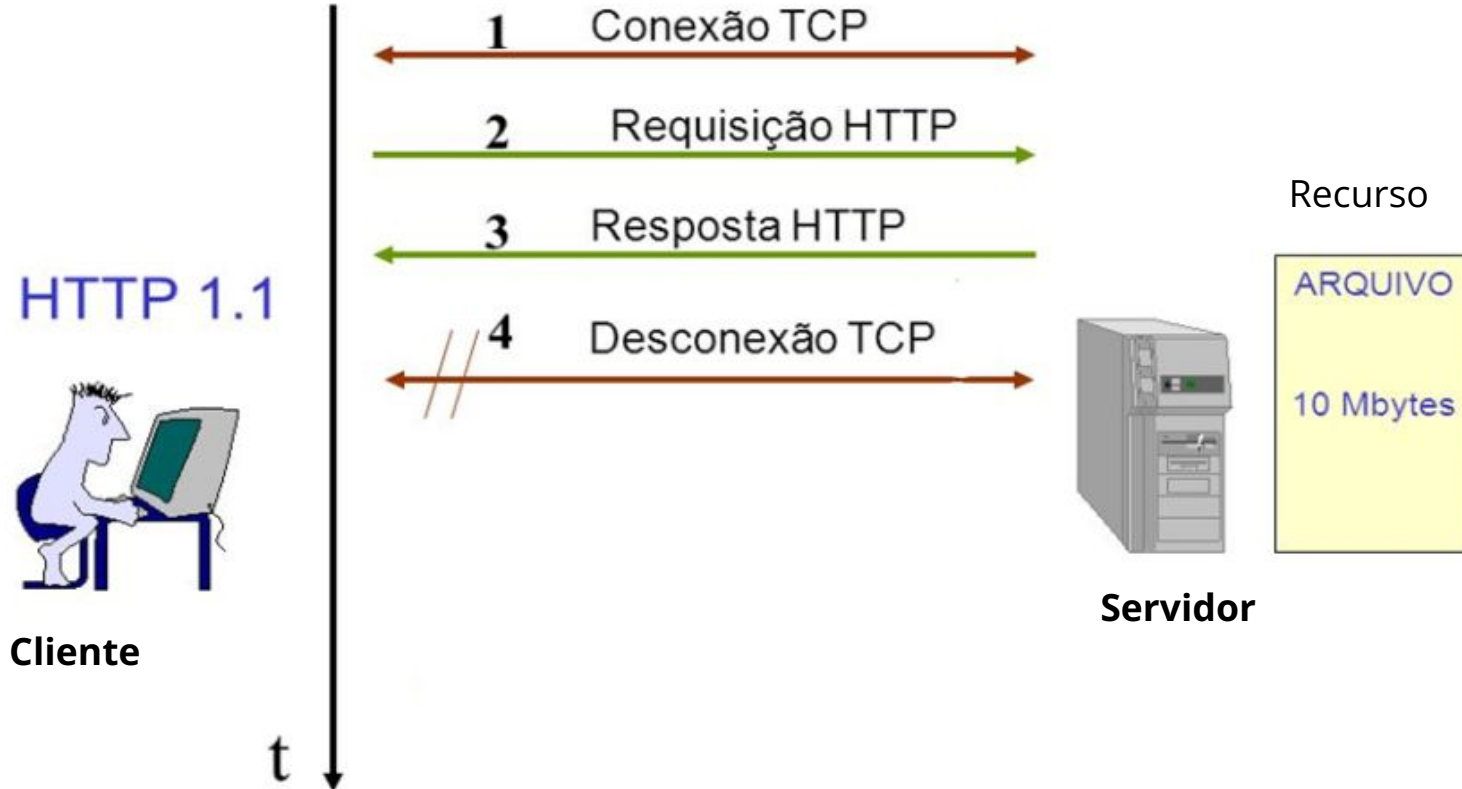


# Protocolo HTTP

O protocolo HTTP é baseado em **requisições** e **respostas** entre **clientes** e **servidores**. O cliente — que fará a requisição; também é conhecido como user agent — solicita um determinado recurso ou serviço, enviando um pacote de informações (**requisição**). O servidor recebe estas informações e envia uma **resposta**, que pode ser um recurso ou serviço solicitado pelo cliente.



# HTTP Funcionamento



# Método HTTP

Quando você vai fazer uma requisição, é preciso que você especifique qual o método será utilizado. Os métodos HTTP, também conhecidos como **verbos**, identificam qual a **ação** que deve ser executada em um determinado recurso. Os principais métodos são:

GET

- Buscar recursos
- Cache

POST

- Criar um novo recurso

PUT

- Atualizar um recurso existente

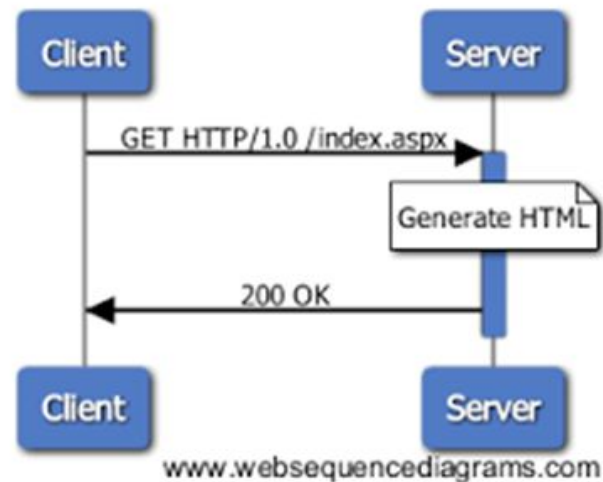
DELETE

- Remover um recurso

# Status HTTP

Toda **requisição** recebe um código de resposta conhecido como **status**. Com o status é possível saber se uma operação foi realizada com sucesso (**200**), se foi movida e agora existe em outro lugar (**301**) ou se não existe mais (**404**). Existem diversos código de Status:

- 1xx - Informacional;
- 2xx - Solicitação feita com sucesso
- 3xx - O Cliente é direcionado para um recurso diferente;
- 4xx - A solicitação contém um erro de algum tipo;
- 5xx - O Servidor encontrou um erro ao realizar o consulta.

