>>> Grupo de Estudos

>>> Aula 1

Name: Jose Carlos[†], Gustavo Casarim

Date: 1 de Novembro de 2018

[†]josecarlosdemoraesfilho@gmail.com

>>> Conteúdo

Arquitetura Cliente/Servidor

* A arquitetura de uma rede envolve um nível de complexidade que não cabe discutir no momento, porém serão ressaltados os aspectos básicos para a construção de um bom sistema web.

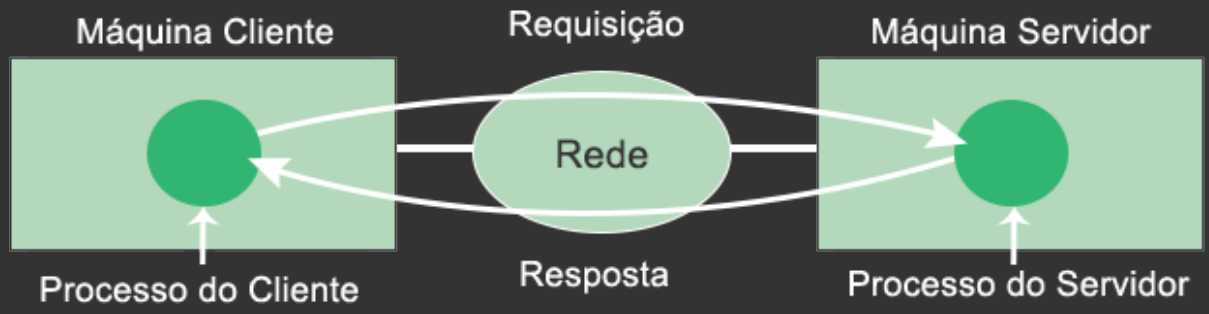

Figura 1: Estrutura ``Cliente - Servidor''

* Existem diversos tipos de redes, contudo, aqui utilizaremos uma abordagem abstrata e simples. A Figura 1 demonstra uma abstração de uma comunicação cliente-servidor.

>>> Protocolos e camadas de Redes

* A ideia é que o servidor processe parte ou toda a ``solução'', para poupar o processamento e espaço de memória do client-side (máquina do usuário). Além disso, fornece melhor segurança de dados e otimização de suporte, por centralizar a aplicação.
* Adotaremos no nosso diálogo um discurso voltado para o mercado e por isso, quando dicutirmos a respeito de um aspecto, ou conjuntos de aspectos de um sistema, tratremos como solução, por entender que elas se referem a necessidade de resolver algum problema previsto pela aplicação.
* Quando se trata de ``scripts'' de websites, chamamos de ``front-end'' o código interpretado pela máquina-cliente (browser) e ``back-end'' o código interpretado pelo servidor.

* Ambas as ``máquinas'' possuem uma hierarquia de camadas que possuem diferentes serviços.
* Uma camada n do client-side se comunica com sua respectiva camada (n) do server-side. Essa comunicação dá-se através das regras estabelecidas em seu respectivo protocolo, ou seja, para a camada n client-side se comunicar com a camada n do server-side, será utilizado o protocolo n (Figura 2).

