Desenvolvimento de Aplicações Web

Protocolo HTTP

Prof. Márcio Barros

Protocolos e a Internet

- Um protocolo é uma forma de ritual, um conjunto de sinais trocados entre dois computadores de modo que um consiga transferir informações para o outro

- O protocolo define a forma com que a informação será trocada, a codificação dos dados, os mecanismos de tratamento de erro e recuperação das informações, entre outros recursos

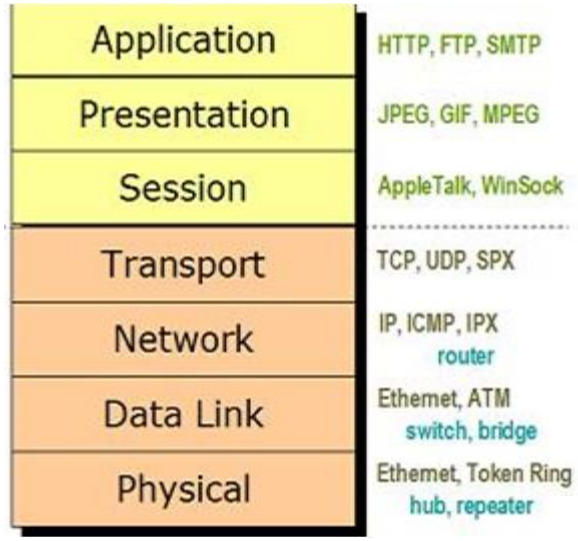
- A Internet é um conjunto de protocolos padronizados que permitem a oferta de diferentes serviços (SMTP, FTP, HTTP)

Modelo OSI/ISO

- Modelo de sete camadas que hoje é usado como padrão em redes de computadores

- Na Internet, o protocolo base é o TCP/IP, mas no nível de aplicação temos os protocolos que mais nos interessam

- Entre eles, destaca-se o HTTP



HyperText Transfer Protocol

- É um conjunto de regras que definem como informações de diferentes tipos (texto, imagens, vídeos, sons) são trocadas na World Wide Web

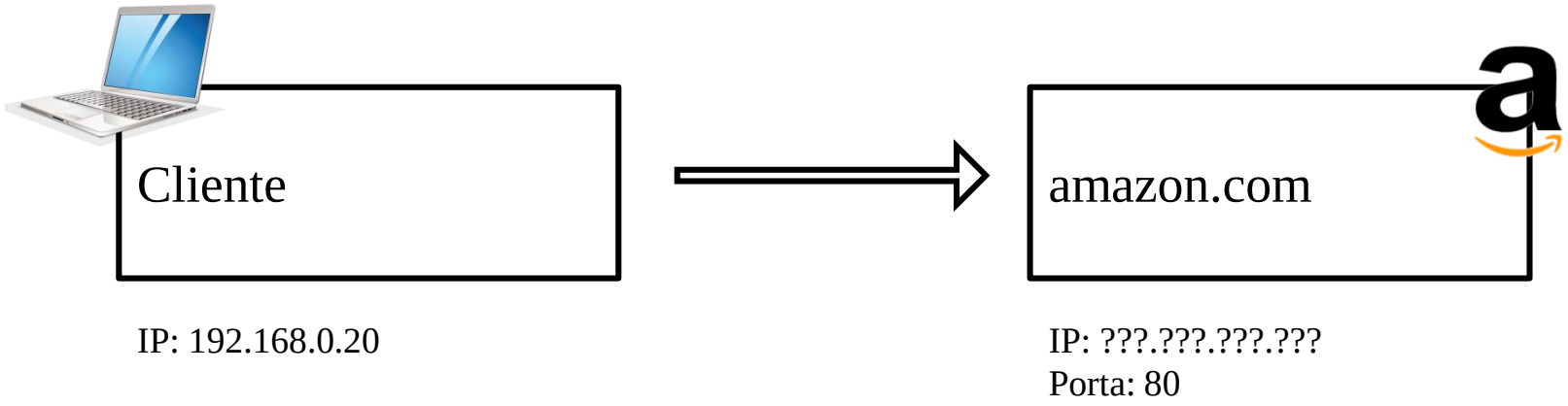
- É baseado em um paradigma do tipo requisito-resposta
  - Um computador que atua como cliente gera um requisito por um recurso que está no computador que atua como servidor
  - O requisito carrega o endereço do recurso desejado na forma de uma URL (*Universal Resource Location*)
  - O servidor localiza e retorna o recurso desejado, normalmente um documento escrito na linguagem HTML
  - Esta comunicação pode ser implementada sobre diversos protocolos, mas ocorre usualmente sobre o TCP/IP

