UNIDADE 1 – CONCEITOS FUNDAMENTAIS DA WEB
MÓDULO 1 – ARQUITETURA CLIENTE-SERVIDOR

01

1 - ARQUITETURA CLIENTE-SERVIDOR

1.1 Introdução

Antes de definir o modelo de arquitetura cliente-servidor, é preciso recordar três termos relativos à área da computação:

Programa de computador é um conjunto de instruções e dados. TANENBAUM (2003, p.2)	**Processo** é um programa de computador em execução. TANENBAUM (2003, p.61)	**Sistema Distribuído** é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente. TANENBAUM (2010, p.1)

Diante dessas definições, é possível dizer que programas de computadores, escritos em alguma linguagem de programação, podem funcionar como programas **servidores** ou como programas **clientes**. Isso significa que podem existir programas que sejam apenas servidores, programas que sejam apenas clientes ou programas que sejam servidores e clientes simultaneamente.

A diferença entre os programas se deve ao propósito, à finalidade para a qual foram escritos. Quando estes programas estão em execução em algum computador, teremos processos servidores ou processos clientes em funcionamento. No decorrer do texto, considere que o uso dos termos servidores e clientes se refiram aos programas em execução, ou seja, processos.

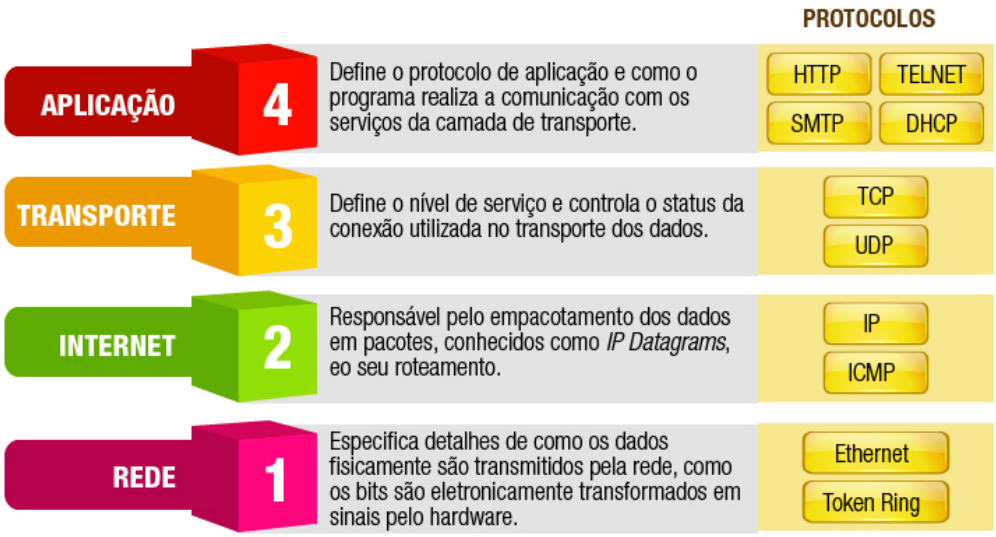
Deste modo, o modelo de arquitetura cliente-servidor de computação é uma estrutura de computação distribuída que divide tarefas ou cargas de trabalho entre os provedores de recursos ou serviços, chamados servidores, e os solicitantes destes recursos ou serviços, chamados de clientes.

02

Geralmente, clientes e servidores são executados em computadores fisicamente separados. No entanto, pode acontecer de clientes e servidores residirem fisicamente no mesmo computador. Em ambos os

casos, a rede de computadores será utilizada como meio de comunicação entre esses dois tipos de processos, uma vez que o sistema é distribuído.

O que diferencia a comunicação, neste caso, será a forma como a rede de computadores será utilizada. No primeiro caso, **o uso da rede se dará de forma plena**, ou seja, todas as camadas que compõem o modelo de redes serão utilizadas para permitir a comunicação. No segundo caso, **apenas algumas camadas serão utilizadas**, uma vez que ambos os processos residem na mesma máquina. Neste último caso, a interface de rede que permite a comunicação entre os processos é denominada de loopback (interface virtual). Como forma de ilustrar esse cenário, observe a figura a seguir, que representa uma das formas possíveis do modelo de redes TCP/IP:



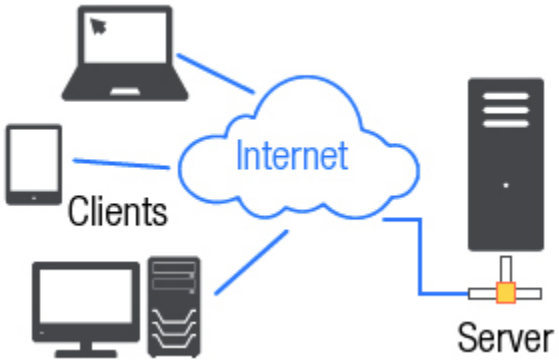
Modelo TCP/IP

Os processos clientes e servidores são executados sobre a camada denominada de “Aplicação”.

No caso de ambos os processos estarem sendo executados na mesma máquina, apenas a camada denominada de “Rede” não é utilizada. Isso significa que algumas camadas são utilizadas e, no caso da interface de loopback, são elas: “Aplicação”, “Transporte” e “Internet”.

03

No outro caso, naquele em que os processos estão sendo executados em máquinas distintas, todas as camadas do modelo são utilizadas. A figura a seguir mostra um diagrama de clientes e servidores se comunicando via internet, o que demonstra a necessidade de comunicação utilizando-se todas as camadas da figura anterior.



Client Server Model

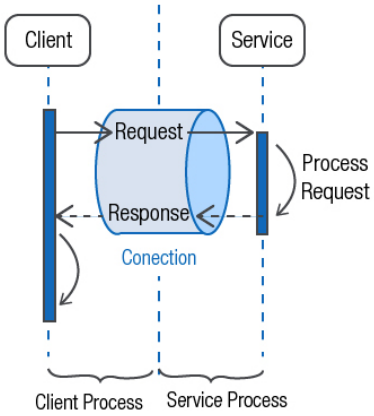
A diferença básica entre as duas formas é o simples fato de usarem ou não a camada física, denominada de **"Rede"** do modelo em epígrafe. A camada física compreende os elementos de *hardware* como: cabos, conectores, placa de rede, dentre outros. Isso significa que é possível desenvolver os programas para WEB em um único computador e depois colocá-los para executar em computadores diferentes que, do ponto de vista do programa (cliente ou servidor), não haverá qualquer tipo de modificação, pois eles são independentes da camada física.

Deste modo, para desenvolver programas para WEB, não se faz necessário o uso de vários computadores. Nesta disciplina, apenas o computador que você já está utilizando é suficiente para desenvolver os programas clientes e também os programas servidores.

2 - Comunicação na Arquitetura Cliente-Servidor

Os processos cliente e servidores trocam mensagens segundo um padrão denominado por requisição-resposta (request-response). Isso significa que o cliente envia uma requisição a um servidor que por sua vez, responde com uma resposta.

Essa troca de mensagens é um exemplo de comunicação interprocessos que rege uma boa parte do funcionamento dos sistemas disponíveis na web. A figura a seguir ilustra esse cenário.



Service Design Patterns

05

Um computador servidor é aquele que executa um ou mais programas servidores os quais compartilham seus recursos ou serviços com os clientes. Um cliente não compartilha qualquer dos seus recursos, mas requisita de um servidor recurso ou serviço.