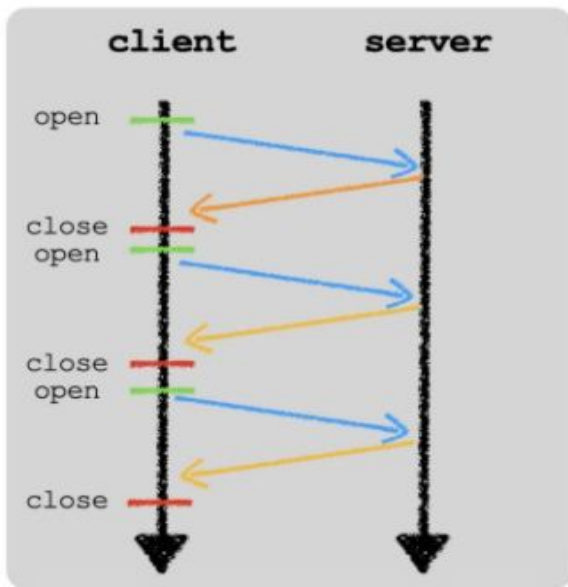


# Protocolo HTTP: Estrutura

Versão (Version)	Tamanho do Cabeçalho (IHL)	Tipo de Serviço (ToS)	Tamanho Total (Total Length)	
Identificação (Identification)			Flags	Deslocamento do Fragmento (Fragment Offset)
Tempo de Vida (TTL)		Protocolo (Protocol)	Soma de verificação do Cabeçalho (Checksum)	
Endereço de Origem (Source Address)				
Endereço de Destino (Destination Address)				
Opções + Complemento (Options + Padding)				

# Protocolo HTTP: Característica

HTTP usa um modelo baseado na **mensagem** em que um **cliente** envia uma mensagem de **solicitação** e o **servidor** retorna uma mensagem de **resposta**. O protocolo é essencialmente **sem conexão** (connectionless): embora HTTP usa o protocolo TCP (stateful - com estado) como o seu mecanismo de transporte, cada troca de solicitação e resposta é uma transação **autônoma** e pode usar uma **conexão TCP diferente**.



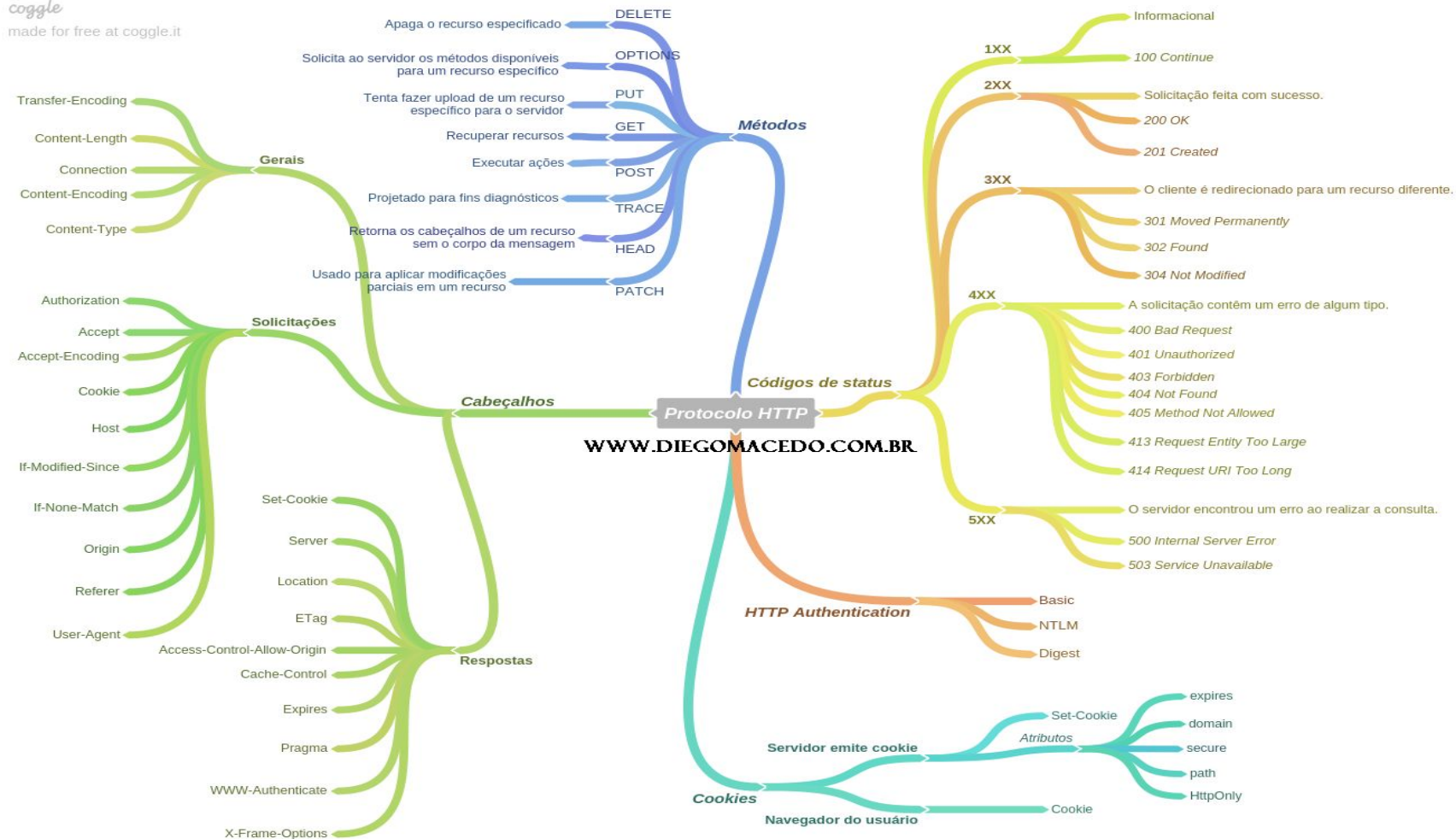
# Protocolo HTTP: Solicitação

```
GET /auth/488/YourDetails.ashx?uid=129 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, application/x-shockwaveflash,
*/*
Referer: https://mdsec.net/auth/488/Home.ashx
Accept-Language: en-GB
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET CLR 1.1.4322)
Accept-Encoding: gzip, deflate
Host: mdsec.net
Connection: Keep-Alive
Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476
```

# Protocolo HTTP: Resposta

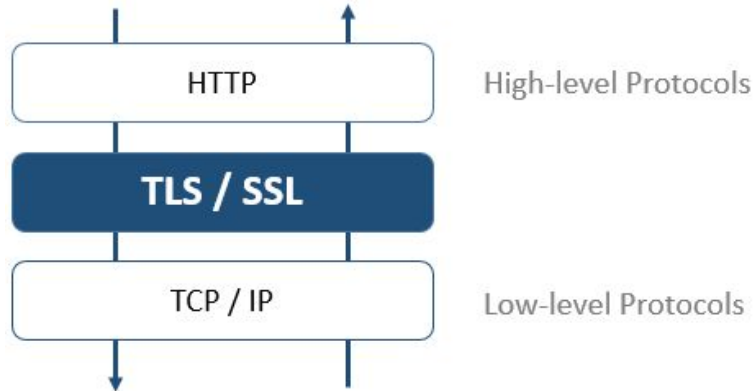
```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 09:23:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1067
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml" ><head><title>Your details</title>
...
```





# HTTPS

O protocolo HTTP usa TCP simples como seu mecanismo de transporte, que não é criptografado e, portanto, pode ser interceptado por um atacante que esteja adequadamente posicionado na rede. Enquanto o **HTTPS** é, essencialmente, o mesmo protocolo da camada de aplicação como HTTP, mas é encapsulado sobre o mecanismo de transporte seguro, **Secure Sockets Layer** (SSL). Isso protege a privacidade e integridade dos dados que passa sobre a rede, reduzindo as possibilidades de ataques de interceptação não-invasivas. Solicitações e respostas HTTP funcionam exatamente da mesma maneira, independentemente de SSL estar sendo usado para o transporte. Secure Sockets Layer (SSL) tem sido estritamente substituída por **Transport Layer Security (TLS)**, mas ultimamente ainda é referido utilizando o nome antigo.