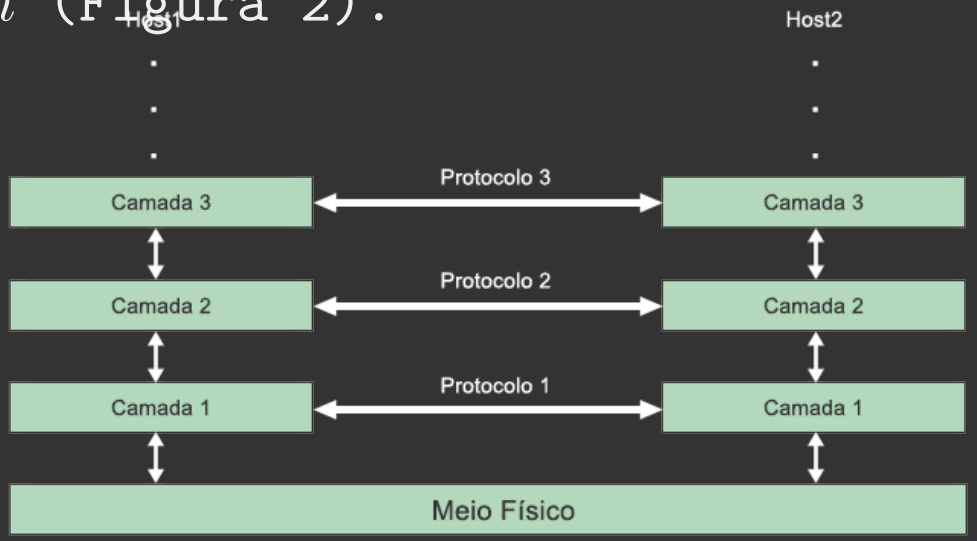

Figura 2: Hierarquia de Protocolos

>>> Protocolos e camadas de Redes

* Um protocolo bem conhecido é o Transmission Control Protocol (TCP) e o UDP (User Datagram Protocol). Esses protocolos se encontram na camada de ``Transporte''.
* O TCP e o UDP são protocolos que utilizam o IP (Internet Protocol) para entregar informações à rede. Um IP é uma sequência de números que identificam um dispositivo na rede.
* A camada de Transporte aceita dados de uma camada acima dela, chamada de ``Aplicação'', onde são associados os protocolos SSH, FTP, HTTP, DNS e outros.
* Os protocolos que serão destacados no nosso grupo de estudos são listados na Tabela 1.

>>> Protocolos e camadas de Redes

* Os protocolos que serão destacados no nosso grupo de estudos são listados na Tabela 1.

Protocolo	Descrição
DNS (Domain Name Service)	Mapeia nomes de hosts para seus respectivos endereços de redes (IP)
HTTP (Hypertext Transfer Protocol)	Busca páginas na World Wide Web
FTP	Utilizado para transferência de arquivos
SMTP	Protocolo de correio eletrônico

Tabela 1: Protocolos de Redes

DNS e URLs

>>> DNS (Domain Name System)

* Domain Name System (DNS) é um sistema de gerenciamento de nomes atribuídos a computadores conectados à uma rede. Como visto na Figura 2, os computadores/recursos são chamados ``hosts'', quais possuem um endereço binário (ex: IP), o que torna difícil a leitura humana dessas endereços.
* Imagine enviar um e-mail eletrônico para ``fulano@128.111.24.41'' ou acessar o site ``192.168.0.1''. Este e-mail além de não fornecer fácil leitura para humanos, pode ser trocado quando o serviço oferecido pelo servidor troque de máquina.
* A solução desse problema é o uso do DNS, que utiliza um sistema hierárquico de atribuição de nomes baseados em domínios, que é fornecido por meio de um serviço distribuído privado.

>>> Domain Name System (DNS)

* Um exemplo de ``domínio'' é o endereço www.google.com.
* Quando utilizamos um DNS e trocamos de máquina no server-side, basta associar o novo IP ao DNS que está sendo utilizado. Todos os usuários e serviços que ainda utilizarem esse meio, serão redirecionados para o novo IP.

>>> Uniform Resource Locators (URL)

* Quando acessamos um site, sabemos que seus ``links'' são nada mais do que ponteiros para outras páginas. Para que isso funcione, devemos ter uma forma de chegar até essa página específica que queremos acessar.
* Portanto, cada página contém um URL específico que funciona como um endereço universal.
* A partir de um DNS, considera-se um caminho local até o arquivo localizado na máquina.

>>> URL (Uniform Resource Locators)

* As URLs são separadas em três partes, quais são demonstradas na Figura 3.