Neste modelo, os clientes são aqueles que iniciam a comunicação com os servidores, os quais aguardam a requisição dos clientes e somente após processá-las, respondem a solicitação. Deste modo, é possível dizer que **o cliente é a parte ativa da comunicação** enquanto que **o servidor é a parte reativa**, ou por vezes também chamado de passiva, deste diálogo.

Uma característica importante do modelo cliente-servidor é que ele descreve um relacionamento cooperativo de programas na execução de um trabalho. O componente servidor provê um recurso ou serviço para um ou mais clientes. Os clientes são aqueles que têm a responsabilidade de iniciar as devidas requisições de modo que possam fazer uso dos recursos ou serviços disponibilizados.

Como forma de ilustrar esses conceitos, imagine dois programas: um servidor web e um navegador web.

O servidor web, como o próprio nome diz é o nosso programa servidor. Esse servidor tem a finalidade de servir, por meio do protocolo HTTP (Hypertext Transfer Protocol), páginas web, que geralmente são documentos escritos utilizando-se a linguagem de programação HTML (HyperText Markup Language). O protocolo HTTP e a linguagem HTML serão abordados oportunamente. Existem diversos programas que funcionam como servidores web e como exemplo de alguns produtos comerciais utilizados atualmente, observe a tabela 1:

Nome Programa	Fabricante
Apache Http Server (Apache)	Apache
Internet Information Server (IIS)	Microsoft
Nginx	NGINX, Inc
Google Web Server (GWS)	Google

Tabela 1

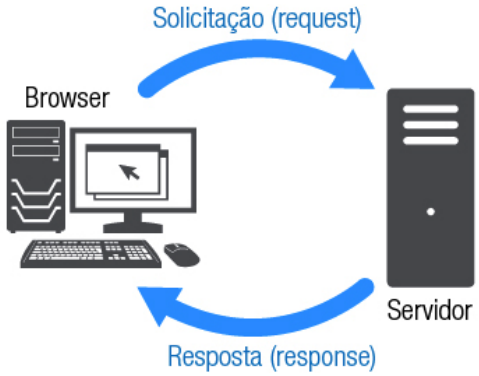
06

O navegador web é o nosso programa cliente, o qual tem a finalidade de consumir as páginas web fornecidas pelo servidor. Existem diversos programas que funcionam como navegadores web e como exemplo de alguns produtos comerciais utilizados atualmente, observe a tabela 2:

Nome Programa	Fabricante
Firefox	Mozzila Foundation
IE	Microsoft
Safari	Apple
Chrome	Google

Tabela 2

Agora, para completar o raciocínio, observe a figura a seguir, que representa um modelo abstrato de alto nível do funcionamento da arquitetura cliente-servidor na WEB.



Requisição/Resposta

07

Portanto, essa disciplina tem por finalidade desenvolver programas clientes ou servidores, ou ambos, para um ambiente WEB. Programas que sejam clientes e servidores simultaneamente, como é o caso dos programas denominados de Proxy, são muito comuns em ambiente de múltiplas camadas, como veremos adiante. É importante frisar que o desenvolvimento dos referidos programas cliente ou servidor exigem o uso de diversas linguagens de programação diferentes que, de algumas formas pré-definidas, se complementam.

Por exemplo, a tabela abaixo exibe um conjunto de linguagens de programação que podem ser utilizadas, geralmente em conjunto, para produzir um programa cliente ou um programa servidor:

Linguagem	Cliente-Servidor
HTML	Lado cliente
JavaScript	Lado cliente *
CSS	Lado cliente
ActionScript	Lado cliente
VBScript	Lado cliente *
ASP	Lado servidor
.Net	Lado servidor
Java	Lado servidor *
PHP	Lado servidor
C	Lado servidor

* As linguagens marcadas com asterisco geralmente são utilizadas conforme descrito na tabela. Porém, em alguns poucos casos, existe a possibilidade dessas mesmas linguagens serem utilizadas no lado contrário da relação. Por exemplo, applets java são escritos para serem executados do lado cliente. NodeJs é um servidor web que executa códigos em JavaScript do lado do servidor. VBScript também pode ser utilizada pelo servidor IIS da Microsoft para construir páginas web dinamicamente.

08

3 - Arquitetura Multicamadas

Em engenharia de *software*, uma arquitetura multicamada, geralmente referenciada como sendo arquitetura n-camadas, é uma arquitetura cliente-servidor.

Essa **arquitetura n-camadas** diz respeito à quantidade de camadas que são utilizadas para dividir o sistema em partes fisicamente separadas.

Naturalmente, a arquitetura cliente-servidor é uma arquitetura de, no mínimo, duas camadas, uma vez que o sistema é dividido em duas partes (dois processos separados) físicas: cliente e servidor.

Porém, existe a possibilidade de dividir o sistema em mais camadas. Uma divisão muito comum neste tipo de arquitetura é aquela que se utiliza de três camadas, principalmente em sistemas WEB. Neste contexto, o sistema é dividido em:

- camada de apresentação,
- camada de lógica do negócio e
- camada de armazenamento de dados.

Um exemplo deste cenário poderia ser o seguinte:

• camada de apresentação,	• a apresentação pode ser escrita em linguagem HTML;
• camada de lógica do negócio e	• a lógica de negócio pode ser escrita em linguagem Java
• camada de armazenamento de dados.	• o armazenamento de dados pode ser escrito em linguagem SQL.

Este cenário acima é um dos objetivos desta disciplina. Ou seja, ao final da disciplina, o aluno deverá ser capaz de construir um sistema que atenda ao cenário descrito.

09

3.1 Tipos de Cliente e Servidor

A arquitetura multicamadas divide o sistema em partes fisicamente separadas. Diante disso, podemos ter, basicamente, dois tipos de clientes: o cliente “magro” (thin/lean/zero/slim client) e o cliente “gordo” (fat/thick/heavy/rich client).

O cliente “magro” é aquele que depende, em sua totalidade, de um servidor para poder funcionar. Isso significa que na ausência de um servidor, o cliente não conseguirá prosseguir na sua execução.

O cliente “gordo” é aquele que depende, parcialmente, de um servidor para poder funcionar. Isso significa que na ausência de um servidor, o cliente conseguirá prosseguir na sua execução apenas naquelas etapas consideradas independentes. Àquelas que são dependentes do servidor ficaram inoperantes.

A determinação se o cliente será “magro” ou “gordo” passa por avaliações técnicas objetivas onde se deve avaliar carga de processamento,*throughput* de rede, latência na comunicação, robustez, segurança, flexibilidade, dentre outros.

Deste modo, fica claro que tal decisão não é trivial, uma vez que muitas das variáveis envolvidas são dinâmicas e inversamente proporcionais. Procurar o equilíbrio é sempre a melhor solução para esses casos complexos.

10

4 - Internet e a World Wide Web

Internet é um sistema global de redes de computadores interligadas que utilizam o conjunto de protocolos padrão da internet (TCP/IP) para servir vários bilhões de usuários no mundo inteiro. É uma rede de várias outras redes, que consiste de milhões de empresas privadas, públicas, acadêmicas e de

governo, com alcance local e global e que está ligada por uma ampla variedade de tecnologias de rede eletrônica, sem fio e ópticas.

World Wide Web (www), ou simplesmente web, é um sistema de informação interligado por meio de documentos de hipertexto (hypertext) que disponibilizam inúmeros conteúdos diferentes tais como texto, imagens, vídeos e outros componentes multimídia.