# HTTPS

**Tudo o que é enviado em HTTP  
é texto simples:**



101101 Nome de usuário: Jill 110101 01101 Senha: BobsCat 100010101  
Conta Nº: 76573624394 01100 101010



**Os mesmos dados com criptografia HTTPS:**



W7fh&688d/6534900fHtGklm63QPIcebgr8o70BX76LP219f+jK763  
0enyHtYmLSy65HvLpOzzEAdnMg71ngp8nbmdJH98iqyQ2GP87w  
e3e8Vc9J654NMo0oqawfgs6difgha91od2wMfv35rjvoP

# Recurso da Web

- Um recurso da web é qualquer **recurso identificável** presente ou conectado à **rede mundial de computadores**. Os recursos são identificados usando Identificadores Uniformes de Recursos (**URI**);
- Um recurso da Web é qualquer um dos recursos criados durante o desenvolvimento de um aplicativo da Web;
- O conceito de **recurso da web** evoluiu ao longo da história da web, desde a noção inicial de documentos ou arquivos endereçáveis estáticos , até uma definição mais genérica e abstrata, agora abrangendo cada "coisa" ou "entidade" que pode ser identificada, nomeada, endereçada ou manipulada , de qualquer forma, na web em geral ou em qualquer **sistema de informação** em rede.

# Recurso da Web

- **URN** – Uniform Resource Name: o nome do recurso que será acessado na internet;
- **URL** – Uniform Resource Locator: refere-se ao local, o Host, que deseja-se acessar determinado recurso. O objetivo da URL é associar um endereço remoto com um nome de recurso (URN) na Internet;
- **URI** – Uniform Resource Identifier: é o identificador do recurso. Pode ser uma imagem, uma página, etc, pois tudo o que está disponível na internet precisa de um identificador único para que não seja confundido na web. Tanto a URN quanto a URL são tipos de URI.

# Relação entre URI, URL e URN

```
<protocol>://<host>[:<port>][<path>[?<query>]]
```

URI = URL + URN

**URI**

Uniform Resource **I**dentifier

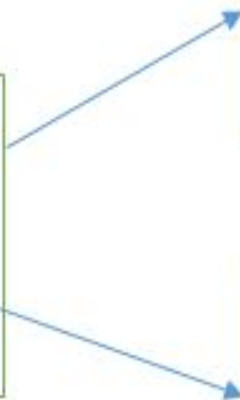
**URL**

Uniform Resource **L**ocator

```
urn:<namespace>:<string>
```

**URN**

Uniform Resource **N**ame



# URI = URL -> URN

Term		Definition
URI	URI	<ul style="list-style-type: none"><li>• Unique identifier for online resources<ul style="list-style-type: none"><li>- Location</li><li>- Identity</li></ul></li></ul>
	URL	<ul style="list-style-type: none"><li>• Locator of online resources</li></ul>
	URN	<ul style="list-style-type: none"><li>• Unique identifier of online resources</li></ul>
	URC	<ul style="list-style-type: none"><li>• Metadata for information</li></ul>

URC : Uniform Resource Characteristic

URI = URL + URN

<http://www2.ifam.edu.br/IFAMaplicaovertpng.png>

URL = <http://www2.ifam.edu.br/>

URN = IFAMaplicaovertpng.png

URI = <http://www2.ifam.edu.br/IFAMaplicaovertpng.png>



# URI = URL + URN

http://127.0.0.1:5500/html/formulario.html?nmNome=Nicolas

`<protocol>://<host>[:<port>][<path>[?<query>]]`

URL = http://127.0.0.1:5500/html/formulario.html?nmNome=

`urn:<namespace>:<string>`

URN = Nicolas

URI = http://127.0.0.1:5500/html/formulario.html?nmNome=Nicolas

# Web

- Web é uma palavra inglesa que significa teia ou rede. O significado de web ganhou outro sentido com o aparecimento da internet. A web passou a designar a rede que conecta computadores por todo mundo, a World Wide Web (WWW).
- A Web significa um sistema de informações ligadas através de **hipermídia** (hiperligações em forma de texto, vídeo, som e outras animações digitais) que permitem ao usuário acessar uma infinidade de conteúdos através da internet.



# Aplicativo Web

- Os Aplicativos Web recebem este nome porque são executados na internet. Ou seja, os dados ou os arquivos que você trabalha são processados e armazenados dentro da web. Estes aplicativos geralmente não precisam ser instalados no seu computador.
- O conceito de Aplicativos Web está relacionado com o armazenamento na nuvem. Toda a informação é guardada de forma permanente em grandes servidores de internet e estes enviam aos nossos dispositivos ou computadores os dados requeridos.

# Navegador: Um Cliente HTTP

- Browser é um programa desenvolvido para permitir a navegação pela web, capaz de processar diversas linguagens. Sua interface vai variar de acordo com o Fabricante (Mozilla, Chrome, Internet Explorer, etc...), onde quem escolhe é o usuário.
- O browser ou web browser é responsável pela comunicação com os servidores (servidores web), é ele que processa os dados recebidos pelos servidores da Internet e processa as respostas. Antigamente, os primeiros navegadores tinham apenas texto, mas com o tempo foram aperfeiçoados, foram criados mecanismos para interagir com o usuário, com interfaces rápidas, coloridas e de fácil acesso.
- Os browsers contêm plugins, que podem ser instalados, que ativam diversas novas funcionalidades, permitindo que o utilizador tenha acesso a diferentes tipos de conteúdos e aplicativos.