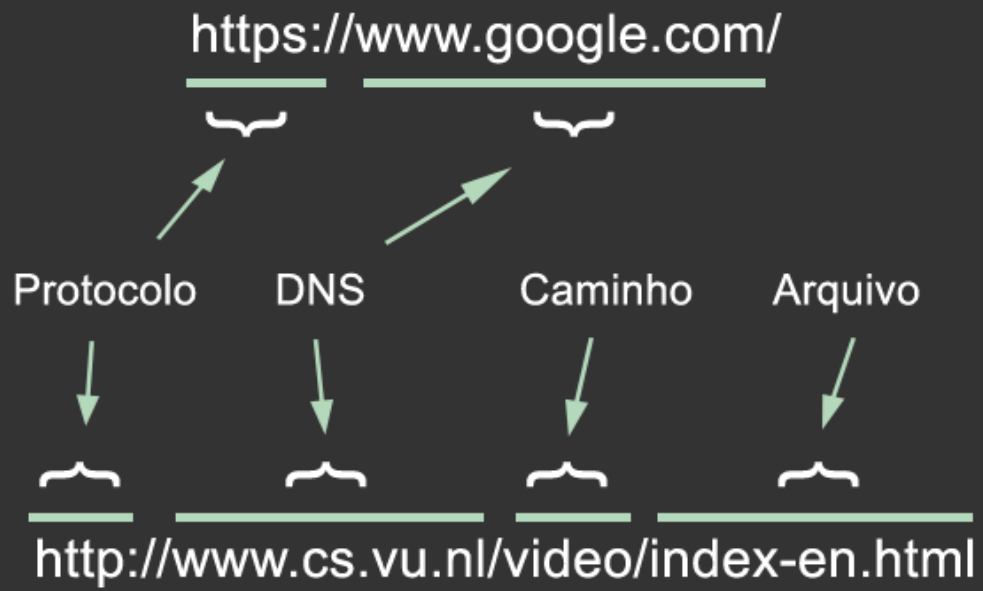

Figura 3: Estrutura de uma URL

HTTP (Hypertext Transfer Protocol)

>>> O Protocolo HTTP

* O HTTP é o protocolo de tranferência utilizado em toda a World Wide Web (Rede Mundial de Computadores), este faz a gestão do mecanismo de comunicação de cliente-servidor especificando quais mensagens cada um pode enviar e receber.
* Nessa troca de mensagens, o HTTP utiliza um componente chamado MIME (Multipurpose Internet Mail Extensions), qual trasmite os dados em contextos semelhantes à mensagens de e-mail.
* O MIME identifica o tipo de documento que está sendo tranferido. Essa estrutura deve ser configurada em ambos os lados, cliente e servidor.
* A configuração da estrutura de cada documento transferido, é baseada em um dos MIME types, qual deve ser informado como um tipo e um subtipo, ambos em tipo ``string'' separados pelo caractere ``'. A string resultande dessa combinação é case-insensitive.

>>> MIME Types

* Na Tabela 2 são listados alguns dos MIME Types e uma breve descrição de cada um.

tipo/subtipo	Descrição
text/plain, text/html, text/css, text/javascript	Texto legível tanto para máquina quanto para humanos.
image/gif, image/png, image/jpg	Qualquer tipo de imagem, incluindo gifs.
audio/wav, audio/mpeg	Arquivos de áudio.
video/mp4, video/ogg, video/webm	Arquivos de vídeo.
application/xml, application/pdf	Outros tipos de dados binários.

Tabela 2: MIME Types

>>> Métodos de Requisição HTTP

* As mensagens trocadas pelas máquinas utilizam operações chamadas ``métodos''. Essas mensagens consistem em uma sequência de linhas de caracteres ASCII, onde a primeira linha contém o método da solicitação.
* Também pode ser informado um ou mais ``HEADER'' (Cabeçalho), com informações adicionais (ou metadados) da requisição.
* Por fim, o corpo da mensagem também é adicionado no final desta sequência de linhas.
* Para exemplificar o uso desses métodos, faremos alguns testes em um software chamado Postman[1].

[1] https://www.getpostman.com/

>>> Métodos de Requisição HTTP

* Antes de alguns exemplos práticos, vejamos os principais métodos e suas respectivas descrições na Tabela 3.