Essa interligação é feita por meio da chamada “navegação web” que é materializada por meio dos links (hyperlinks), espalhados por diversas partes dos documentos de hipertexto, como conteúdo do mesmo.

A internet traz uma extensa gama de recursos de informação e serviços, tais como serviços de transferência de arquivos (FTP, TFTP), serviços de transferência de hipertexto (HTTP) e sua variante segura (HTTPS), serviços de transferência de e-mail (SMTP, POP, IMAP), serviços de resolução de nomes (DNS), serviços de acesso remoto (telnet, ssh), dentre vários outros. Cada uma dessas siglas utilizadas pelos inúmeros serviços disponibilizados na internet é denominada de protocolo cujo objetivo é estabelecer regras que governam a sintaxe, a semântica e a sincronização da comunicação. Os exemplos anteriores são todos pertencentes a camada de “aplicação” do modelo TCP/IP. Nesta disciplina, apenas o protocolo HTTP será objeto de estudo detalhado mais adiante. O DNS e o HTTPS serão tratados de maneira superficial.

11

RESUMO

Esse módulo tem como objetivo definir o modelo de arquitetura distribuída denominado de cliente-servidor. Além disso, esse módulo demonstra o modelo de comunicação, que se utiliza de mensagens de formato pré-definido denominado de requisição-resposta.

Outro ponto importante é a separação entre os conceitos relativos aos termos Internet e WEB, que popularmente, são utilizados como se sinônimos fossem. Os referidos termos são complementares e compreendê-los adequadamente é fundamental para enxergar suas interfaces. O primeiro é uma rede de computadores enquanto o segundo um sistema de informação.

Como estudado, o modelo cliente-servidor é naturalmente caracterizado como um modelo de arquitetura de duas camadas, uma vez que os processos cliente e servidor são fisicamente separados. Contudo, o modelo de n-camadas é muito utilizado para estruturar os sistemas computacionais presentes no ambiente WEB. Isso significa que vários sistemas presentes, atualmente na WEB, possuem três, quatro, cinco ou inúmeras camadas.

UNIDADE 1 – CONCEITOS FUNDAMENTAIS DA WEB
MÓDULO 2 – HYPERTEXT TRANSFER PROTOCOL (HTTP) – PARTE 1

01

1 - CONCEITOS

Antes de começar a falar sobre o HTTP, é essencial definir alguns termos:

Hipertexto é um texto estruturado exibido em uma tela de computador ou dispositivo eletrônico que possui referências (hiperlinks) para outro texto no qual o leitor poderá acessá-lo imediatamente ou poderá conhecê-lo progressivamente, ao longo do tempo, em múltiplos níveis de detalhes.

Hipermídia é uma extensão de um hipertexto que possui acesso não linear.

Por **acesso não linear**, temos inúmeros outros exemplos não computacionais como um catálogo de endereços, de telefones, dicionários, dentre outros. Isso significa que o acesso ao texto não obedece a uma linearidade com começo, meio e fim. Ou seja, a menor parte possível da informação representada é caracterizada como sendo completa por si só (autocontida).

Diante disso, o protocolo de transferência de hipertexto, cuja sigla é HTTP, é um protocolo pertencente à camada de aplicação do modelo TCP/IP utilizado pelos sistemas de informação de hipermídia distribuídos e colaborativos presentes na WEB (HTTP). O HTTP é um protocolo baseado no modelo de requisição-resposta, conforme descrito anteriormente. Esse protocolo é um dos pilares de sustentação da World Wide Web, na sua forma atual.

02

O protocolo HTTP é o resultado de um esforço coordenado, basicamente, entre o World Wide Web Consortium (W3C) e o Internet Engineering Task Force (IETF). Esse esforço produziu um documento conhecido como request for comments (RFC) de número 2616 (RFC 2616). Essa RFC2616 de junho de 1999 definiu o protocolo HTTP/1.1 (versão 1.1), o qual é utilizado em larga escala até os dias de hoje.

O HTTP/1.1 é uma revisão do HTTP/1.0 e do HTTP/0.9. Sua principal evolução é a criação das conexões persistentes (keep-alive). Nas versões anteriores a 1.1, o protocolo somente conseguia executar um par de requisição/resposta para cada conexão TCP enquanto que, na versão atual, o protocolo permite executar inúmeros pares de requisição/resposta para cada conexão TCP. Esse avanço melhora em muito a eficiência de transmissão de dados do protocolo.

Existem dois recursos básicos que fazem do HTTP um protocolo simples e extremamente poderoso:

Stateless

É um protocolo sem estado, o que significa que cada par de requisição/resposta é uma transação independente e não relacionada a qualquer par de requisição/resposta anterior. Do ponto de vista do servidor, cada par de requisição/resposta que chega até ele é considerado, sempre, como se fosse o primeiro par.

Independente de Mídia

Significa que qualquer tipo de dado pode ser transmitido desde que, tanto o cliente quanto o servidor, saibam como interpretar o tipo de dado que está sendo transportado.

03

2 - Elementos Essenciais

a) URI, URL e URN

Referência (*reference*) é uma relação entre objetos na qual um deles é responsável por designar ou agir como sendo o significado que o interliga ao outro objeto.

Metadado (*metadata*) é um dado que descreve outro dado.

Os metadados são divididos em duas partes complementares:

Estrutural	Descritivo
Que descreve o design e a especificação da estrutura que irá conter os dados.	Que descreve uma instância individual dos valores dos dados da aplicação, ou seja, o conteúdo dos dados, ou o dado de fato.

Os conceitos são abstratos e podem ser aplicados em inúmeras situações. Um bom exemplo de materialização destes conceitos é aquele associado ao endereçamento utilizado pela Empresa Brasileira de Correios e Telégrafos (ECT), ou simplesmente Correios. Essa empresa utiliza algumas informações para poder realizar a logística de entrega de produtos no território brasileiro. Essas informações são divididas em dois grupos: endereçamento do remetente e endereçamento do destinatário. O endereçamento do destinatário, por exemplo, deve respeitar a seguinte forma e ordem:

a) Nome do Destinatário;
b) Tipo do Logradouro + Nome do Logradouro + Número do Lote + Complemento (se houver);
c) Nome do Bairro;
d) Nome da Localidade + Sigla da Unidade da Federação;
e) CEP.

04

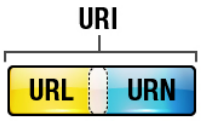
Portanto, podemos dizer que o endereçamento do destinatário é um exemplo de referência, pois este objeto, endereçamento do destinatário, descreve o significado daquele objeto, destinatário (ser humano/pessoa de facto) propriamente dito, que deverá receber o produto em seu destino.

Um ponto importante que se deve perceber é que a referência é um metadado, ou seja, possui estrutura (forma e ordem) bem como sua descrição (valores dos campos).

No que diz respeito ao desenvolvimento WEB, os termos URI, URL e URN são também considerados referências ou metadados. Abaixo a definição de cada termo:

- URI: Uniform Resource Identifier (Identificador uniforme de Recurso);
- URL: Uniform Resource Locator (Localizador Uniforme de Recurso);
- URN: Uniform Resource Name (Nome Uniforme de Recurso).