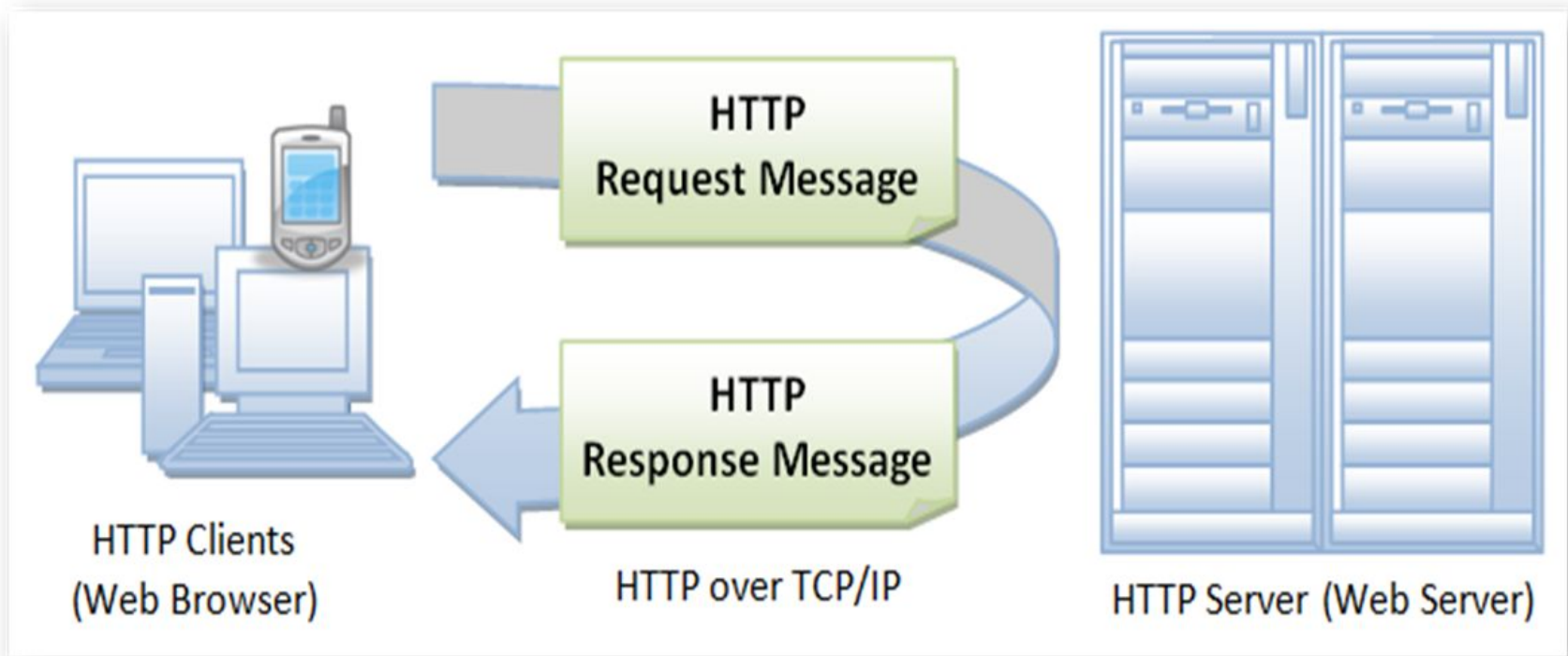
# Navegador: Um Cliente HTTP



# Servidor Web (Servidor HTTP)

- Um programa de computador responsável por aceitar pedidos **HTTP** de Clientes, geralmente os navegadores ou aplicativos, e servi-los com respostas HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos HTML com objetos embutidos (imagens, áudio, etc.);
- Os servidores web são responsáveis por armazenar e trocar informações com outras máquinas. Por causa disso, pelo menos dois participantes são envolvidos em cada troca de informações: um **cliente**, que solicita informações, e um **servidor**, que atende a esses pedidos.
- Cada lado exige também um programa especializado para negociar a troca de dados. No caso do cliente, é usado um browser (navegador). No lado do servidor, porém, as coisas não são tão simples. Existem várias opções de software disponíveis, mas todos têm uma tarefa semelhante: negociar transferência de dados entre clientes e servidores via HTTP, o protocolo de comunicação da Web.

# Servidor Web (Servidor HTTP)



# Servidor web HTTP (web server)

- Referente ao software, um **Servidor Web HTTP** inclui diversos componentes que controlam como os usuários acessam os arquivos hospedados (armazenados para disponibilização), no mínimo um servidor HTTP. Um servidor HTTP é um software que compreende URLs (endereço web) e HTTP (o protocolo que seu navegador utiliza para visualizar páginas web).
- Servidor web é um software responsável por aceitar pedidos em HTTP de clientes, geralmente os navegadores, e servi-los com respostas em HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos em HTML com objetos embutidos (imagens, mídias, etc)



# Servidor Web (Servidor HTTP)

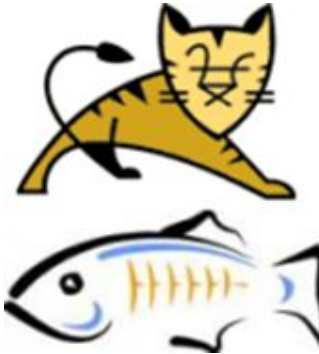
## Servidor Web

### Servidores Web disponíveis:

- Apache
- Lighttpd
- Microsoft IIS
- Zeus Web Server
- Sun Java System Web Server
- Xitami Web Server
- TUX \*
- KHTTPd \*