Terminologia

- Endereço IP: rótulo numérico atribuído a cada dispositivo (computador, impressora, celular) que participa de uma rede de troca de dados que usa o protocolo IP para comunicação
- TCP/IP: dois protocolos da Internet nos níveis abaixo da aplicação. TCP permite a entrega confiável de pacotes, na mesma ordem em que foram transmitidos e com verificação de erros, em uma rede pública ou privada
- Número de porta: número de 16 bits que, quando associado ao endereço IP, identifica unicamente um destino para uma sessão de comunicação
- Socket: combinação de um endereço IP com um número de porta, que representa uma ponta em uma comunicação
- Endereço virtual: endereço de um recurso em de uma ponta da sessão de comunicação
- URL = protocolo + endereço IP + porta + endereço virtual

Como funciona a comunicação?

Como funciona quando um cliente quer acessar um servidor cujo IP o cliente não conhece?



Como funciona a comunicação?

Cliente

Qual é o IP para
www.amazon.com?

52.84.173.227

DNS Server

IP: 181.213.132.2
Porta: 53

Cliente

GET index.html

<html>...</html>

amazon.com

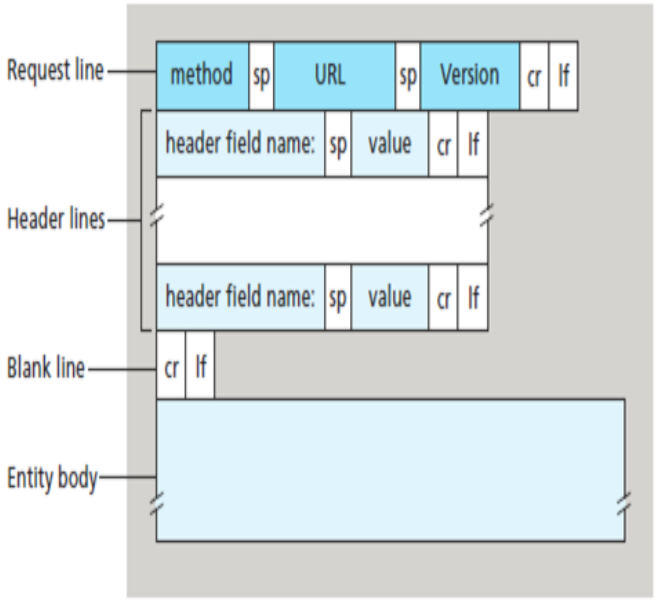
IP: 52.84.173.227
Porta: 80

Protocolo HTTP

- O protocolo HTTP é *stateless*
  - Todas as informações sobre uma conexão são armazenadas apenas durante um ciclo de requisito-resposta
  - O servidor não guarda memória entre diferentes pedidos realizados pelo cliente
  - O servidor nem mesmo sabe se dois pedidos em sequência vieram do mesmo cliente
  - Se um mesmo cliente pede o mesmo recurso duas vezes seguidas, o servidor responde à segunda requisição da mesma forma que a primeira
  - Isto simplifica a implementação do servidor, mas exige que as aplicações implementem recursos para armazenar dados entre duas ou mais requisições de um mesmo cliente

Requisição HTTP

- Uma requisição HTTP é um comando em formato de texto
  - A primeira linha contém o método desejado, a URL do recurso que será manipulado e a versão do protocolo
  - As linhas subsequentes contém o cabeçalho (header) da requisição, representado como pares de nome e valor
  - A última parte é o conteúdo da requisição, que é utilizado somente em alguns métodos