Apesar de atualmente, o termo URL ainda ser utilizado com maior frequência que o termo recomendado URI, iremos considerá-los como sinônimos na maioria dos casos referentes a essa disciplina, apesar do termo URL ser considerado um subconjunto do termo URI. Abaixo, uma figura que ilustra a relação entre os três termos:



URI: Uniform Resource Identifier (Identificador uniforme de Recurso)

É uma cadeia compacta de caracteres usada para identificar um recurso físico (RFC3986.). O principal propósito desta identificação, não se resumindo a este, é permitir a interação com representações do recurso através de uma rede, tipicamente a internet, usando protocolos específicos. URI's são identificados em grupos definindo uma sintaxe específica bem como seus protocolos associados.

URL: Uniform Resource Locator (Localizador Uniforme de Recurso)

É uma referência para um recurso que especifica a localização do recurso em um computador de rede bem como o mecanismo para recuperá-lo. Popularmente, as URL são conhecidas como endereços WEB. (RFC3305)

URN: Uniform Resource Name (Nome Uniforme de Recurso)

É aquele que identifica um recurso pelo nome sem dizer a sua localização ou como acessá-lo. O conceito relativo a URN foi formalmente definido em 1997, e em 2005, para evitar confusão com o termo URL, ambos os termos URN e URL foram desaconselhados em favor do termo mais flexível, abstrato e genérico denominado de URI. (RFC3986)

05

As URIs seguem uma especificação predefinida na RFC3986 (http://tools.ietf.org/html/rfc3986). Uma sintaxe genérica da URI consiste em uma sequência hierárquica de componentes referenciadas como o scheme, authority, path, query e fragmet.

```
URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part = "//" authority path-abempty
                / path-absolute
                / path-rootless
                / path-empty
```

Um exemplo de aplicação da forma genérica, seria o endereço web (fictício) abaixo:

```
http : //www.aiec.br/over/there?name=ferret#nose
\_/ \_____/\_____/ \_____/ \__/
  |       |       |       |    |
scheme  authority  path    query  fragment
```

Outros exemplos de URI, não necessariamente voltados para web, estão descritos abaixo:

ftp://ftp.is.co.za/rfc/rfc1808.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2

Desta forma, pode-se perceber que um dos objetivos principais da existência de URI é evitar que dois ou mais exemplares do mesmo recurso possam ter nomes diferentes e, reciprocamente, que recursos diferentes possam ter nomes idênticos.

06

b) Domain Name System (DNS)

O DNS, Sistema de Nomes de Domínio, é um sistema de gerenciamento de nomes distribuído e hierárquico para computadores, serviços ou qualquer recurso conectado a uma Internet ou uma rede privada. Esse protocolo também é baseado em um modelo de requisição-resposta, assim como o HTTP.

O servidor de DNS tem por finalidade precípua traduzir nomes URI em endereços IP. Além disso, o servidor DNS também consegue realizar o processo inverso, ou seja, traduzir endereços IP em seus respectivos nomes URI. Esta última função também é conhecida como DNS reverso.

O DNS é um protocolo de camada de aplicação do modelo TCP/IP que utiliza, primariamente e por padrão, a porta 53 do protocolo UDP (User Datagram Protocol) da camada de transporte para responder as requisições que lhe são enviadas. Porém, quando o servidor DNS tem que enviar uma resposta cujo tamanho exceda 512 bytes ou para transferências de zona, o mesmo utiliza a porta 53 do protocolo TCP (Transmission Control Protocol).

Transferências de zona de domínio são mecanismos utilizados pelos administradores deste serviço para replicarem a base de dados de DNS entre os servidores distribuídos pela rede que oferecem esse tipo de serviço.

07

3 - Elementos Acessórios

HTTPS (HTTP + SSL/TLS)

O protocolo HTTPS (*Hypertext Transfer Protocol Secure*) ou, protocolo de transferência de hipertexto seguro, oferece suporte a troca de mensagens entre o cliente e o servidor de forma segura.

Isso significa que as mensagens são criptografadas visando garantir a integridade e a confidencialidade das mesmas. Além disso, o uso do HTTPS garante a autenticidade da comunicação estabelecida entre o cliente e o servidor.

Esse protocolo não deve ser considerado um protocolo novo, uma vez que o mesmo apenas adiciona uma camada extra sob o HTTP. Ou seja, o HTTPS é o próprio protocolo HTTP acrescido de uma camada de segurança denominada de SSL (*Secure Sockets Layer*), atualmente renomeada para TLS (*Transport Layer Security*). O acréscimo desta camada extra, obriga, necessariamente, a necessidade de se utilizar uma outra porta TCP para essa forma de comunicação. Deste modo, o protocolo HTTPS utiliza, por padrão, a porta TCP de número 443 ao invés da porta padrão TCP de número 80 utilizada pelo protocolo HTTP.

08

Resumo

O objetivo deste módulo é compreender os fundamentos que envolvem o protocolo HTTP bem como seus pilares de sustentação e sua relação com os protocolos DNS e HTTPS.

Os protocolos HTTP, HTTPS e DNS são protocolos baseados no modelo de mensagens denominado de requisição-resposta. Isso quer dizer que o processo cliente é a parte ativa da comunicação e o processo servidor é a parte reativa, ou por vezes chamado de passiva, da relação.

Além disso, as URIs são consideradas elementos essenciais de um sistema WEB, uma vez que têm a responsabilidade de identificar, de maneira inequívoca, recursos existentes na rede. As URIs são elementos de extrema relevância para o funcionamento da WEB bem como para sua popularização, uma vez que tornou o uso da WEB mais simples. Podemos dizer que o poder da URI associado aos protocolos de DNS e HTTP formam três dos quatro pilares associados aos sistemas WEB. Além desses, a linguagem HTML, que será objeto de estudo mais adiante, é considerada o quarto pilar de sustentação desse ambiente computacional.

UNIDADE 1 – CONCEITOS FUNDAMENTAIS DA WEB
MÓDULO 3 – HYPERTEXT TRANSFER PROTOCOL (HTTP) – PARTE 2

01

1 - MENSAGENS

Um cliente HTTP é um processo, geralmente um navegador web, que estabelece uma conexão com o processo servidor com o propósito de enviar uma ou várias mensagens de requisição.

Um servidor HTTP é um processo, geralmente um web server, que aceita requisições na ordem em que elas vão chegando e enviando uma mensagem de resposta para cada requisição feita. Não se esqueça de que na versão 1.1 do protocolo, a relação de cardinalidade entre a requisição e a resposta é de um para um, ou seja, **para cada requisição existe uma única resposta**.

O protocolo HTTP utiliza uma URI para identificar um recurso e estabelecer uma conexão com o mesmo. Uma vez estabelecida a conexão, as mensagens HTTP seguem o padrão da RFC5322 (Internet Mail) e da RFC 2045 (Multipurpose Internet Mail Extension - MIME).

As mensagens são descritas como **requisição** (*request*) e **resposta** (*response*). A requisição é a mensagem enviada do cliente para o servidor enquanto que a resposta é a mensagem enviada do servidor para o cliente.

As mensagens HTTP usam um formato de mensagem genérica descrito na RFC 822 para transferência do dado requerido. Essa mensagem genérica consiste em quatro itens:

I- Start-line
II - Header-fields
III - Empty-line
IV - Message-body (opcional)

Vejamos a seguir cada item.