Tomcat



GlassFish

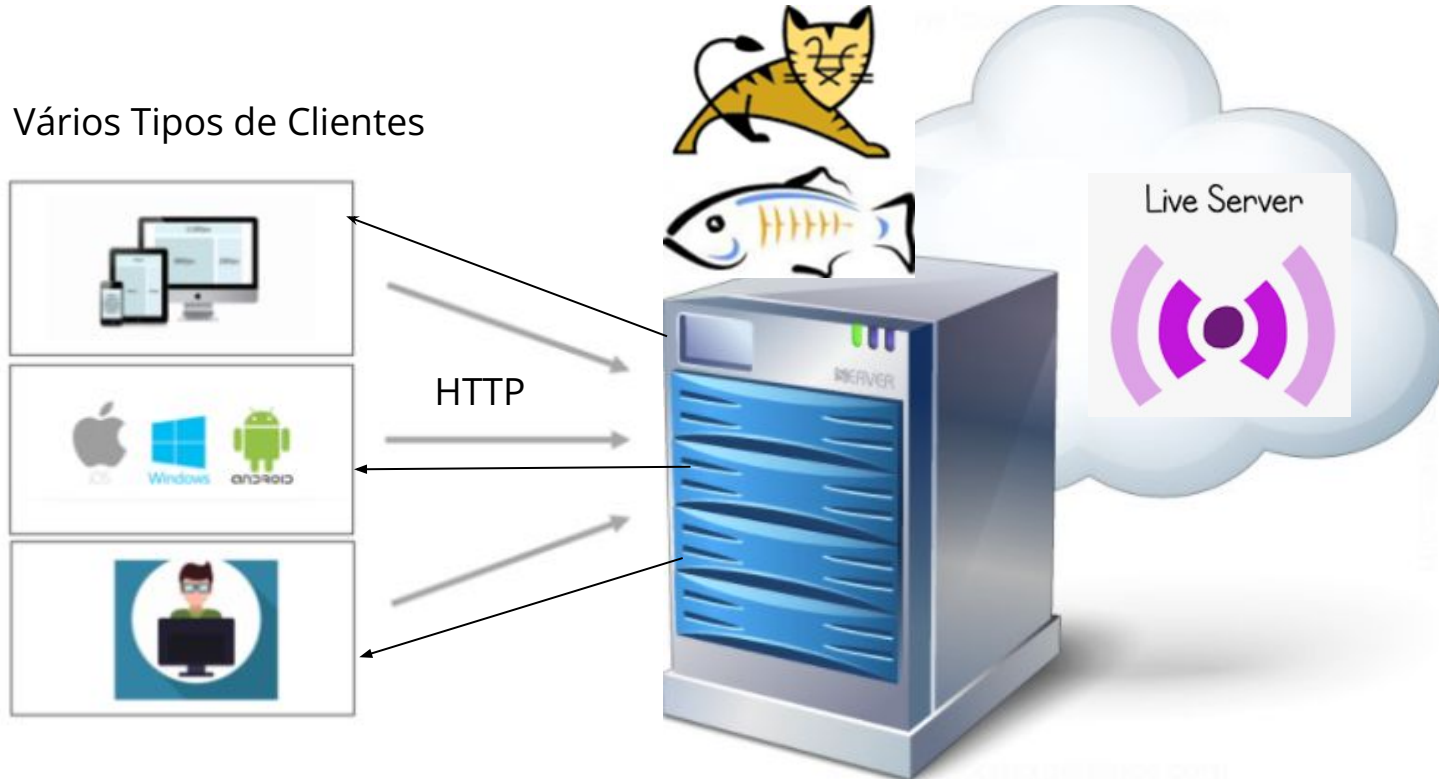


Live Server



# Relação Cliente/Servidor: Serviço Web

Vários Tipos de Clientes



# Verificar dados do Protocolo HTTP no Navegador:

1. Acessar o Navegador Chrome.
2. Acessar URI:  
<http://www2.ifam.edu.br/>
3. Pressionar: F12
4. Pressionar: CTRL + R
5. Selecionar Aba: Network
6. Na coluna Name selecionar:  
[www2.ifam.edu.br](http://www2.ifam.edu.br/)
7. Selecionar Aba: Headers

The screenshot displays the Chrome DevTools interface with the Network and Headers tabs active. The Network tab shows a list of requests, with 'www2.ifam.edu.br' selected. The Headers tab shows the details of the selected request.

**Request Details:**

- Request URL: <http://www2.ifam.edu.br/>
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 200.129.168.177:80
- Referrer Policy: origin

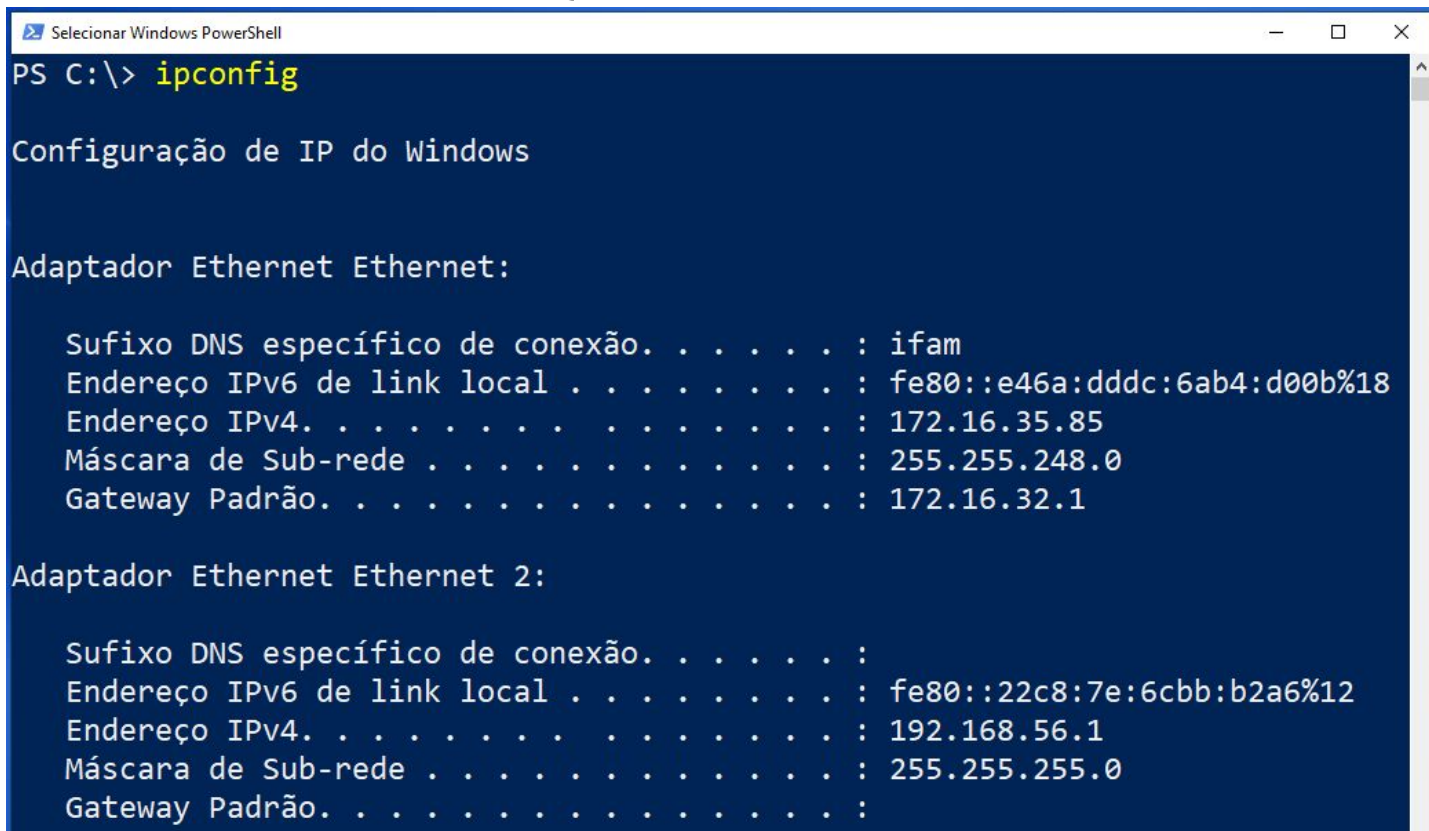
**Response Headers:**

- Accept-Ranges: bytes
- Age: 3058
- Cache-Control: max-age=0, s-maxage=86400, must-revalidate
- Content-Encoding: gzip
- Content-Language: pt-br
- Content-Length: 14893
- Content-Type: text/html; charset=utf-8
- Date: Mon, 19 Feb 2024 21:00:23 GMT
- Expires: Fri, 21 Feb 2014 21:00:23 GMT
- Server: Zope/(2.13.22; python 2.6.6; linux2) ZServer/1.1
- Vary: Accept-Encoding
- Via: 1.1 varnish-v4
- X-Cache: HIT
- X-Cache-Hits: 112
- X-Cache-Operation: plone.app.caching.moderateCaching
- X-Cache-Rule: plone.content.itemView
- X-UA-Compatible: IE=edge,chrome=1
- X-Varnish: 593126 589833

**Request Headers:**

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng;q=0.8,application/signed-exchange;q=0.7
- Accept-Encoding: gzip, deflate
- Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
- Cache-Control: max-age=0
- Connection: keep-alive
- Cookie: \_ga=GA1.3.22831575.1698708329; \_gid=GA1.3.1898515206.1708365396