Métodos de requisição

- GET: o cliente pede um recurso ao servidor, indicando a URL do recurso desejado
- POST: usado para enviar dados para o servidor, visando atualizar um recurso armazenado por ele
- PUT: solicita que um recurso, enviado na requisição, seja armazenado em uma URL do servidor
- DELETE: solicita que o servidor remova o recurso armazenado em uma URL
- HEAD: idêntico ao GET, mas o servidor não retorna o recurso na resposta – apenas indica se ele existe
- TRACE: reenvia os dados recebidos na requisição para o cliente, apenas para depuração
- CONNECT: usado para criar "túneis HTTP", que permitem a troca de dados bidirecional e permanente entre cliente e servidor

Cabeçalho da requisição

- Existem quatro tipos de campos de cabeçalho (os principais estão listados abaixo)

Geral	Requisição do Cliente	Resposta do Servidor	Entidade
Cache-Control	Accept	Age	Allow
Connection	Accept-Charset	Etag	Content-Encoding
Date	Accept-Encoding	Location	Content-Language
Pragma	Accept-Language	Proxy-Authenticate	Content-Length
Transfer-Encoding	Authorization	Retry-After	Content-Type
	Cookie	Server	Expires
	Host	Set-Cookie	Last-Modified
	Max-Forwards	WWW-Authenticate	
	Referrer		
	User-Agent		

Mais detalhes em:
https://www.tutorialspoint.com/http/http_header_fields.htm

Requisição HTTP

```
GET /website/template/photography/ HTTP/1.1
Accept:*/*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.httpwatch.com
Connection: Keep-Alive
```

Exemplo de método GET

```
POST /website/template/photography/ HTTP/1.1
Accept:*/*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.httpwatch.com
Connection: Keep-Alive


nome=Por%20do%Sol&tipo=jpg
```