02

I - Start-line

Essa linha possui uma sintaxe genérica que pode ser ou apenas requisição ou apenas resposta (status).

```
start-line = Request-Line | Response-Line (Status-Line)
```

Veja abaixo um exemplo de cada um desses possíveis valores:

```
GET /hello.htm HTTP/1.1 (um possível valor de request-line)
HTTP/1.1 200 OK (um possível valor de status-line)
```


II- Header-fields

Os campos de cabeçalho fornecem as informações necessárias ou sobre a requisição ou sobre a resposta. Além disso, caso exista, fornece informações sobre o corpo da mensagem também. Existem quatro tipos de campos de cabeçalhos:

1. General-header;
2. Request-header;
3. Response-header;
4. Entity-header.

Todos os campos de cabeçalhos presentes nos quatro grupos descritos anteriormente seguem o mesmo formato genérico de campo. O formato segue a sintaxe geral:

```
message-header=field-name“ : ”[ field-value ]
```

Essa sintaxe descreve um nome seguido de um sinal de dois pontos (:) e em seguida pelo seu valor.
Alguns exemplos estão demonstrados abaixo:

```
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain
```

General-header
Campos aplicáveis tanto à requisição quanto para a resposta.

Request-header
Campos aplicáveis exclusivamente à requisição.

Response-header
Campos aplicáveis exclusivamente à resposta.

Entity-header
Campos que definem metadados sobre o corpo da mensagem, caso exista.

04

III - Empty-line

Uma linha vazia. Essa linha tem por finalidade precípua, separar os campos de cabeçalho do corpo da mensagem.

IV - Message-body

O corpo da mensagem é uma parte opcional. Isso significa que ele pode estar presente ou não. Quando estiver presente, é utilizada para levar os dados tanto da requisição do cliente quanto da resposta do servidor. Nestes casos, os campos de cabeçalho ***content-type*** e ***content-length*** devem ser especificados para dizer o tipo e o tamanho do conteúdo associados. Isso é fundamental, pois, como o protocolo HTTP é independente de mídia, é através desses campos que o cliente e o servidor conseguem interpretar adequadamente a informação.

Abaixo, dois possíveis exemplos de dados para o corpo da mensagem de uma requisição e de uma resposta, respectivamente:

Request

```
Faculdade=AIEC&Curso=AnaliseSistemas&Disciplina=ProgramacaoWEB
```

Response

```
<html>

<body>

<h1>Bem vindo ao Curso de Programação WEB da AIEC!</h1>

</body>

</html>
```

05

2 - Mensagem de Requisição

Uma mensagem de requisição é aquela enviada pelo cliente ao servidor e segue o seguinte formato:

```
I- Request-line

II - Headers (General|Request|Entity) fields

III- An empty line

IV- Message-body (opcional)
```

I- Request-line

Essa linha começa especificando três campos: method, request-URI e http-version. Esses valores devem ser separados por um único caractere de espaço(SP). A fórmula genérica desses campos está descrita abaixo:

```
Request-Line = Method SP Request-URI SP HTTP-Version
```

A seguir a explicação de cada um dos campos.

06

I.I Method

O método de requisição indica o mecanismo de transferência que será usado para se estabelecer a comunicação com o servidor apontado pela request-uri. O nome do método é case-sensitive e sempre deve ser escrito em letras maiúsculas.

Abaixo, segue os nomes, bem como uma breve explicação sobre cada um dos oito métodos suportados no HTTP/1.1:

Nome do Método	Descrição
GET	É utilizado para recuperar uma representação específica da informação localizada no servidor. Esse método não deve produzir qualquer tipo de efeito sobre os dados.
HEAD	Correspondente ao GET, exceto pela ausência do message-body. Este método é muito usado para recuperar apenas os metadados da resposta sem qualquer tipo de conteúdo.
POST	É utilizado para enviar dados ao servidor. Por exemplo, upload de arquivos, formulários html, itens de banco de dados, dentre outros.
PUT	É utilizado para armazenar os dados no servidor. Caso a URI se refira a um recurso existente, o mesmo será substituído. Caso contrário, será criado.
DELETE	É utilizado para remover os dados do servidor.
CONNECT	É utilizado para estabelecer um túnel com o servidor. Por exemplo, conexões criptografadas por meio do protocolo SSL (*Secure Socket Layer*)
OPTIONS	É utilizado para retornar uma descrição dos métodos HTTP suportados pelo servidor.
TRACE	Executa um teste de mensagem de loopback no servidor.

07

I.II Request-URI

Identifica o recurso sobre o qual será feita a requisição.

O mais comum é o uso deste campo para identificar um recurso no servidor de destino para o qual a requisição está sendo feita. Por exemplo, um cliente que deseja obter o arquivo index.html do servidor, deverá construir a seguinte linha:

```
GET /index.html HTTP/1.1
```

I.III HTTP-Version

Esse campo indica a versão do protocolo HTTP que está sendo utilizada pelo cliente. O mais comum é o uso da versão 1.1, que atualmente, é a versão corrente do referido protocolo.

08

II- Headers (General|Request|Entity)

Essa linha pode conter uma série de headers utilizados pelo cliente para informar ao servidor sobre a requisição. Abaixo, segue uma lista com os headers considerados mais importantes:

Nome Header	Descrição	Exemplo	Status
Accept	Tipo de conteúdo	Accept: text/plain	permanente
Accept-Charset	Conjunto de caracteres aceito	Accept-Charset: utf-8	permanente
Accept-Encoding	Lista de compactação permitida	Accept-Encoding: gzip, deflate	permanente
Accept-Datetime	Tempo	Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT	permanente
Authorization	Credenciais de autenticação para o HTTP conteúdo	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==	permanente
Connection	Opção de controle para a conexão corrente	Connection: keep-alive Connection: Upgrade	permanente
Cookie	Um cookie http enviado anteriormente ao cliente	Cookie: $Version=1; Skin=new;	permanente
Content-Length	O tamanho do request-body	Content-Length: 348	permanente
Content-Type	O MIME-Type presente no request-body	Content-Type: application/x-www-form-urlencoded	permanente
Host	O domínio do servidor e a porta TCP na qual o processo esteja escutando (listening)	Host: www.aiec.br:80 Host: www.aiec.br	permanente
User-Agent	Um string identificando o cliente	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0	permanente

09

Abaixo, segue um exemplo do uso de uma requisição HTTP que solicita o recurso “index.html” do servidor “www.aiec.br”

```
GET /index.html HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.aiec.br
```

```
Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive
```

III - Empty-Line

Já descrito anteriormente.

IV - Message Body

Já descrito anteriormente.

10

3- Mensagem de Resposta

Após receber e interpretar uma requisição, o servidor responde com uma mensagem HTTP com o seguinte formato genérico:

```
I- Status-line (Response-Line)
II - Headers (General|Response|Entity) fields
III- An empty line
IV- Message-body (opcional)
```

I - Status-Line

Essa linha consiste de três campos: versão do protocolo, código de status e uma breve descrição associada ao código de status. Os campos são separados pelo caractere de espaço em branco.