# Verificar o IP do computador



```
PS C:\> ipconfig

Configuração de IP do Windows

Adaptador Ethernet Ethernet:

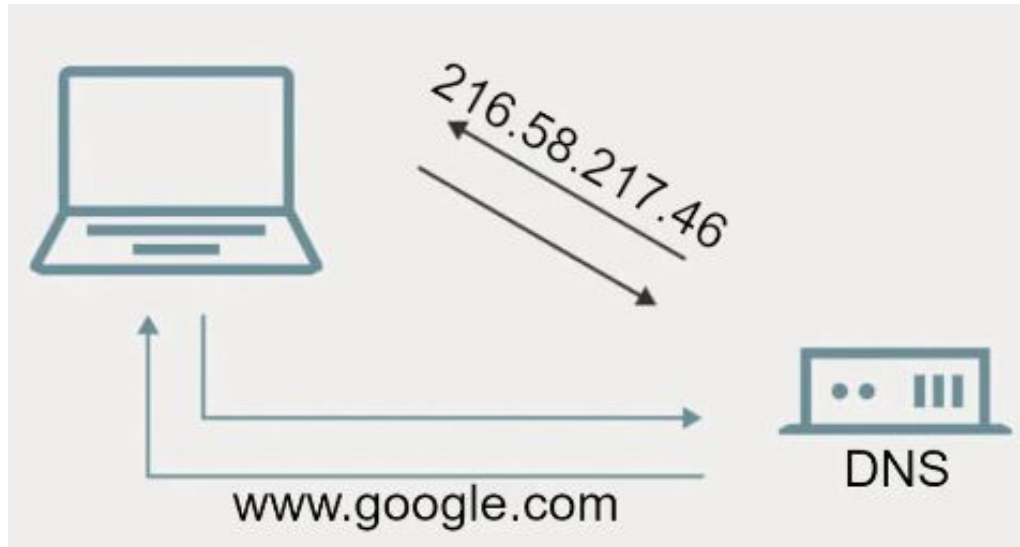
    Sufixo DNS específico de conexão. . . . . : ifam
    Endereço IPv6 de link local . . . . . : fe80::e46a:dddc:6ab4:d00b%18
    Endereço IPv4. . . . . : 172.16.35.85
    Máscara de Sub-rede . . . . . : 255.255.248.0
    Gateway Padrão. . . . . : 172.16.32.1

Adaptador Ethernet Ethernet 2:

    Sufixo DNS específico de conexão. . . . . :
    Endereço IPv6 de link local . . . . . : fe80::22c8:7e:6cbb:b2a6%12
    Endereço IPv4. . . . . : 192.168.56.1
    Máscara de Sub-rede . . . . . : 255.255.255.0
    Gateway Padrão. . . . . :
```

# Servidor DNS

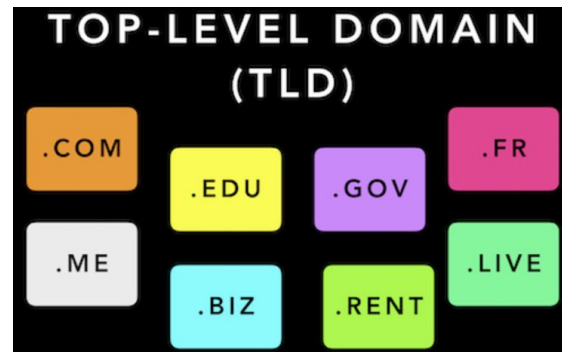
- O sistema **DNS** (Domain Name System), isto é, Sistema de Nomes de Domínios da internet funciona praticamente como uma agenda de telefone ao gerenciar o mapeamento entre nomes e números. Os servidores DNS converte solicitações de nomes em endereços IP, controlando qual servidor um usuário final alcançará quando digitar um nome de domínio no navegador da web.
- O DNS, de forma técnica, é um serviço hierárquico e distribuído para computadores, serviços ou qualquer recurso conectado à Internet que funciona como um sistema de tradução de **nomes de domínios** (hosts) para **endereços IP**.



# DNS : (Top-Level Domain, TLD)

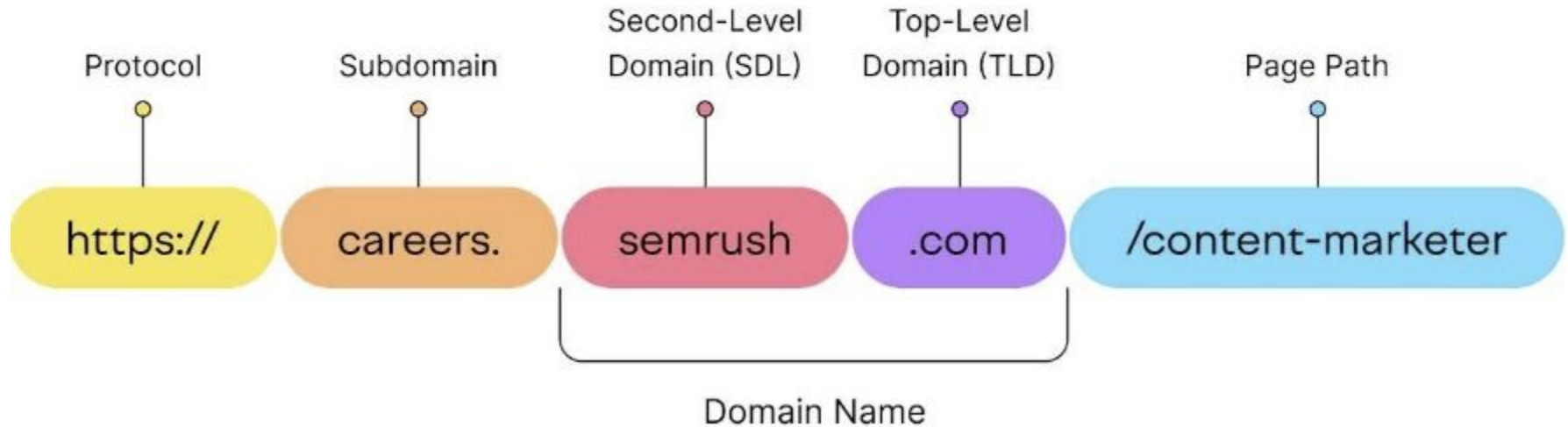
- O domínio de nível superior (Top-Level Domain, TLD), também conhecido como uma extensão de domínio, é a segunda parte do nome de domínio.
- Os domínios de nível superior representam o tipo de organização que você tem. Por exemplo, os sites do governo usam o domínio de nível superior ".gov", enquanto as empresas comerciais normalmente usam o ".com".
- Outros TLDs incluem:
  - .mil
  - .edu
  - .ca
  - .net
  - .org

**<http://ifam.edu.br>**



DNS : Second-Level Domain (SDL) e Top-Level Domain (TLD)

## Parts of a URL



# DNS: Obter o IP de um determinado domínio

Google Admin Toolbox Dig

Nome  
www.ifam.edu.br

A

AAAA

ANY

CAA

CNAME

DNSKEY

DS

MX

NS

PTR

SOA

SRV

TLSA

TSIG

TXT

CNAME

TTL:

59 minutes 53 seconds

TARGET:

tucano2.ifam.edu.br.