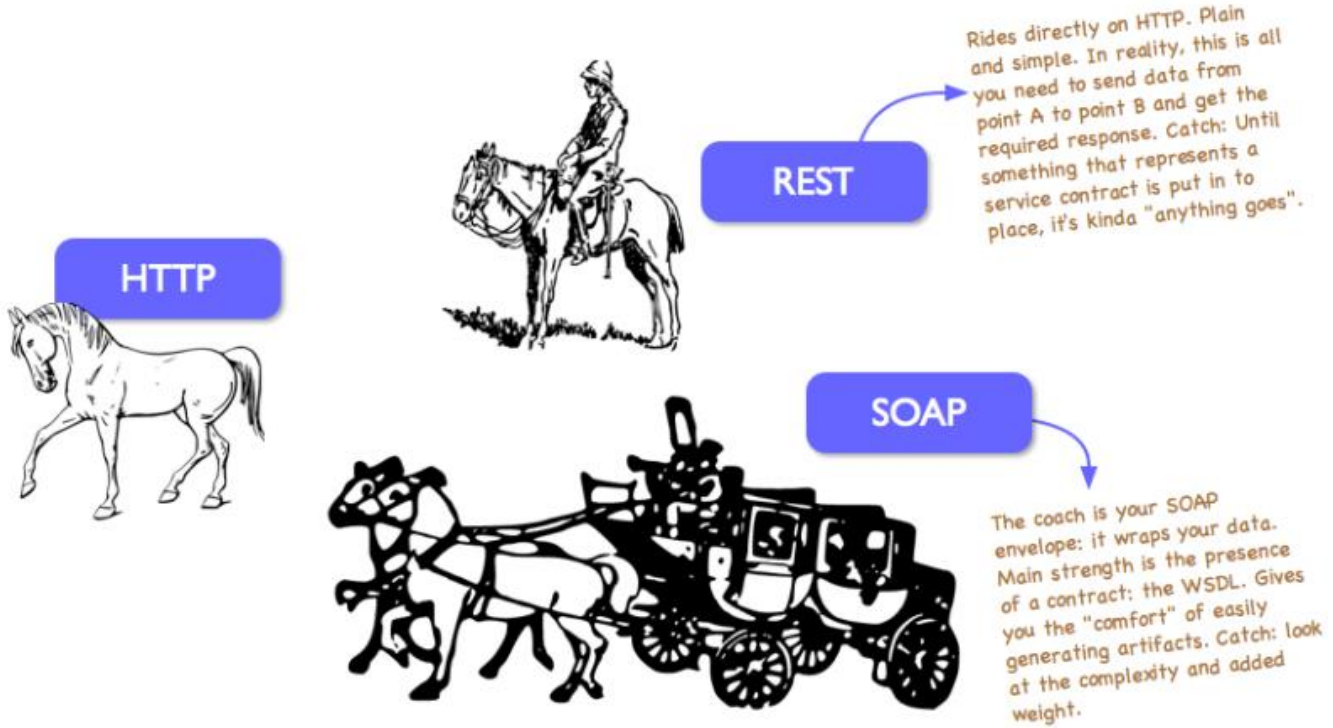
Exemplo de método POST

Requisição HTTP

- Estilo REST (Representational State Transfer)
  - O lado servidor das aplicações Web modernas é desenvolvido como uma API utilizada por diversos clientes (web, desktop, mobile, TV, etc)
  - O estilo REST indica que as aplicações Web devem oferecer serviços que respondem de forma diferente de acordo com o tipo de requisição e que pegam/retornam dados diretamente nesta requisição/resposta
  - O estilo REST está substituindo o modelo SOAP, definido originalmente para serviços Web, mas criticado devido ao excesso de overhead na comunicação

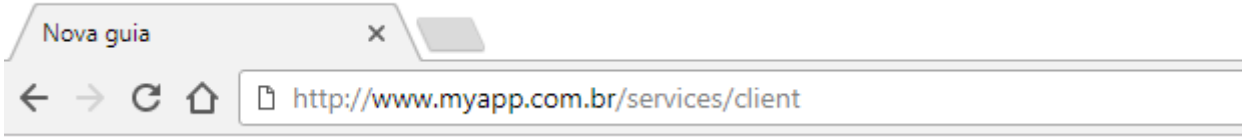
Tipos de requisição

- Estilos REST x SOAP



https://stackoverflow.com/questions/19884295/soap-vs-rest-differences

Tipos de requisição

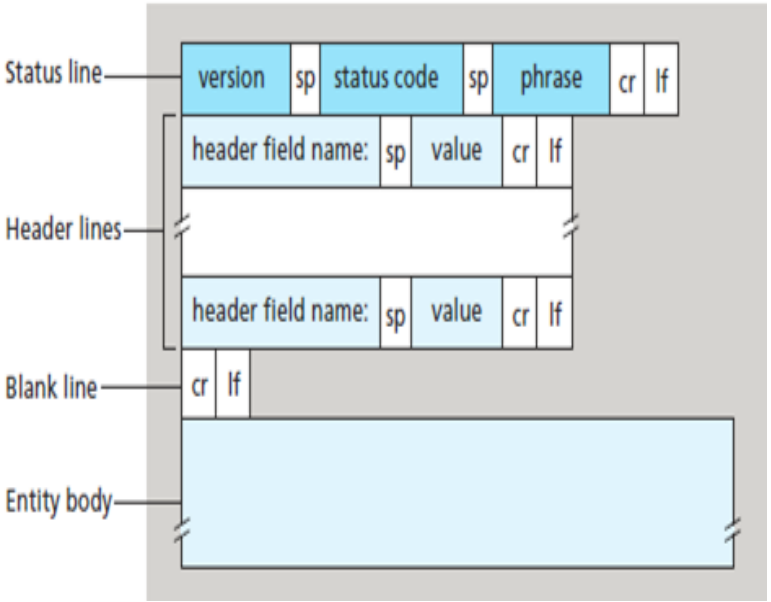


CRUD

PUT GET POST DELETE

Resposta HTTP

- A resposta HTTP possui três seções
  - Uma linha de status, contendo a versão do protocolo, um código de status e uma mensagem de status
  - Uma sequência de linhas de cabeçalho
  - O corpo da entidade, que contém o objeto requisitado, quando este retorno se aplicar