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase
```

HTTP-Version

Esse campo indica a versão do protocolo HTTP que está sendo utilizada pelo servidor. O mais comum é o uso da versão 1.1, que atualmente, é a versão corrente do referido protocolo.

Status-Code

O status-code é um inteiro de 3 algarismos onde o primeiro algarismo define a gênero/classe da resposta e os dois últimos algarismos definem a espécie.

Gênero/Classe	Descrição
1	1xx: Informação
2	2xx: Sucesso
3	3xx: Redirecionamento
4	4xx: Erro do Cliente
5	5xx: Erro do Servidor

Reason-Phrase

Esse campo está diretamente associado ao status-code. Abaixo, alguns exemplos de descrições associadas aos seus respectivos status-code:

Status-Code	Reason-Phrase
101	switching protocols
200	ok
201	created
202	accepted
301	moved permanetly
305	use proxy (since HTTP/1.1)
400	bad request
404	not found
500	internal server error
501	not implemented
502	bad gateway

II- Headers (General|Response|Entity)

Essa linha pode conter uma série de headers utilizados pelo servidor para informar ao cliente sobre a resposta. Abaixo, segue uma lista com os headers considerados mais importantes:

Nome Header	Descrição	Exemplo	Status
Accept-Patch	Especifica que tipo de formatos de documentos o servidor aceita	Accept-Patch: text/example;charset=utf-8	permanente

Age	A idade, em segundos, de um objeto que passou por um cache de proxy	Age: 12	permanente
Allow	Actions válidas para um recurso específico	Allow: GET, HEAD	permanente
Accept-Ranges	A forma de transferência de conteúdo suportada pelo servidor	Accept-Ranges: bytes	permanente
Cache-Control	Diz a todos os mecanismos de cache presentes entre o servidor e o cliente, o tempo máximo, em segundos, que o objeto pode ser armazenado	Cache-Control: max-age=3600	permanente
Connection	Controle de opções para a conexão corrente	Connection: close	permanente
Content-Encoding	O tipo de compactação utilizado nos dados	Content-Encoding: gzip	permanente
Content-Language	A linguagem ou linguagens do conteúdo	Content-Language: pt_BR	permanente
Content-Length	O tamanho do response-body	Content-Length: 348	permanente
Content-Type	O MIME-type do response-body	Content-Type: text/html; charset=utf-8	permanente
E-Tag	Um identificador exclusivo do recurso (hash function)	ETag: "737060cd8c284d8af7ad3082f209582d"	permanente
Server	O nome do servidor	Server: Apache/2.4.1 (Unix)	permanente
Set-Cookie	Um cookie HTTP	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1	permanente
Status	O status-code do HTTP	Status: 200 OK	permanente

12

Abaixo, segue um exemplo do uso de uma resposta HTTP que informa o recurso solicitado:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Accept-Ranges: bytes
Connection: close

<html>
<head>
<title>AIEC</title>
</head>
<body>
Página principal do site da AIEC. Seja bem-vindo!
</body>
</html>
```

III - Empty-Line

Já descrito anteriormente.

IV - Message Body

Já descrito anteriormente.

13

RESUMO

O objetivo deste módulo é compreender os detalhes pré-definidos que envolvem as mensagens de requisição e resposta, ou seja, as mensagens que o cliente envia (requisição) ao servidor e aquelas em que o servidor devolve (resposta) ao cliente.

As mensagens HTTP usam um formato de mensagem genérica descrito na RFC 822 para transferência do dado requerido. Essa mensagem genérica consiste em quatro itens: Start-line, Header-fields, Empty-line, Message-body (opcional). O referido formato, apesar de pré-definido, possui certa flexibilidade com relação a sua composição, uma vez que determinados campos, bem como seus respectivos valores são considerados opcionais.

UNIDADE 1 – CONCEITOS FUNDAMENTAIS DA WEB
MÓDULO 4 – HYPERTEXT MARKUP LANGUAGE (HTML)

01

1- FUNDAMENTOS

A linguagem HTML (HyperText Markup Language) foi criada por Tim Berners-Lee em meados de 1991, mas somente em 1995 a primeira especificação padronizada da linguagem, a HTML 2.0, foi realizada. A versão 4.01 da linguagem, publicada em 1999, foi a versão, até hoje, utilizada pelo maior período de tempo.

Atualmente, a linguagem HTML já se encontra na versão 5.0, que foi publicada no ano de 2012. Essa última versão é considerada uma extensão da versão 4.01. O primeiro site construído pelo autor utilizando a referida linguagem foi disponibilizado no seguinte endereço:http://info.cern.ch/hypertext/WWW/TheProject.html

Atualmente, a construção de interfaces para sistemas web utiliza-se de várias tecnologias disponíveis, algumas classificadas como sendo padrões abertos (domínio público) e outras como **padrões fechados** (domínio proprietário). O World Wide Web Consortium (W3C) é o órgão internacional responsável por definir os **padrões abertos** para as tecnologias envolvidas na construção de sistemas web. Especificamente para construção das interfaces web, o **W3C padroniza três linguagens de programação** cujos nomes são:

- HyperText Markup Language (HTML);
- Cascading Style Sheets (CSS)
- JavaScript.

Cada uma dessas linguagens cumpre um objetivo diferente e complementar na construção de interfaces web. O HTML é responsável pelo conteúdo, enquanto que o CSS é responsável pelo formato de apresentação, restando ao JavaScript manipular o comportamento existente na relação criada entre o conteúdo e sua forma. Neste momento, apenas a linguagem HTML será objeto de estudo. As demais são importantes e complementares no estudo de programação web, porém não serão objeto de estudo deste curso.

Como o próprio nome sugere, HTML é uma linguagem de marcação, que se utiliza de elementos denominados de tags, que servem para estruturar o conteúdo e dizer, geralmente, aos navegadores web, como o referido conteúdo deve ser interpretado e apresentado na interface (geralmente a tela/monitor do computador, utilizada pelo usuário do sistema).

02

Nosso objetivo neste momento é descrever alguns fundamentos necessários à compreensão de todo o curso de programação. Deste modo, antes de começarmos a descrever a linguagem HTML, é necessário definirmos alguns termos:

- *Símbolo*
- *Alfabeto*
- *Gramática*
- *Elemento*
- *Léxico*
- *Sintaxe*
- *Semântica*

Todos os termos serão exemplificados ao longo do nosso estudo, de modo a torná-los concretos àqueles que os leem. Por ora, apenas, três serão exemplificados: símbolo, alfabeto e elemento. Veja a seguir.

Símbolo
Tem a função de representar algo abstrato por força de convenção, semelhança ou contiguidade semântica.

Alfabeto
Um conjunto de símbolos discretos (finitos) definidos em uma língua.

Gramática
É o estudo dos símbolos e elementos de uma linguagem bem como suas possíveis combinações.

Elemento
É a união de símbolos para materialização do pensamento humano em termos significativos.

Léxico
É o acervo de símbolos e elementos de uma determinada linguagem.

Sintaxe
É o estudo das regras que envolvem a escrita e a disposição dos elementos (possíveis combinações) na construção do código.

Semântica
É o estudo dos significados dos elementos levando-se em consideração sua disposição.

03

Um **símbolo** pode ser exemplificado como sendo as letras do alfabeto como A, B, ou F, por exemplo. Já um alfabeto, especifica a quantidade e o domínio de valores permitidos como, por exemplo, o **alfabeto** latino, que possui 26 letras começando em A e finalizando em Z. Um**elemento** pode ser exemplificado como sendo as palavras, tais como *manga, sabedoria,* que são um conjunto de símbolos.

Os exemplos são úteis para criar uma relação de significado entre novos conceitos e aqueles já existentes. Isso significa que é fundamental que os exemplos sejam vistos como uma possibilidade dentre as várias, inúmeras, infinitas situações nas quais os referidos termos supracitados possam ser aplicados. O aluno que se utiliza de um único exemplo como sendo a única forma de materializar os conceitos teóricos está correndo um grande risco de impor limitações significativas ao seu raciocínio. Deste modo, o exemplo deve ser percebido como uma possibilidade dentre inúmeras outras existentes.