A

TTL:

59 minutes 53 seconds

DATA:

200.129.168.170



# Verificar a rota de um IP

Windows PowerShell

```
PS C:\> date
```

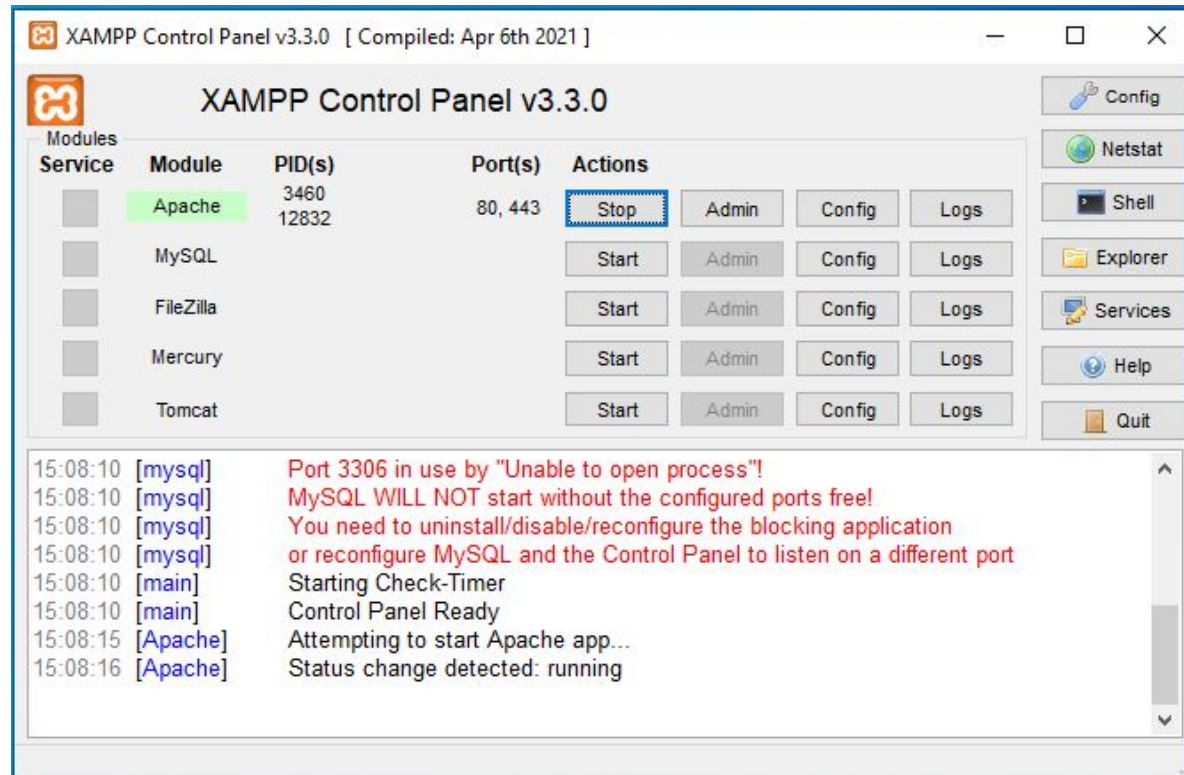
```
segunda-feira, 19 de fevereiro de 2024 18:16:36
```

```
PS C:\> tracert www.ifam.edu.br
```

```
Rastreando a rota para tucano2.ifam.edu.br [200.129.168.170]  
com no máximo 30 saltos:
```

1	2 ms	<1 ms	6 ms	pfCMZLRedeInterna.ifam [172.16.32.1]
2	10 ms	2 ms	4 ms	200.129.166.10
3	2 ms	2 ms	4 ms	200.17.10.33
4	6 ms	3 ms	1 ms	C8819D65.ufam.edu.br [200.129.157.101]
5	2 ms	4 ms	5 ms	C8819D15.ufam.edu.br [200.129.157.21]

# Servidor HTTP Apache



Software: Apache - PID: 3460 e 12832 - Port: 80 (http) e 443 (https)

O comando utilizado para validar as portas em uso é: **netstat -ona** e pressionar Enter, com isso irá listar as portas que estão em uso:

```
Anaconda Prompt (anaconda3)

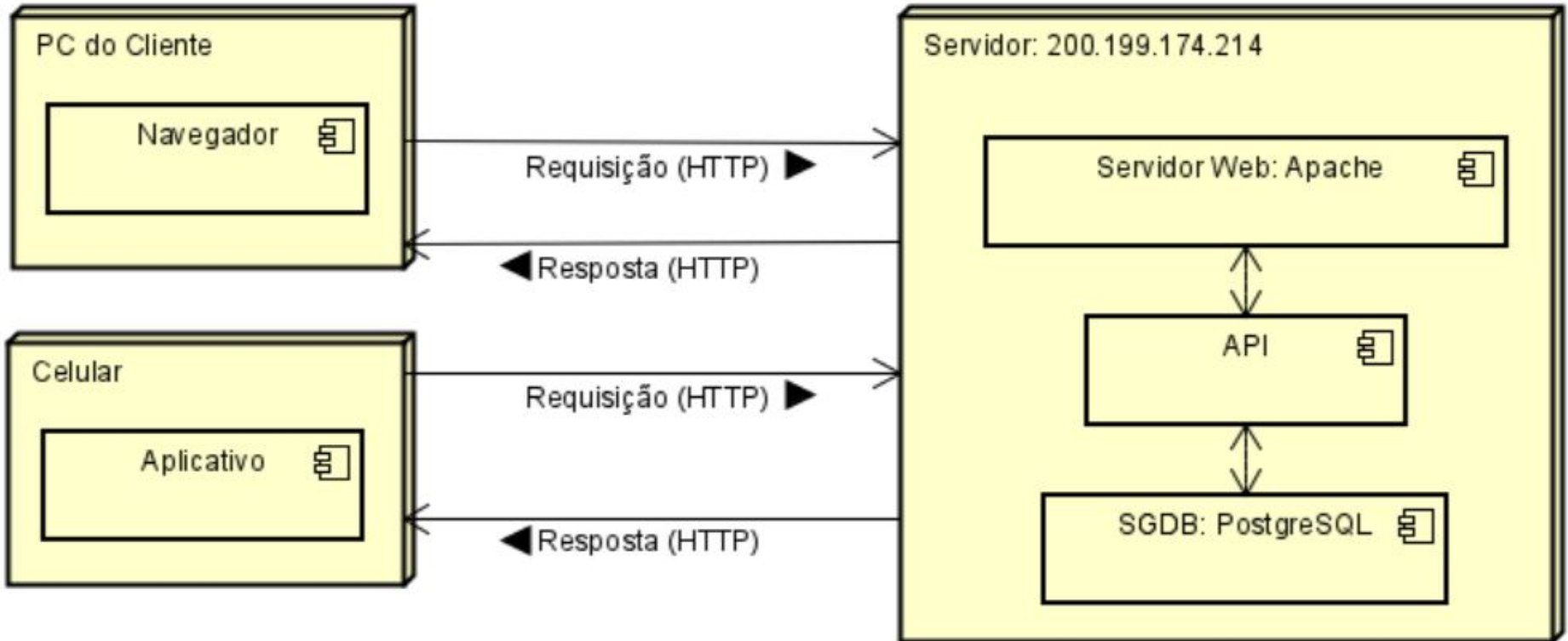
(base) C:\>date
Data atual: 20/02/2024
Digite a nova data: (dd-mm-aa)

(base) C:\>netstat -ona

Conexões ativas
```