Resposta HTTP: Status

- 1XX Informational: indica que a requisição foi recebida e está sendo processada
- 200 Success: a requisição foi executada com sucesso e os dados estão disponíveis na resposta
- 3XX Redirection: indica que alguma ação deve ser tomada para dar continuidade ao processamento da requisição
  - 301 Permanently Moved: o recurso solicitado foi movido permanentemente para outro local. O novo endereço está no campo Location da resposta (e o navegador sabe disso ...)
- 4XX Client Error: indica um erro no lado cliente
  - 400 Bad Request: indica que a requisição não pode ser corretamente interpretada pelo servidor
  - 404 Not Found: indica que o recurso solicitado não existe no servidor (ao menos no diretório virtual que foi indicado)
- 5XX Server Error: indica um erro no lado servidor
  - 505 Unsupportted HTTP Version: a versão de protocolo HTTP usada na requisição não é suportada pelo servidor

Resposta HTTP

```
HTTP/1.1 200 OK
Date: Wed, 21 Aug 2013 09:02:49 GMT
Server: Apache
Cache-Control: no-cache
Pragma: no-cache
Set-Cookie: name=fulano#5567#2017-03-21T01-27-00.000-0500|...
Content-Language: en
Content-Encoding: gzip
Content-Length: 8162
Content-Type: text/html;charset=UTF-8
Expires: 0

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ”...">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:og=”..." >
<head>...
```

Exemplo de resposta HTTP

Análise do HTTP

Independente de plataforma

Pode ser usado sobre firewalls

Independente de conexão, o que exige menos das camadas inferiores de rede para rodar

Privacidade zero: todos podem ver o que está sendo transmitido, o que gera problemas, por exemplo, para transferir senhas.

A falta de criptografia também permite que o conteúdo seja alterado ao longo do caminho.

Falta de estado exige que aplicações criem mecanismos para manter seu estado.

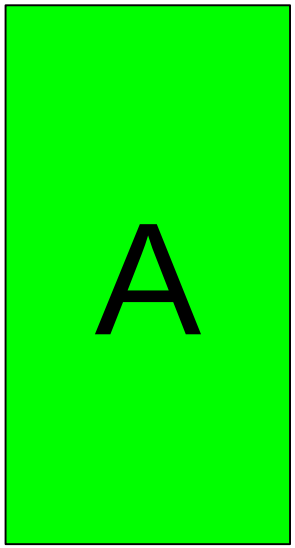
Protocolo HTTPS

- Alternativa mais comum ao HTTP
  - É o protocolo HTTP sobre um protocolo de camada de sessão chamado SSL (*Secure Socket Layer*)
  - O protocolo HTTPS criptografa uma mensagem antes de enviar pela rede e descriptografa a mensagem no destino
  - O protocolo usa a porta 443 ao invés da porta 80, que é usada pelo HTTP

- O protocolo exige um certificado, que é utilizado para criptografar os dados que transitam via HTTP
  - O certificado é emitido por uma *Certificate Authority* (CA), uma organização que emite certificados para uso por terceiros e que é vista com confiança pelos dois lados da comunicação
  - Ex. VeriSign, Amazon, GoDaddy, ...