As definições dos termos anteriores nos ajudarão a compreender melhor a linguagem de programação HTML.

Um fato importante que deve ser considerado e que não pode ser confundido é que um determinado símbolo ou elemento, sintaticamente igual ou diferente, pode vir a ter significado, igual ou diferente, quando comparado entre duas ou mais linguagens.

Por exemplo, na língua portuguesa, o elemento (palavra) “maior” é sintaticamente diferente do elemento (sinal) “>” da linguagem matemática. Apesar de serem sintaticamente diferentes, esses dois elementos são iguais semanticamente, ou seja, possuem o mesmo significado. Outra situação que se pode encontrar diz respeito ao elemento (acento) circunflexo “^” na língua portuguesa e o elemento (sinal) circunflexo “^” na matemática. Apesar dos elementos serem sintaticamente idênticos, a sua semântica é diferente, ou seja, na língua portuguesa significa um acento fonético enquanto que na língua matemática significa uma operação aritmética de potenciação.

04

1.1 Linguagens Compiladas, Interpretadas e Híbridas

A diferenciação entre esses três conceitos é essencial para compreender qualquer linguagem de programação, tanto a linguagem HTML, objeto de estudo deste módulo, como as linguagens JAVA e SQL que serão objeto de estudo mais adiante, ainda nesta disciplina.

São aquelas em que um código fonte é traduzido em linguagem de máquina em momento anterior à execução do programa.

São aquelas em que um código fonte é traduzido em linguagem de máquina durante a execução do programa.

São aquelas que se utilizam das duas técnicas anteriores, procurando amplificar as vantagens de cada uma, bem como reduzir as desvantagens de ambas.

A essência que separa os dois conceitos, compilação e interpretação, está relacionada ao momento no tempo em que a tradução do código fonte para linguagem de máquina é realizada. Além disso, essa essência produz consequências (vantagens e desvantagens) para ambas as formas.

05

As **linguagens compiladas** produzem como resultado do processo de compilação um arquivo binário executável. Isso significa que o código fonte do programa é desnecessário e dispensável para cada execução do programa, bastando para tanto a existência apenas do arquivo binário executável. Como exemplo de linguagem compilada temos a linguagem C, cujos compiladores produzem arquivos binários executáveis para diferentes plataformas/sistemas operacionais. Por exemplo, os formatos PE, PE32+, ELF, PEF, são tipos de arquivos binários executáveis. Os formatos PE (32 bits) e PE32+ (64 bits) são utilizados pelos sistemas operacionais Windows da Microsoft cujo arquivo binário executável possui a extensão ".exe". A execução de programas a partir de binários executáveis é extremamente rápida em função das instruções binárias, inteligíveis para o computador, já estarem prontas.

As **linguagens interpretadas** não produzem arquivos binários executáveis. Isso significa que o código fonte do programa é necessário e indispensável para cada execução do programa, ou seja, o código fonte sempre deve estar disponível para que o programa possa ser executado. Neste tipo de linguagem, as sentenças vão sendo traduzidas em sequências de uma ou mais sub-rotinas (procedimentos, funções, métodos) já compiladas previamente em linguagem de máquina. Ou seja, à medida que as sentenças vão sendo lidas pelo interpretador, elas vão sendo traduzidas e, consequentemente, executadas. Como exemplo deste tipo de linguagem temos a linguagem HTML. Saiba+

As **linguagens híbridas** procuram utilizar o melhor da interpretação com o melhor da compilação procurando potencializar as vantagens e reduzir as desvantagens. Um bom exemplo de linguagens que se utilizam desta técnica são aquelas classificadas como multiplataformas como, por exemplo o JAVA, o C#, o PHP, dentre outras.

Uma observação importante que deve ser percebida é que o **código fonte sempre é essencial**, independente se a linguagem é compilada, interpretada ou híbrida. Apesar de nas linguagens compiladas, o código fonte não ser necessário para a execução do programa, o mesmo continua sendo

essencial para todo o restante, como evolução do programa, correção do programa, extensão do programa, melhorias do programa, dentre outras. Deste modo, jamais apague o código fonte de um programa.

ELF
O formato ELF é um formato para sistemas operacionais do tipo Unix-like, como o Linux, por exemplo.

PEF
O formato PEF é muito utilizado pelo sistema operacional MAC OS da Apple para representação de binários executáveis.

Saiba+
Os navegadores web são considerados interpretadores da linguagem HTML. Além disso, a execução de programas a partir do código fonte introduz um atraso considerável na sua execução em função das instruções não estarem representadas em sequencias binárias, inteligíveis para o computador, ou seja, as instruções ainda não estão prontas para serem executadas.

06

2 - Elementos e Atributos da Linguagem

A linguagem HTML, HyperText Markup Language, como o próprio nome diz, é uma **linguagem de marcação de hipertexto**. Diferentemente de outras linguagens de programação como JAVA e C, por exemplo, a HTML **não possui** as seguintes funcionalidades:

- operadores aritméticos (somar, subtrair, multiplicar, dividir, potenciação e radiciação)
- operadores lógicos (and, or, not)
- operadores relacionais (menor, maior, igual, diferente)
- instruções de decisão (if, if-else, switch-case)
- instruções de repetição (for, while, do-while)
- tipo de dado (boolean, char, int, double)

Devido à ausência destas funcionalidades, a linguagem HTML é considerada, por alguns autores, como sendo apenas uma linguagem de marcação e não uma linguagem de programação. Apesar desta divergência existente na literatura, ela será considerada, apenas nos casos que a referida controvérsia for objeto de questionamento de forma direita. Para todos os demais casos, esse módulo irá considerar a linguagem HMTL como sendo uma linguagem de programação com limitações de representação.

Deste modo, a utilização da linguagem HTML é feita apenas para marcar o texto que representa o **conteúdo** de um documento web. Essa marcação é feita através de **elementos** que especificam sua estrutura. Existem diversos elementos HTML que serão demonstrados ao longo do módulo, porém antes de começar a estudar os elementos de modo específico, é importante conhecer a estrutura genérica de um elemento.

07

Geralmente, um elemento HTML é delimitado por tags ou marcas de **abertura** e de **fechamento**. Apesar de existirem elementos que possuem apenas tag de abertura, o uso dessa forma será evitado sempre que possível.

Desta maneira, esse módulo irá utilizar, em sua maior parte, elementos com tags de abertura e fechamento.

Uma **tag de abertura** é constituída por um símbolo de menor (<), um nome de tag (tagname) e um símbolo de maior (>).

Já uma **tag de fechamento** é constituída por um símbolo de menor (<), uma barra simples (/), o nome da tag (tagname) e um símbolo de maior (>).

Essas duas tags, abertura e fechamento definem um **conteúdo** do elemento. A fórmula geral é mostrada abaixo:

```
<tagname>
Content
</tagname>
```

Essa fórmula geral poderia ser escrita em uma única linha, da seguinte forma:

```
<tagname> Content </tagname>
```

Ambas as formas, uma linha ou múltiplas linhas, são válidas e utilizadas com bastante frequência na escrita de documentos web.

08

Não se esqueça: o símbolo de menor e maior utilizado pelos elementos da linguagem HTML não possuem qualquer significado com suas representações em língua portuguesa ou em língua matemática. No HTML, esses símbolos, popularmente conhecidos como sendo maior (>) e menor (<), são combinados para formarem um elemento denominado **tag**, que é a representação sintática genérica do acervo de elementos da linguagem em estudo.