METHOD (Método)	Descrição da solicitação
GET	Leitura de dados. (ex: uma página web)
HEAD	Leitura de cabeçalhos.
PUT	Armazenamento de dados. (Geralmente utilizado em atualizações)
PATCH	Armazenamento de dados. (Semelhante ao PUT, com algumas considerações)
POST	Acréscimo de dados ou recursos.
DELETE	Remoção de dados.
TRACE	Ecoa solicitação recebida.
CONNECT	Reserva uma conexão para uso futuro.
OPTIONS	Consulta opções configuradas para uma determinada solicitação.

Tabela 3: Métodos de requisição

>>> Códigos de Status

* Uma requisição sempre obtém um status como resposta, mesmo em caso de falhas. Esse Status é representado por uma sequência de três dígitos.
* O primeiro dígito divide os códigos de status em grupos relacionados. Por exemplo, o primeiro dígito 2 (2xx) separa as mensagens de sucesso para a requisição, enquanto as de dígito 4 (4xx) separam as respostas de falha na requisição.
* Alguns dos status mais utilizados são listados na Tabela 4.
* Os códigos de Status são extremamente úteis para tratamento das respostas e erros em um software. Dessa forma o desenvolvedor pode tomar as devidas providências ou ``tratar'' a mensagem para o usuário.

>>> Códigos de Status

* Não confundir o erro 400 com erro de Conexão. Quando houver um erro de conexão, este erro será local, já o erro 400 houve uma conexão, porém não atende às necessidades configuradas no server-side.

Código	Descrição
200	Resposta padrão para sucesso na operação.
204	Operação efetuada com sucesso, mas não é necessário nenhuma resposta.
400	Sintaxe de requisição incorreta, erro na requisição.
401	Erro de autenticação
404	Recurso solicitado não encontrado (famoso ``Not Found'').
405	Solicitação de recurso não permitido pelo servidor

Tabela 4: Códigos de Status

>>> Headers

* Os headers, como dito anteriormente, são metadados da requisição ou resposta.
* Nos headers estarão contidas informações como ``browser sendo utilizado'', ``Sistema Operacional sendo utilizado'', ``Data e hora do envio da mensagem'', ``MIME Type'', entre outros.
* Na Tabela 5 são listados alguns HEADERS.

HEADER	Descrição
User-Agent	Informações sobre o navegador e sua plataforma.
Accept	O tipo de páginas o cliente pode manipular (Métodos que o servidor aceita)
Authorization	Uma lista das credenciais do cliente.
Cookie	Envia um cookie definido anteriormente de volta ao servidor.
Date	Data e hora em que a mensagem foi enviada.
Content-Type	O tipo MIME da página.
Content-Length	O comprimento da página em bytes.

Tabela 5: HEADERS

Métodos de Requisição em Prática

>>> Método GET

* O método GET solicita a representação de um recurso especificado, ou seja, o método GET deve apenas recuperar dados. ACOMPANHE O EXEMPLO.

>>> Método POST

* O Método POST é utilizado para submeter uma entidade a um recurso específico, às vezes causando uma mudança no estado do recurso ou solicitando alterações ao servidor (ACOMPANHE O EXEMPLO). Em linhas gerais, significa que sempre que precisar inserir um NOVO REGISTRO, o método utilizado será o POST. ACOMPANHE O EXEMPLO.

>>> Método PUT

* O Método PUT substitui todas as atuais representações de seu recurso alvo pela carga de dados da requisição; Utilizado na atualização de um registro que por definição é feito em sua totalidade. ACOMPANHE O EXEMPLO.

>>> Método PATCH

* O Método PATCH é utilizado para aplicar modificações parciais em um recurso; Utilizado para atualizar uma informação específica de um recurso. ACOMPANHE O EXEMPLO.

>>> Método DELETE

* O Método DELETE remove um recurso específico; Utilizado na exclusão de um registro. ACOMPANHE O EXEMPLO.

>>> Referências

Todo o conteúdo desta aula foi baseado no livro [Tan] [Dev18]


Mozilla Developer, *Http request methods*, 2018, Last accessed at October 18 2018.


Andrew S Tanenbaum, *Computer networks (5th ed.)*.