Proto	Endereço local	Endereço externo	Estado	PID
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING	3460
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	1384
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING	3460
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:623	0.0.0.0:0	LISTENING	4876
TCP	0.0.0.0:3306	0.0.0.0:0	LISTENING	5984
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING	8520
TCP	0.0.0.0:5432	0.0.0.0:0	LISTENING	5828
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING	2168
TCP	0.0.0.0:16992	0.0.0.0:0	LISTENING	4876
TCP	0.0.0.0:33060	0.0.0.0:0	LISTENING	5984
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING	1068
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING	8
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING	2084
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING	1872
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING	3720
TCP	0.0.0.0:49720	0.0.0.0:0	LISTENING	1052
TCP	127.0.0.1:49676	127.0.0.1:49677	ESTABLISHED	4876
TCP	127.0.0.1:49677	127.0.0.1:49676	ESTABLISHED	4876
TCP	127.0.0.1:49715	127.0.0.1:49716	ESTABLISHED	5984

# Relação Cliente/Servidor: Serviço Web



# Conceitos Básicos

## LocalHost:

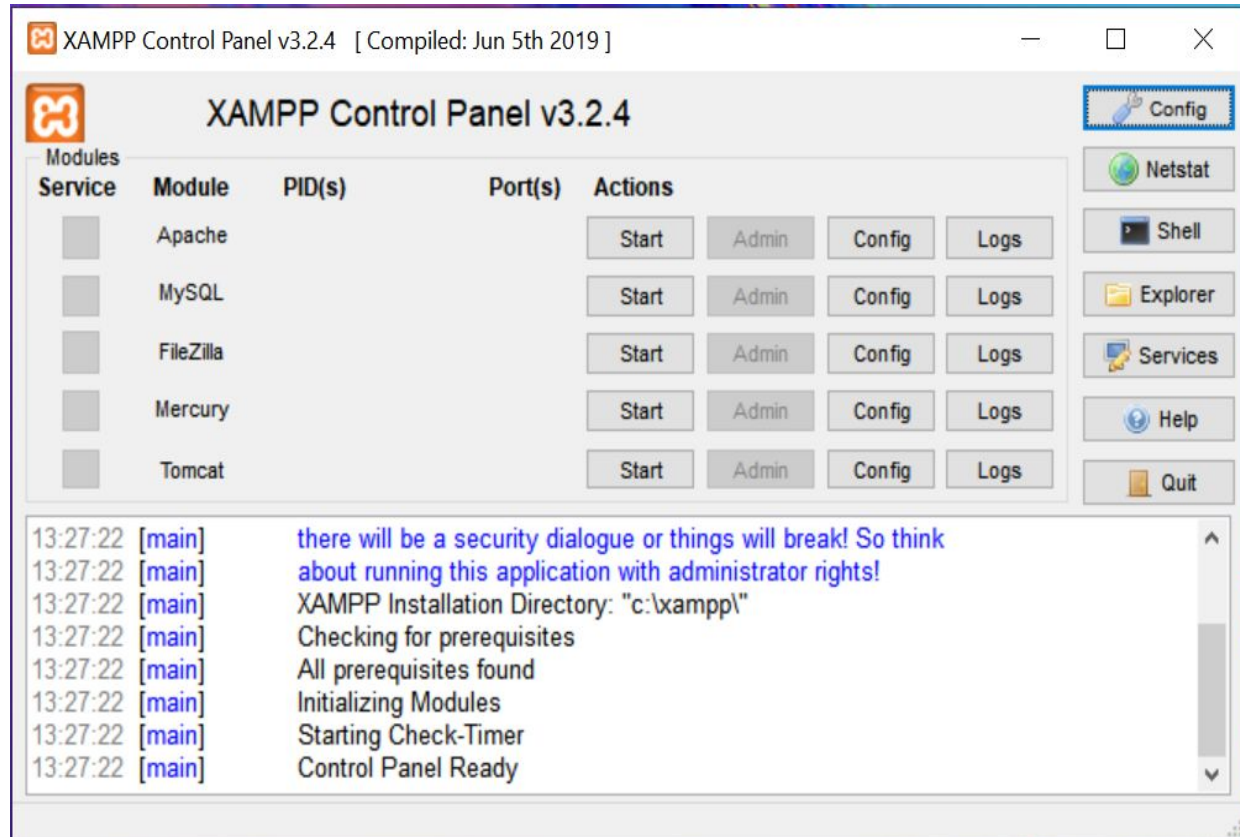
- Na computação, o termo localhost se refere à localização do sistema que está sendo usado, ou seja, o computador do usuário. É um dispositivo loopback ao qual é atribuído o endereço IP 127.0.0.1 no IPv4, ou ::1 no IPv6, e pode ser usado por aplicações **TCP/IP** para **testarem** a comunicação consigo mesmo.

## Endereço IP ou Loopback:

- O endereço IP do LocalHost 127.0.0.1 (IPv4) é considerado um endereço de “loopback”, porque as informações enviadas a ele são roteadas de volta para a máquina local.
- Porta **80** : Protocolo HTTP. Ex: **localhost:80/**
- Porta **443** : Protocolo HTTPS. Ex: **127.0.0.1:443/**

# XAMPP

XAMPP (**X**= multi sistema operacional, **A**= Apache, **M** = MySQL ou MariaDB, **P** = PHP, **P**=Perl) é um pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Servidor HTTP Apache com suporte as linguagens PHP e Perl. De plataforma, software livre.



# Caminho Físico e Caminho Relativo no XAMMP

Entendendo que:

1. URI = URL + URN
2. URN = Nome de um recurso na Web
3. URL = Localização do recurso na web:
  - a. Padrão: protocolo://host:porta/path/nomeDoRecursoSolicitado;
  - b. Ex: http://localhost:80/path/nomeDoRecursoSolicitado;



# Caminho Físico e Relativo: XAMPP

Para a plataforma XAMPP os Servidores HTTP Apache possui os seguintes caminhos:

## 1. Servidor HTTP **Apache**:

Caminho Físico : C:\xampp\htdocs

Caminho Relativo: <http://localhost:80>

## 2. Servidor HTTP **Apache** com uma pasta de trabalho chamado “**web**”, e um recurso chamado “**index.html**”

Caminho Físico : C:\xampp\htdocs\web\index.html

Caminho Relativo: <http://localhost:80/web/index.html>

## 3. Fazer a prática com um arquivo básico index.html usando o XAMPP.