Ao longo do módulo, será percebido que os nomes de tags irão variar conforme o significado que cada um se propõe, mas todos terão essa mesma estrutura tipográfica.

Observe que a tag de fechamento se diferencia da tag de abertura apenas pela presença da barra simples (/) entre o sinal de menor (<) e o nome da tag (tagname). Ou seja, um elemento HTML é definido por duas tags, sendo uma de abertura e uma de fechamento que possuem o mesmo nome e se diferenciam pela ausência ou presença, respectivamente, da barra simples (/).

Esquecer de diferenciar as tags de abertura e fechamento através do símbolo de barra simples (/) é um **erro de sintaxe**.

Observe também que o nome das tags deve ser escrito em **caracteres minúsculos**, o que significa que escrevê-los em caracteres maiúsculos é um erro de sintaxe.

09

Muitas tags de abertura possuem **atributos** cuja finalidade é oferecer informações adicionais sobre um elemento. Esses atributos podem ser considerados opcionais ou obrigatórios. Além disso, cada atributo possui um **nome** bem como um respectivo **valor ou conjunto de valores**, separados por um sinal de igualdade (=). Além disso, os atributos são informados após o nome da tag (tagname) de abertura e antes do sinal de maior (>). Cada valor de um atributo deve ser informado entre aspas duplas ou simples e cada par de atributo/valor deve ser separado do seu antecessor ou sucessor por um caractere de espaço (espaço em branco). A fórmula geral seria da seguinte forma:

```
<tagname atributo1=”valor1” atributo2=”valor2” ... atributoN=”valorN”>
Content
</tagname>
```

Apesar de existir a possibilidade de se utilizar alguns atributos sem valor, essa funcionalidade será evitada, sempre que possível, e desta maneira, geralmente, os exemplos deste módulo se utilizará de atributos com seus respectivos valores.

Todos os códigos fontes descritos ao longo do módulo devem ser digitados em editores de texto ASCII puro (plain-text), como por exemplo, o bloco de notas, notepad++, dentre outros. Esses editores são, geralmente, simples e armazenam apenas o código binário equivalente a cada caractere utilizado em seu conteúdo. Processadores de texto, como o Word (Microsoft Office), Writer (LibreOffice), jamais devem ser utilizados pois eles armazenam vários códigos binários além daquele caractere específico, como códigos relacionados a formatação do texto (tipo de fonte, tamanho da fonte, estilos, cor da fonte, dentre outros).

10

3 - CLASSIFICAÇÃO DOS ELEMENTOS DE UM DOCUMENTO HTML

Um documento HTML é definido pelo uso do elemento *<html>*. O elemento *<html>* é dividido em duas seções, denominado de cabeçalho e corpo. O cabeçalho é definido pelo elemento *<head>* e o corpo pelo elemento *<body>*. Desta forma, abaixo podemos demonstrar uma estrutura típica de um documento web:

```
<!DOCTYPE html>
<html>
<head>
<title>Programação WEB - AIEC!</title>
</head>
<body>
<p>Bem vindo ao módulo 4 da unidade 1!</p>
</body>
</html>
```

Os elementos <title> e <p> servem respectivamente para definir o título do documento e um parágrafo como seu conteúdo. O título aparece no canto superior esquerdo da janela do seu navegador, seja na área da guia de navegação ou na barra de título, neste último caso para os navegadores que não oferecem suporte a guias. Já o parágrafo, aparece na área de conteúdo do navegador, conforme pode ser observado na imagem abaixo. O código acima pode ser digitado em um editor de texto ASCII puro, por exemplo, como o bloco de notas, e pode ser salvo em um arquivo cuja extensão seja, por exemplo, ".html". Depois basta abrir o arquivo, com um clique duplo, por exemplo, no seu navegador web padrão já instalado em seu computador para visualizar um resultado semelhante ao exibido pela tela abaixo:

11

A classificação por categoria separa os principais elementos da linguagem em vários grupos. Alguns grupos, bem como seus respectivos elementos, que serão tratados no módulo são:

- básicos,
- formatação,
- formulários e entradas,
- imagens,
- links,
- listas e
- tabelas.

Abaixo, uma tabela com os elementos mais comuns utilizados para construir documentos web.

3.1 Elementos Básicos

Tag	Descrição
<!DOCTYPE>	Define o tipo de documento
<html>	Define um documento HTML
<title>	Define o título para o documento
<body>	Define o corpo do documento
<h1> a <h6>	Define os cabeçalhos HTML
<p>	Define um parágrafo
** **	Define uma quebra de linha
<hr>	Define uma mudança de tema do conteúdo

<!-- … -->	Comentários

3.2 Elementos de Formatação

Tag	Descrição
<abbr>	Define uma abreviação
<address>	Define a informação do autor/proprietário de um documento
<b>	Define um texto em negrito
<cite>	Define o título do trabalho
****	Define uma ênfase do texto
<q>	Define uma citação curta

3.3 Elementos de Formulários e Entradas

Tag	Descrição
<form>	Define um formulário de entrada para o usuário
<input>	Define um controle de entrada
<textarea>	Define um controle de entrada de múltiplas linhas
<button>	Define um botão clicável
<select>	Define uma lista de seleção drop-down
<optgroup>	Define um grupo de opções relacionadas em uma lista de seleção drop-down
<option>	Define uma opção da lista de seleção drop-down

<fieldset>	Grupo relacionado de elementos no formulário

3.4 Elementos de Imagens

Tag	Descrição
<img>	Define uma imagem
<map>	Define uma image-map no lado do cliente
<area>	Define uma área dentro de uma image-map

3.5 Elementos de Links

Tag	Descrição
<a>	Define um hyperlink
<link>	Define um relacionamento entre um documento e um recurso externo

3.6 Elementos de Listas

Tag	Descrição
** **	Define uma lista não ordenada
** **	Define uma lista ordenada
** **	Define um item da lista
<dl>	Define uma lista de descrição
<dt>	Define um termo/nome em uma lista descrição
<dd>	Define a descrição de um termo/nome em uma lista de descrição

<menu>	Define uma lista/menu de comandos

14

3.7 Elementos de Tabelas

Tag	Descrição
<table>	Define uma tabela
<thead>	Grupo de cabeçalho contido na tabela
<tbody>	Grupo de corpo contido na tabela
<tfoot>	Grupo de rodapé contido na tabela
<th>	Define uma célula de cabeçalho na tabela
<tr>	Define uma linha da tabela
<td>	Define uma célula da tabela

15

4 - Exemplos de códigos em HTML

Clique sobre os links para visualizar os exemplos. Cada arquivo fonte demonstra o uso de uma tag de forma simples e resumida.

- formulário
- hyperlink imagem
- hyperlinks
- imagem
- index
- lista de descrição
- tabela

16

RESUMO

Neste módulo caracterizamos a linguagem HTML como sendo uma linguagem de programação com limitações de representação. Vimos que HTML é uma linguagem de marcação, que se utiliza de elementos denominados de tags, que servem para estruturar o conteúdo e dizer como o referido conteúdo deve ser interpretado e apresentado na interface.

Vimos também que há linguagens compiladas, interpretadas e híbridas e demonstramos aqui os principais elementos envolvidos na construção de documentos web: básicos, formatação, formulários e entradas, imagens, links, listas e tabelas.