Protocolo HTTPS

Criptografia de chave pública e privada



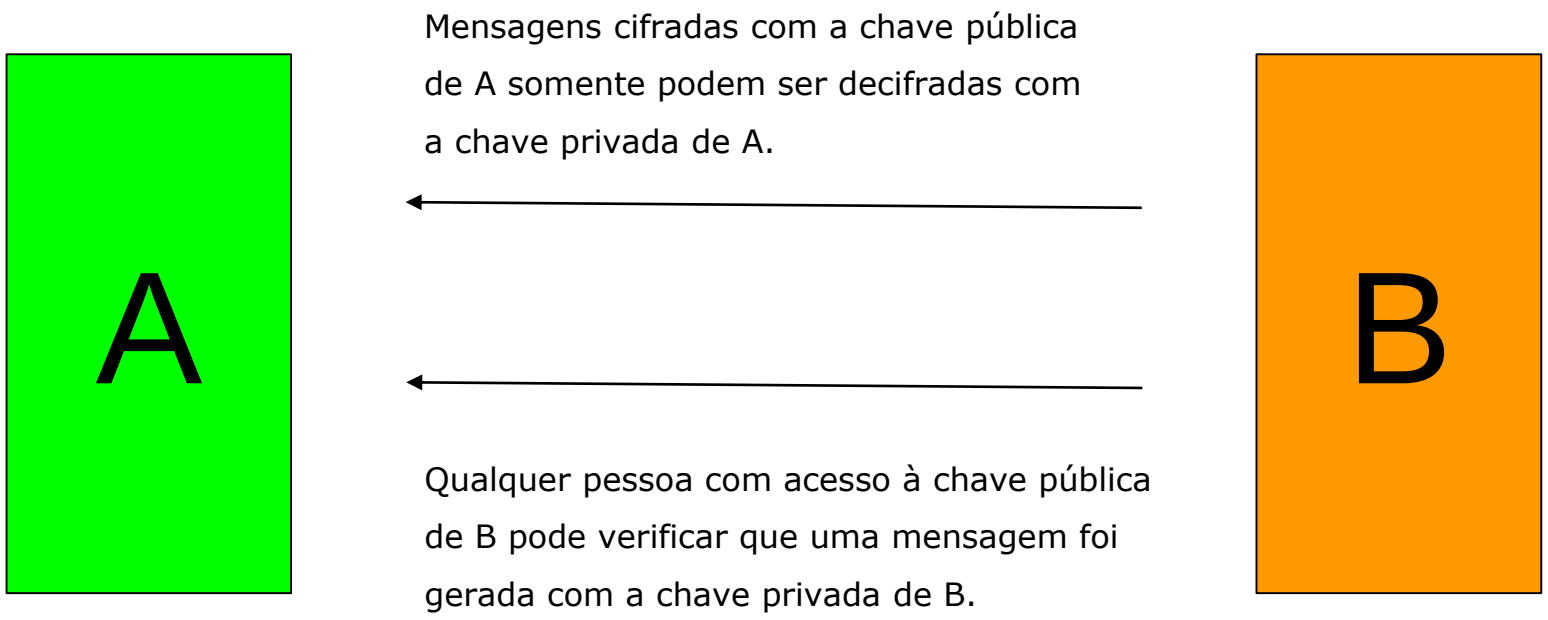
Cada agente envolvido na comunicação deve possuir um par de chaves.

A chave privada é conhecida apenas pelo próprio agente e deve ser mantida em segredo.

A chave pública pode ser distribuída pela Internet, de modo que qualquer pessoa interessada possa ter acesso a ela.

Protocolo HTTPS

Criptografia de chave pública e privada



Protocolo HTTPS

Navegador
(cliente)

1 Pede o endereço de www.site.com

www.site.com

2 Envia um certificado com sua chave pública, indicando a *certificate authority* que emitiu o certificado.

3 Navegador conhece a *certificate authority* e verifica se o certificado recebido do site é válido.

4 Gera um novo par de chaves e envia para o servidor, cifrando a nova chave privada pela chave pública do servidor. Somente o servidor pode decifrar a mensagem, utilizando a sua chave privada.

5 Somente as duas máquinas conhecem a nova chave privada e podem estabelecer um canal seguro através dela.

https://www.youtube.com/watch?v=T4Df5_cojAs

Protocolo HTTPS

- Limitações do protocolo
  - Um pouco mais lento que o HTTP (em função do tempo que é necessário para criptografar os dados)
  - Não previne que as informações privadas sejam roubadas no cache do navegador, pois os dados não são armazenados de forma criptografada